Surrogate-based constrained multi-objective optimization for the compression of CNNs

Gabriel Bicalho Ferreira [1], **Antônio de Barros** [1], **Issah Ibrahim**[2], **Rodrigo Silva** [1]

[1]Department of Computer Science – Universidade Federal de Ouro Preto
Ouro Preto – MG – Brazil

[2]Department of Electrical and Computer Engineering – McGill University
Montreal – QC – Canada.

gabriel.bicalho1@aluno.ufop.edu.br

Abstract. *The deployment of deep neural networks on devices with limited resources presents significant challenges. To address this issue, techniques such as pruning and quantization are frequently used. However, these techniques require careful tuning to ensure that the final model is computationally efficient without sacrificing too much quality.To automate the compression process and improve its robustness and affordability, we propose a Surrogate-based Multi-objective Constrained Evolutionary Algorithm for compressing artificial neural networks. Experimental results obtained for the CIFAR10 dataset using the ResNet50 and VGG16 architectures indicate that the proposed approach enables more efficient tuning related to pruning and quantization algorithms, resulting in higher quality non-dominated sets compared to the plain optimization method. Furthermore, our method produced leaner versions of the tested models while maintaining accuracy.*

Resumo. *A implantação de redes neurais profundas em dispositivos com recursos limitados apresenta desafios significativos. Para enfrentar esse problema, técnicas como poda e quantização são frequentemente utilizadas. No entanto, essas técnicas exigem ajustes cuidadosos para garantir que o modelo final seja computacionalmente eficiente sem sacrificar muita qualidade. Para automatizar o processo de compressão e melhorar sua robustez e acessibilidade, propomos um Algoritmo Evolucionário Multiobjetivo com Restrições Baseado em Surrogates para a compressão de redes neurais artificiais. Resultados experimentais obtidos para o conjunto de dados CIFAR10 utilizando as arquiteturas ResNet50 e VGG16 indicam que a abordagem proposta permite um ajuste mais eficiente em relação aos algoritmos de poda e quantização, resultando em conjuntos não dominados de maior qualidade em comparação com o método de otimização sem Surrogates. Além disso, nosso método produziu versões mais enxutas das arquiteturas testadas mantendo a precisão.*

1. Introduction

Convolutional Neural Networks (CNNs) have proven to be highly effective in various real-world tasks involving visual data processing, such as object detection, image and video processing [Pouyanfar et al. 2018].

Recent advances in these techniques have led to an increase in the accuracy of CNNs at the cost of a significant increase in their size, which in turn increases their complexity [Liang et al. 2021]. These complex structures pose a major challenge for deployment on edge devices such as smartphones and microcontrollers, which have limited processing and memory resources [Goldbarg 2021]. They are also a challenge in distributed [Verbraeken et al. 2020] and federated learning [Li et al. 2020] since this models must be sent through the network, where there are concerns about bandwidth costs.

In this context, an area of research that is gaining increasing prominence is the field of neural network compression techniques, in particular, the techniques of pruning and quantization. In pruning, some less significant neuron connections are removed from the model. In quantization, the precision of the model parameters is reduced. In [Liang et al. 2021], a review study compares pruning and quantization strategies, and assess their impact on neural networks. By applying these techniques, the size of the model an the number of computations can be greatly reduced. Overall, it shows that different types of pruning showed varying trade-offs in terms of compression, accuracy, and speedup. On the other hand, quantization demonstrated significant improvement in performance and reduced storage requirements.

Many decisions must be taking when applying pruning and quantization. For instance, the amount of connections to be pruned and the layers to be quantized. In the mentioned studies, it has become clear that the compression problem is really a multi-objective optimization problem in the sense that it has to identify optimal trade-offs between model computational performance and quality.

The ANN compression problem has been treated as a multi-objective optimization problem in [Hong et al. 2020, Chen et al. 2021, Xu et al. 2021, Wang et al. 2021, Hirsch and Katz 2022, Fernandes Jr. and Yen 2021]. With the exception of [Hirsch and Katz 2022], that solves the problem with reinforcement learning, all the other studies use some type of evolutionary algorithm (EA) as optimizer. In [Hong et al. 2020, Wang et al. 2021] and [Chen et al. 2021], multi-objective EAs, such as NSGA-II and MOEA/D, are used. In [Fernandes Jr. and Yen 2021], to alleviate the computational cost of a multi-objective evolutionary algorithm, an Evolutionary Strategy is used optimize three subproblems, called Knee, Hard Boundary and Light Boundary. Thus, it does not look for other trade-offs in the quality $\times$ complexity space. Finally, in [Xu et al. 2021] a single objective genetic algorithm is used to minimize the model loss while the other objectives are treated as constraints. Again, this avoids the computational cost of a multi-objective evolutionary algorithm.

EAs do not require that objective functions and constraints are provided in algebraic form. Besides, since they evolve a population of solutions in parallel, each solution can be mapped to a different trade-off among the objectives [Coello Coello et al. 2020]. These characteristics make them suitable for the neural network compression problem. On the other hand, they tend to require a large amount of objective function and constraints evaluations which is challenging in the ANN compression case since the inference and retraining are computationally demanding.

The use of evolutionary algorithms for problems with expensive function evaluations is a well-known concern [Coello Coello et al. 2020]. A well established tech-

nique to deal with this issue is the use of surrogate models also called meta-models [Silva 2018, Tong et al. 2021]. The idea is, instead of using the expensive objectives and constraint functions, one should build a computationally cheap model of them and use this cheap model to guide the optimization process.

Thus, to automate the decisions related to the pruning and quantization algorithms and make the compression process more robust and affordable in terms of execution time, we propose a Surrogate-based Multiobjective Constrained Evolutionary Algorithm for the ANN compression problem. The main contributions of this work can be defined as follows:

1. A constrained multi-objective algorithm to tackle the problem of compressing artificial neural networks (ANN). Our approach differs from existing multi-objective approaches in the literature, such as [Hong et al. 2020], [Wang et al. 2021] and [Chen et al. 2021], by incorporating quality constraints into the optimization process. This enables us to avoid evaluating poor solutions while still exploring trade-offs among multiple objectives.
2. A surrogate-based approach (SBA) to address the problem of compressing artificial neural networks (ANN). By using an SBA the algorithm can reduces the computational cost of the multi-objective optimization process in comparison to directly using an evolutionary algorithm. By employing surrogate models, it possible to efficiently search the solution space and approximate the Pareto front of the multi-objective optimization problem.
3. A study on the selection of surrogate models for the problem of compressing two popular architectures of convolutional neural networks (VGG16 and ResNet50) applied to the CIFAR10 dataset. While there is ample literature on the use of surrogate-based optimization (SBO) in single-objective unconstrained problems, the literature on SBO for multi-objective and constrained problems is limited, as noted by Tong et al. [Tong et al. 2021]. In [Runarsson 2004] surrogate selection is addressed for constrained single objective problems and in [Silva 2018] for multi-objective unconstrained problems. Since the ANN compression problem studied here is multi-objective and constrained, we explore various surrogate modeling techniques and evaluate their effectiveness in solving it.

2. Artificial Neural Network Compression as a Multi-objective Optimization Problem.

As we have seen, deep artificial neural networks are robust, complex, and highly effective structures for solving various types of problems. However, their high computational cost can hinder their implementation on different devices [Wang et al. 2021]. In this section, we provide a brief overview of two compression techniques, pruning and quantization, including the relevant parameters that directly influence the final performance and quality of the compressed model.

2.1. Pruning

Pruning can be defined as a compression technique for neural networks that involves selectively removing connections or neurons with low importance scores while preserving the network's accuracy [Liang et al. 2021, Vadera and Ameen 2022]. The importance

score of a connection or neuron can be based on various criteria, such as their weight magnitude, sensitivity, or activation level [Vadera and Ameen 2022]. The pruning process typically involves three steps: (1) determining the importance scores of each connection or neuron, (2) removing the connections or neurons with the lowest scores, and (3) fine-tuning the remaining network to restore its accuracy. Pruning can be applied iteratively to achieve higher compression rates, and it can be combined with other compression techniques, such as quantization, to further reduce the network's size and complexity.

2.2. Quantization

The quantization of an artificial neural network is a model compression technique that reduces the number of bits required to represent each parameter of an ANN [Han et al. 2015]. Generally, the parameters of an ANN are represented by 32-bit floating points, which increases the storage cost and complexity of calculations. In addition to the benefits of making the model faster and reducing storage costs, low-precision parameters can still store enough information for model inference while maintaining accuracy [Wang et al. 2021].

2.3. Compression Design Variables

Compression techniques require the definition of specific parameters that may vary according to the adopted strategy. In pruning, it is possible to remove weights, biases, or both simultaneously in a certain layer. The parameters to be defined include the type of pruning (weight or bias), the layer to be pruned, and the intensity (amount of connections removed). Additionally, it is possible to quantize previously pruned structures. To this end, design variables can be defined in order to optimize the compression process to obtain the most adequate resultant model. Table 1 defines the domain for each of the selected design variables.

Table 1. Compression Design Variables

Variable Code	Definition	Domain
x_1	Linear Layer Pruning	$x_1 \in \{0, 1, 2\}$
x_2	Convolutional Layer Pruning	$x_2 \in \{0, 1, 2\}$
x_3	Pruning Intensity for the Linear Layer	$0 < x_3 < 1$
x_4	Pruning Intensity for the Convolutional Layer	$0 < x_4 < 1$
x_5	Pruning Type for the Linear Layer	$x_5 \in \{0, 1, 2\}$
x_6	Pruning Type for the Convolutional Layer	$x_6 \in \{0, 1, 2\}$
x_7	Quantization	$x_7 \in \{0, 1\}$

Each variable presented in Table 1 has a domain in which each value represents a compression strategy. Variables one through six refer to compression by pruning, with x_7 being the representing whether the quantization is applied. The value 0 indicates no quantization and the value 1 represents the application of dynamic quantization.

2.4. Objective Functions

As mentioned in the Introduction, the compression problem can be seen as a multi-objective optimization problem which aims at finding solutions with a good balance between the quality and performance (execution time and space) of the resultant model.

More specifically, in this work, the compression problem is formulated as folows:

$$\begin{aligned} & \min_{\mathbf{x}} f_1(\mathbf{x}, \text{model}) : \text{non-zero weights rate} \\ & \max_{\mathbf{x}} f_2(\mathbf{x}, \text{model}) : \text{accuracy} \\ s.t. \quad & \\ & f_2(\mathbf{x}, \text{model}) \geq \tau \end{aligned} \tag{1}$$

where, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$.

The non-zero weights rate f_1 is related to the sparsity of the weights and biases tensors in neural network model. Minimizing this rate means increasing compression levels, which can be desirable in terms of computational efficiency and resource utilization. Conversely, maximizing the objective (f_2) enhances the overall quality of the neural network, as measured by relevant performance metrics such as accuracy.

When analyzing the problem of neural network compression in the literature, it is possible to observe a conflict between the amount of pruning performed and the accuracy obtained. This conflict is demonstrated in studies such as [Wang et al. 2021] and [Liang et al. 2021]. Thus, this paper addresses a multi-objective compression problem that involves seven variables 2.3, two objectives, and an accuracy constraint which aims to avoid spending computational resources with the evaluation of poor solutions.

3. Two-Archive Evolutionary Algorithm for Constrained Multiobjective Optimization (CTAEA)

Constrained multi-objective problems require a balance between convergence, diversity, and solution feasibility [Li et al. 2018]. In this sense, the algorithm chosen to optimize surrogate models was CTAEA, considered state of the art for solving restricted multi-objective problems [Li et al. 2018, Blank and Deb 2020].

The algorithm starts by randomly generating an initial population of solutions, which are evaluated based on their fitness with respect to the multiple objectives and constraints. The solutions are then divided into two sets, one for the Convergence Archive (CA) and the other for the Diversity Archive (DA), based on their fitness and diversity.

The CA is responsible for maintaining a set of solutions that are considered to be Pareto-optimal or near-Pareto-optimal, in terms of convergence towards the optimal solution. This archive is updated by selecting the best solutions from the current population and storing them if they are not dominated by any other solution in the archive.

The DA is responsible for maintaining a set of solutions that covers different regions of the search space and ensure diversity in the population. This archive is updated by selecting solutions that are diverse from the current population and storing them if they are not dominated by any other solution in the archive.

In each iteration, the algorithm performs a selection process to choose parents for the crossover and mutation operators. The offspring solutions are evaluated, and those that satisfy the constraints are added to the population. The population is then divided into the CA and the DA, and the process is repeated until the termination criteria are met.

One of the advantages of the CTAEA is its ability to handle problems with constraints, which is achieved by selecting feasible solutions and penalizing infeasible solutions. Additionally, the algorithm is designed to maintain a balance between convergence and diversity, which helps to avoid premature convergence and ensure that a good set of solutions is obtained.

4. Multiobjective constrained optimization based on surrogate models.

The proposed method creates one surrogate model for each the of the objectives and constraints of the compression problem. Firstly, a random set of the search space is sampled and evaluated using the original objective and constraint functions. Then a set of candidate surrogate models are generated for each objective and constrained function and the best surrogate from the set (given a quality metric M_o for the objective and M_c for the constraints) is selected. With the selected models, it is possible to build a surrogate problem similar to the original, which will be optimized by the CTAEA. After each optimization run, a set of feasible and non-dominated solutions is generated. From this set, k sample points are randomly selected and evaluated in the original problem. Finally new surrogate models are generated with the updated sample set and methods is repeated until the the budget of original problem evaluations is over. The method returns the set of all non-dominated points which were evaluated in the original problem over the method iterations.

In the next section, the metrics used for selecting surrogate models for objectives and constraints are detailed.

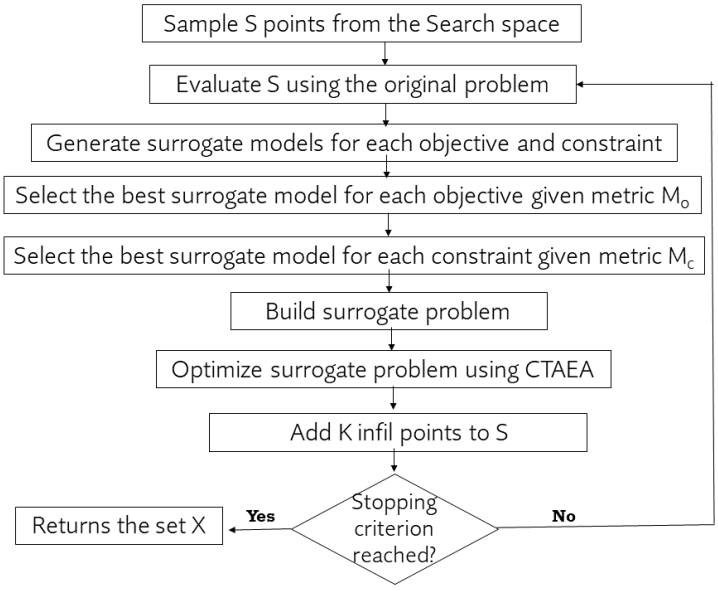

Figure 1. Surrogate-based optimization

4.1. Surrogate model selection

Selecting an appropriate model for the objective functions and constraints is a critical step for the performance of surrogate-based optimization algorithms. However, this task is not straightforward, as each function in the original problem has unique characteristics

that may require a different surrogate model. To address this, a set of surrogate models is trained at each iteration, and the best one is chosen based on a given metric. The question then arises as to what the best metric is for selecting surrogates for our problem. While this issue has been previously addressed in [Runarsson 2004] for single objective constrained problems and in [Silva 2018] for unconstrained multi-objective problems, the compression problem is both constrained and multi-objective. Therefore, revisiting this question seems necessary.

4.2. Metrics for surrogate model selection

4.2.1. The Mean Squared Error (MSE)

It is a statistical metric used to measure the average squared differences between the predicted and the actual values. Equation 2 shows how it is computed.

$$\frac{1}{n}\left(\sum_{i=0}^{n-1}(Y_i - Y_{pi})^2\right) \tag{2}$$

Where n represents the number of samples, Y_i is the actual value of each sample, and $\hat{Y}_i$ is the predicted value.

4.2.2. Mean absolute percentage error (MAPE)

It is a statistical metric used to measure the average absolute percentage differences between the predicted values and the actual values in a dataset. It is mathematically defined by Equation 3.

$$\frac{1}{n}\left(\sum_{i=0}^{n-1}\frac{|Y_i - \hat{Y}_i|}{MAX(\epsilon, |y_i|)}\right) \tag{3}$$

Where n represents the number of samples, Y_i is the actual value, $\hat{Y}_i$ is the predicted value, and ϵ is a very small positive value to avoid division by zero.

4.2.3. Spearman

Spearman correlation is a statistical measure of the strength and direction of the monotonic relationship between two variables. Unlike Pearson correlation, which measures the linear relationship between two variables, Spearman correlation evaluates how well the relationship between two variables can be described using a monotonic function, which is a function that either consistently increases or decreases. The Spearman correlation coefficient, denoted by the symbol rho (ρ), ranges between -1 and 1, where a value of -1 indicates a perfect negative monotonic relationship, a value of 1 indicates a perfect positive monotonic relationship, and a value of 0 indicates no monotonic relationship between the two variables. It can be mathematically defined as follows:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{4}$$

where d_i is the difference between the ranks of the i-th observation of the first variable and the i-th observation of the second variable, and n is the number of observations.

5. Experimental setup

This section outlines the methodology used to evaluate the behavior and effectiveness of the proposed method.

5.1. Dataset

In this work, the CIFAR dataset [CStoronto sd] is used as testbed for the proposed methodology. The CIFAR is a collection of labeled images used for object recognition tasks. The name CIFAR stands for "Canadian Institute for Advanced Research," where the dataset was first created. There are several versions of the CIFAR dataset, but the most commonly used are CIFAR-10 and CIFAR-100.

CIFAR-10, used in this work, contains 60,000 32x32 color images in 10 classes, with 6,000 images per class. The classes include common objects such as airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

5.2. Artificial Neural Networks Architectures

In the same way as the dataset, the architectures were chosen according to their usage in the literature, which facilitates comparison among different works. The selected architectures were VGG16 [Simonyan and Zisserman 2014] and Resnet50 [He et al. 2016], which will be explored in this section.

5.2.1. VGG16

It is a deep convolutional neural network model that has 13 convolutional layers and 3 fully connected layers with more than 138 million parameters [Sugata and Yang 2017].

The VGG16 architecture uses the ReLu (Rectified Linear Unit) activation function, which returns 0 for negative values and the value itself for positive inputs. In addition, fixed-size 3x3 convolutional masks are used in the convolution process, which are moved in windows with a 1x1 spacing. When necessary, padding of size 1x1 is applied to the edges of the input image. The final output of VGG16 is a probability distribution defined by the softmax function.

5.2.2. Resnet50

Like the VGG16, the Resnet50 is a deep convolutional neural network that uses the ReLu activation function. However, there are significant differences between the architectures. Resnet50 has 50 layers, 49 of which are convolutional and only one is fully connected. The convolutional filters follow the standard size of 3x3 [Bendjillali et al. 2020], and after the convolution process, data normalization is performed.

Despite having a larger number of layers than the VGG16 architecture, Resnet50 has fewer parameters, making it less complex [Bendjillali et al. 2020]. An estimated value of less than 26 million parameters is assigned to this architecture.

The settings used for training both architectures are: (i) Epochs: 40; (ii) Learning rate: $1e-3$; (iii) L2 regularization penalty: $1e-3$; (iv) Batch size: 50; (v) Loss function: Cross-entropy; (vi) Optimizer: Adam.

5.3. Experimental procedure

The proposed approach was tested as follows:

1. The untrained versions of VGG16 and Resnet50 from PyTorch [PyTorch 2018] are loaded.
2. Both architectures are trained with CIFAR10 training set.
3. Five independent executions of the procedure presented in Section 4 are performed to solve the compression problem defined by Eq. 1. The parameters used in all the runs were:
   (a) Initial sampling: 50 function evaluations
   (b) Optimizer: CTEAE
   (c) Number of generations: 1000
   (d) Population size: 50
   (e) Evaluation function budget: 100 (approximately 10 hours)
   (f) Infills per iteration: 2
   (g) Surrogate models: KNeighborsRegressor [Pedregosa et al. 2011], RandomForest [Pedregosa et al. 2011], XGBoost [Chen and Guestrin 2016], LightGBM [Ke et al. 2017]
   (h) Surrogate selection metric: MSE, MAPE, Spearman, Random
   (i) Number of independent runs: 5 for each surrogate selection metric.
4. The 4 surrogate selection metrics presented in Section 4.2 were tested for the selection models for objectives and constraints. While all the objectives used the same metric at a time, a different metric was allowed for the constraint given rise to 16 different configurations.
5. The quality of each compressed solution is assessed with CIFAR10 test set.
6. The final set of all the evaluated solutions as well as the hypervolume of the non-dominated set are stored for each run. The hypervolume can be defined as:

   Definition 1 (Hypervolume Indicator). Given a point set $S \subseteq \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$, the hypervolume indicator of S is the measure of the region weakly dominated by S and bounded above by r, i.e.:

   $$H(S) = \Lambda(\{q \in \mathbb{R}^d | \exists p \in S : p \leq q \text{ and } q \leq r\}) \tag{5}$$

   where $\Lambda(\cdot)$ denotes the Lebesgue measure.
7. The Friedman test was used to compare the different algorithms and algorithm versions and, when necessary, the Dunn's test was used as a post-hoc. The adopted confidence level α was set to 0.05.

To evaluate the effectiveness of the approach, we conducted a test using the pure CTAEA without employing surrogate models. The experiment was performed within a budget of 100 function evaluations, which roughly translates to 10 hours of CPU time on the machines described in Table 2. It should be noted that due to the limitations of the current quantization implementation, GPUs were not utilized in this study [PyTorch 2023].

Table 2. Hardware Setting

Hardware	Configuration
CPU	Intel(R) Xeon(R) 2.200 GHz
System RAM	12.68 GB
Disk	107.72 GB
CPU Frequency	2199.998 MHz

The pure CTAEA was tested in two different configurations: (i) 10 generations with a population size of 10, and (ii) 20 generations with a population size of 5. This choice was made to minimally strike a balance between having a sufficiently large population size and providing enough generations for the population to evolve and improve.

6. Results

In this section, we analyze the performance of the proposed approach across three aspects. Firstly, we evaluate the effect of the surrogate model selection metric in Figures 2, 3, 4 and 5. Secondly, we compare the proposed surrogate-based method with the raw versions of the optimizer in Figures 6 and 7. Lastly, we assess the improvement caused by the algorithm by comparing the final set of evaluated solutions with the initial deep learning model in Figures 8, 9, 10 and 11.

6.1. Effect of the surrogate selection metric

Figures 2, 3, 4 and 5 shows the box-plots for the hypervolume of the obtained non-dominated solutions as a function of the surrogate selection metric for the models VGG16 and Resnet50. It can be seen that the proposed method did not show a significant difference in performance across the various selection metrics for both models. This contrasts with the findings in [Silva 2018] and [Runarsson 2004], who suggest the use of metrics based on the preservation of solution ranks, such as Spearman's correlation. However, it is important to note that the budget for evaluating the original problem in this study is much smaller than in previous studies. For instance, [Runarsson 2004] allowed for a budget of 10,000 evaluations, while [Silva 2018] utilized a budget of 500 evaluations in their experiments.

Given the limited number of function evaluations, the results in Figures 2, 3, 4 and 5 suggest that the choice of surrogate selection metric has a negligible impact. In fact, in some cases, it may even be worse than random because the amount of available information is insufficient for accurate evaluation of the surrogate models. For example, Figure 3 demonstrates that the random selection of surrogates for the objective functions resulted in sets of non-dominated solutions with larger hypervolumes, on average.

6.2. Effect of the use of surrogate models

In this section, we aim to determine whether the surrogate-based approach confers an advantage over the plain version of the CTAEA by performing a comparison between the two methods.

Figures 6 and 7 displays the box-plots for the surrogate-based configurations with the highest and lowest hypervolumes, on average, as well as the two plain CTAEA con-

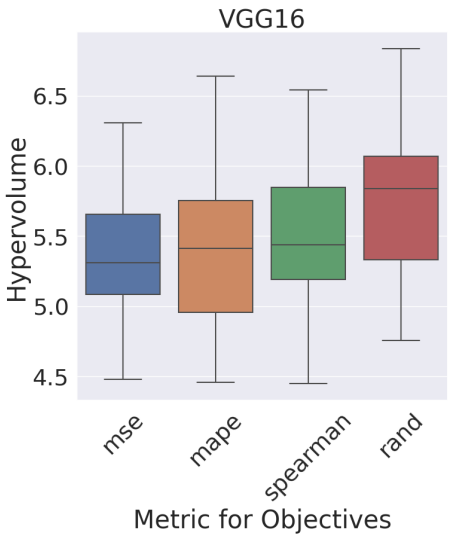

Figure 2. Surrogate selection metric for the objective functions - VGG16

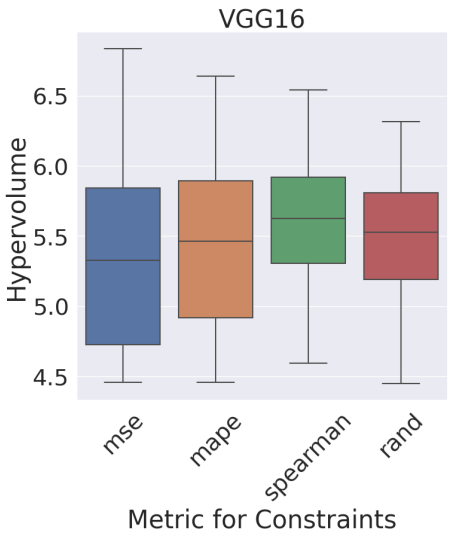

Figure 3. Surrogate selection metric for the constraint function - VGG16

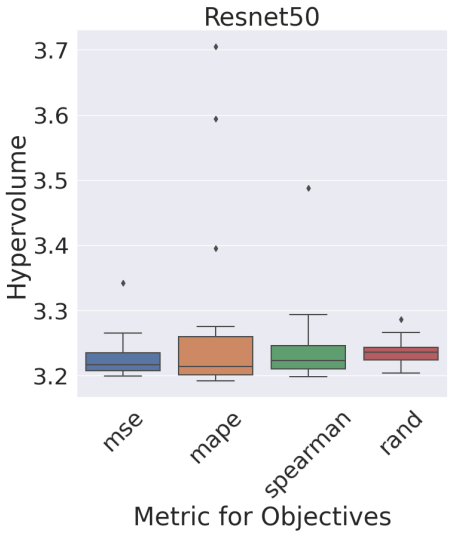

Figure 4. Surrogate selection metric for the objective functions - Resnet50

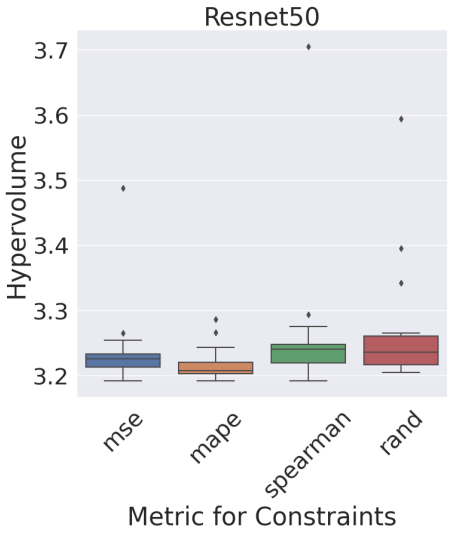

Figure 5. Surrogate selection metric for the constraint function - Resnet50

figurations. Specifically, CTAEA1 employs 20 individuals that undergo 10 generations of evolution, while CTAEA2 uses 10 individuals that undergo 20 generations of evolution.

Regarding the VGG16, as shown in Figure 6, it can be observed that the worst performing surrogate-based approach used MAPE for objective surrogate selection and MSE for constraint selection. Conversely, the best performing version selected the objective surrogates randomly and used MAPE for the constraints. CTAEA2, which undergoes more generations of evolution, demonstrated competitiveness with the worst performing surrogate-based version. However, the best performing surrogate-based version outperformed all other contenderS ($p < 0.05$). CTEA1, on the other hand, performed poorly in this particular scenario.

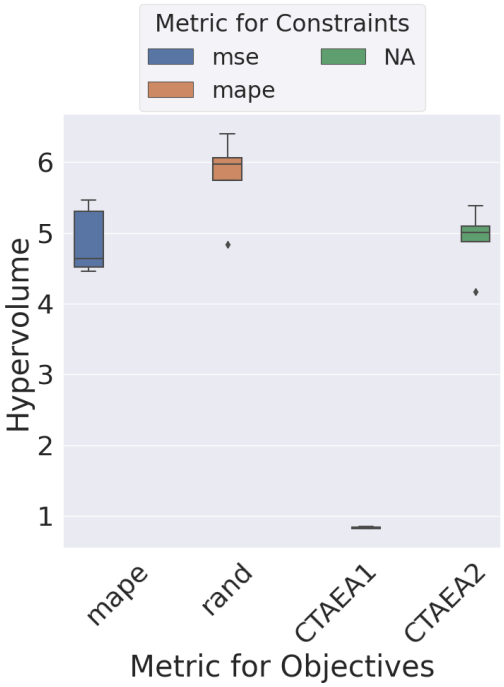

Figure 6. Comparison between the surrogate-based and the raw version of CTAEA - VGG16

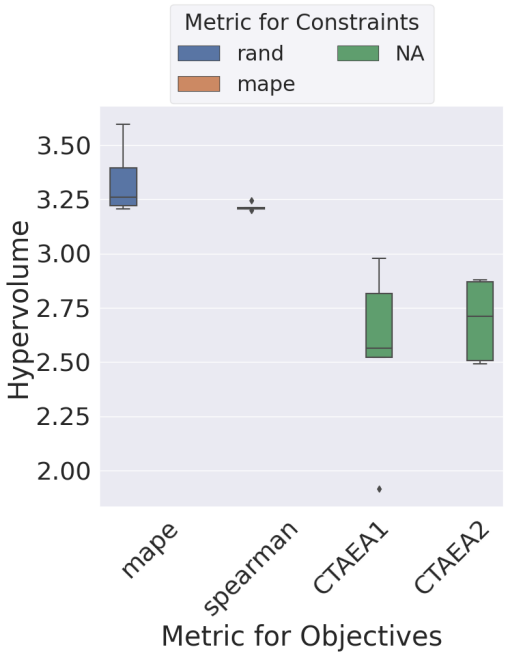

Figure 7. Comparison between the surrogate-based and the raw version of CTAEA - Resnet50

Concerning the Resnet50, as depicted in Figure 7, no statistical difference ($p \geq 0.05$) was observed between the surrogate-based approaches, and both were superior ($p < 0.05$) to the plain versions of the CTAEA. Interestingly, no statistical difference was observed between the two CTAEAs in this instance.

6.3. Final solutions

Figures 8, 9, 10 and 11 shows the non-dominated solutions with the lowest hypervolume achieved by the best surrogate-based settings, compared to the original model. Figures 8 and 10 illustrate that even the worst set of the best version of the proposed method could set a considerable number of weights to zero while preserving accuracy close to the original for VGG16 and Resnet50, respectively. Moreover, in some solutions, the proposed method improved accuracy by, possibly, reducing overfitting.

The test set inference time of the compressed models is shown in Figures 9 and 11. Although there is no direct relationship with the zeros rate, it is apparent that the compressed models have significantly faster inference times.

7. Conclusion

The goal of the work is to show the application of a surrogate-based optimization technique to handle the parametrization of the quantization and prunning for the compression of artificial neural networks. Further studies remain to be done in more datasets using more architectures. Nevertheless, the results obtained so far indicate that, given the same computational budget, the best surrogate-based versions outperform the plain optimizer. Besides, the obtained set of solutions were much leaner an did not lose a lot of accuracy when compared with the pre-compression model on the tested dataset. In summary,

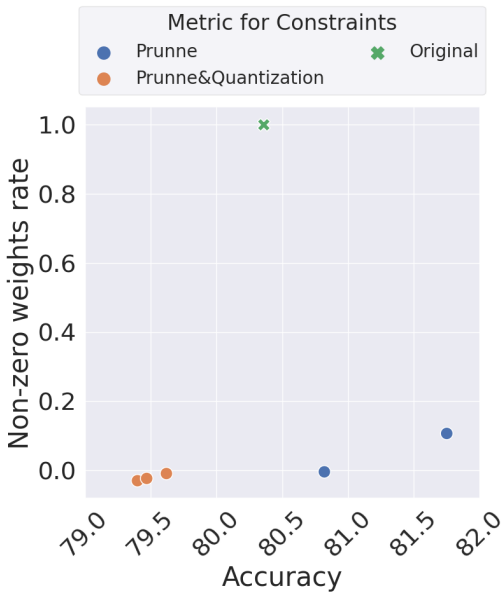

Figure 8. VGG16 - Objective function space

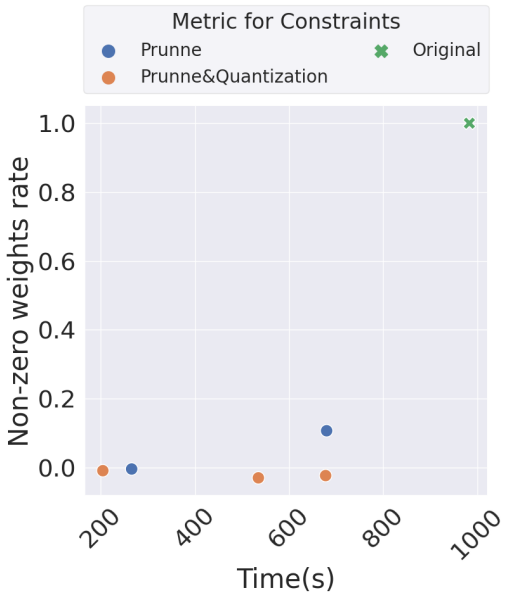

Figure 9. VGG16 - Test set inference Time by Non-zero weights rate

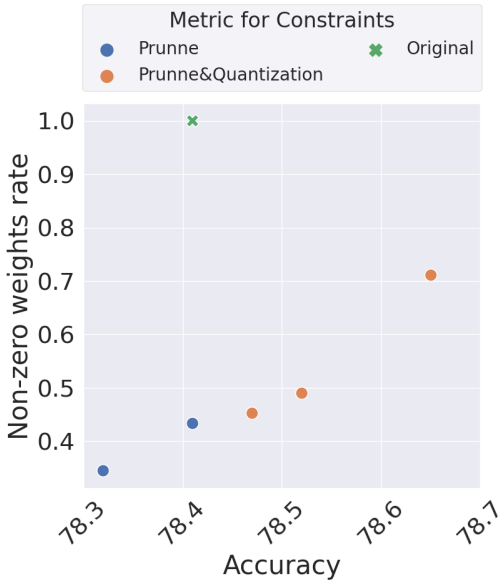

Figure 10. Resnet50 - Objective function space

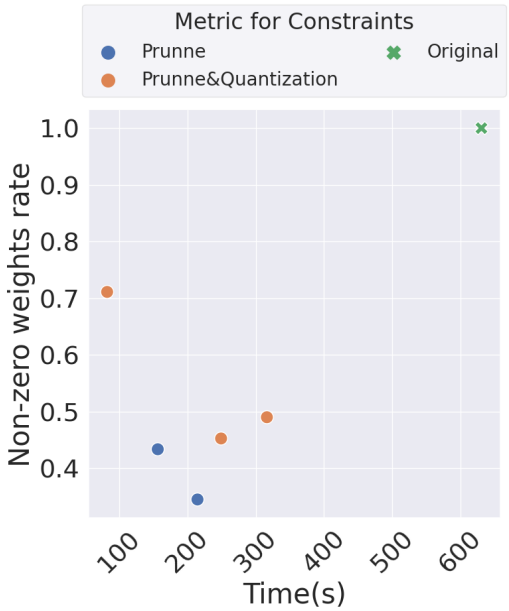

Figure 11. Resnet50 - Test set inference Time by Non-zero weights rate

our proposed approach offers a promising solution for the compression of deep neural networks.

Acknowledgments

This work was supported by CNPq - National Council for Scientific and Technological Development, CAPES - Coordination for the Improvement of Higher Education Personnel, UFOP - Federal University of Ouro Preto.

ChatGPT (https://chat.openai.com/) was used to improve language throughout the text. The "Revise" prompt was used with paragraphs of pre-existing English text as input. The responses generated by ChatGPT were then reviewed by the authors for their accuracy and meaning before being integrated back into the text.

References

Bendjillali, R. I., Beladgham, M., Merit, K., and Taleb-Ahmed, A. (2020). Illumination-robust face recognition based on deep convolutional neural networks architectures. *Indonesian Journal of Electrical Engineering and Computer Science*, 18(2):1015–1027.

Blank, J. and Deb, K. (2020). pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.

Chen, J., Xu, Y., Sun, W., and Huang, L. (2021). Joint sparse neural network compression via multi-application multi-objective optimization. *Applied Intelligence*, pages 1–18.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. https://xgboost.readthedocs.io. Acess in: 04/20/2023.

Coello Coello, C. A., González Brambila, S., Figueroa Gamboa, J., Castillo Tapia, M. G., and Hernández Gómez, R. (2020). Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 6:221–236.

CStoronto (s.d). The CIFAR-10 dataset. Accessed: 2023-04-04.

Fernandes Jr., F. E. and Yen, G. G. (2021). Pruning deep convolutional neural networks architectures with evolution strategy. *Information Sciences*, 552:29–47.

Goldbarg, M. A. S. d. S. (2021). Análise de técnicas de compressão em redes neurais profundas por poda em dataset de imagens. B.S. thesis, Universidade Federal do Rio Grande do Norte.

Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hirsch, L. and Katz, G. (2022). Multi-objective pruning of dense neural networks using deep reinforcement learning. *Information Sciences*, 610:381–400.

Hong, W., Yang, P., Wang, Y., and Tang, K. (2020). Multi-objective magnitude-based pruning for latency-aware deep neural network compression. In Bäck, T., Preuss, M., Deutz, A., Wang, H., Doerr, C., Emmerich, M., and Trautmann, H., editors, *Parallel Problem Solving from Nature – PPSN XVI*, pages 470–483, Cham. Springer International Publishing.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., and Ye, Q. (2017). LightGBM: A highly efficient gradient boosting decision tree. https://lightgbm.readthedocs.io. Acess in: 04/20/2023.

Li, K., Chen, R., Fu, G., and Yao, X. (2018). Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315.

Li, L., Fan, Y., Tse, M., and Lin, K.-Y. (2020). A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854.

Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36.

PyTorch, A. D. I. (2018). Pytorch.

PyTorch, Q. (2023). QUANTIZATION TUTORIAL. Accessed: 2023-06-29.

Runarsson, T. P. (2004). Constrained evolutionary optimization by approximate ranking and surrogate models. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 401–410, Berlin, Heidelberg. Springer Berlin Heidelberg.

Silva, R. C. P. (2018). *Surrogate problem evaluation and selection for optimization with expensive function evaluations*. McGill University (Canada).

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sugata, T. and Yang, C. (2017). Leaf app: Leaf recognition with deep convolutional neural networks. In *IOP Conference Series: Materials Science and Engineering*, volume 273, page 012004. IOP Publishing.

Tong, H., Huang, C., Minku, L. L., and Yao, X. (2021). Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. *Information Sciences*, 562:414–437.

Vadera, S. and Ameen, S. (2022). Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. (2020). A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2).

Wang, Z., Luo, T., Li, M., Zhou, J. T., Goh, R. S. M., and Zhen, L. (2021). Evolutionary multi-objective model compression for deep neural networks. *IEEE Computational Intelligence Magazine*, 16(3):10–21.

Xu, K., Zhang, D., An, J., Liu, L., Liu, L., and Wang, D. (2021). Genexp: Multi-objective pruning for deep neural network based on genetic algorithm. *Neurocomputing*, 451:81–94.