Neural Architecture Search for Efficient Mixed-Precision Quantization in Audio Denoising

Alessandro Cerioli
GN Audio
Technical University of Denmark
Ballerup, Denmark
Kongens Lyngby, Denmark
alcerioli@jabra.com
alceri@dtu.dk

Clément Laroche
Tobias Piechowiak
GN Audio
Ballerup, Denmark
claroche@jabra.com
topiechowiak@jabra.com

Luca Pezzarossa
Martin Schoeberl
Technical University of Denmark
Kongens Lyngby, Denmark
lpez@dtu.dk
masca@dtu.dk

***Abstract*—This work presents a neural architecture search approach based on evolutionary algorithms to tackle the problem of mixed-precision quantization considering 8, 16, and 32-bit depths for NSNet2, a neural network used for audio denoising tasks. The goal is to preserve high network performance while reducing the model size, making it suitable for deployment on embedded systems such as headsets. The neural architecture search (NAS) algorithm considers metrics such as (a) Perceptual Evaluation of Speech Quality (PESQ) to assess the audio quality after de-noising, (b) the inference time of the neural network, and (c) its memory footprint. Our results showcase remarkable improvements, with a 29.93% reduction in inference time and a 57.88% reduction in memory footprint compared to the full-precision *float32* baseline, while preserving high performance in audio de-noising with a negligible drop of PESQ metric. This work provides a framework for estimating the optimal mixed-precision quantization configuration, which is beneficial for deploying neural networks on resource-constrained devices.**

I. Introduction

Nowadays, the research area of deep learning for audio processing is gaining prominence, showing potential for industrial applications [1]. In embedded systems like headsets, it is crucial to define small-sized models. Audio de-noising is a widely used approach to improve signal quality by reducing undesired noise [2], [3]. Additionally, deep learning in audio includes areas like noise classification, source separation, and speech recognition [4], [5]. However, creating models effective for devices with limited computational resources requires compression techniques to reduce model size without significantly compromising performance.

Neural network compression techniques are central to tiny machine learning (TinyML) [6], which focuses on deploying models on resource-constrained devices like microcontrollers. Unlike internet of things (IoT), TinyML emphasizes on-device computation to ensure real-time performance and data privacy. This requires models with minimal memory and energy use, making compression techniques essential for deployment on such devices.

In TinyML, model compression is crucial to meet hardware constraints. Key techniques include pruning, knowledge distillation, and quantization. Pruning concerns removing the weights that contribute less to the network's output [7]. Instead, knowledge distillation involves using a large model (called a *teacher*) to teach a smaller model (called a *student*) [8]. Quantization lowers the precision of weights and activations from types like *float32* to *int8* or *float16*, which may slightly reduce accuracy but offers significant computational benefits.

Mixed-precision quantization is an advanced method of quantization that enables the quantization of different neural network layers at varying levels of precision [9], [10]. This strategy permits more aggressive compression of less sensitive layers while maintaining higher precision in critical layers, thus preserving high performance.

This paper presents a framework for finding the optimal mixed-precision quantization configuration of an NSNet2 for de-noising via an evolutionary algorithm-based neural architecture search algorithm. We minimized hardware-related metrics such as inference time and memory footprint while maintaining a de-noising quality almost equivalent to the baseline in *float32*. Furthermore, we maximize an audio-related metric (PESQ) for network quality evaluation instead of more generic metrics such as accuracy. The results demonstrated a notable improvement in hardware performance with a negligible drop in de-noising quality.

The contributions of this paper are the following:

- A framework based on neural architecture search with an evolutionary algorithm for estimating the optimal mixed-precision quantization configuration in the audio field domain, using hardware-related and specific sound quality metrics.
- A specific application of de-noising via mixed-precision quantization of an NSNet2, benchmarking the performance against *float32* full-precision and uniformly quantized baselines.

This paper comprises six sections: Section II provides the background, Section III offers a detailed description of the methodologies adopted, the content of the paper, and the related contribution, Section IV analyzes the results of the experiments, demonstrating the benefits of mixed-precision quantization. Section V presents the related work, comparing

our framework with the state-of-the-art. Finally, Section VI concludes the paper.

II. Background

A. Neural Architecture Search

Neural architecture search (NAS) is an automated system for designing efficient neural networks [11]. It is an optimization algorithm formulated to enhance network performance, i.e., maximizing accuracy and decreasing other computational metrics, such as training time, inference time, memory use, or energy consumption. The optimization of these metrics lies in the estimation of hyper-parameters or the generation of complete architectures. Typically, NAS algorithms optimize neural networks to make them suitable for deployment on embedded systems.

NAS methods adopt various optimization algorithms to detect optimal neural networks. Optimal solutions can be identified using reinforcement learning (RL) techniques [12], evolutionary algorithms (EA) [13], [14], gradient descent [15] and Bayesian optimization [16].

Genetic algorithms are a type of evolutionary algorithm characterized by a well-defined structure. Initially, the *population* (the set of evaluated solutions) is initialized randomly. Afterward, several generations are re-iterated, where each generation is composed of a selection of *parent* solutions, which will generate the offspring through a *crossover* phase and then carry out an evaluation through an objective function to establish which solutions are more fitting and therefore preserved in the next generation.

Evolutionary algorithms are robust to local minima as they maintain a population's diversity and can evaluate non-differentiable objective functions. However, they have slower convergence and a higher computational cost than gradient descent.

B. Mixed-Precision Quantization

The most common quantization methodology is linear quantization, which maps full-precision *float32* values to integer types via specific constants called *scaling factor* and *zero-point* obtainable through network calibration. The equation for linear quantization is in Equation 1.

$$\mathbf{W} = \lfloor \mathbf{Q} - z \rfloor \cdot s \tag{1}$$

Where $\mathbf{W}$ is a matrix in 32-bit floating point, $\mathbf{Q}$ is the quantized matrix, S is the *scaling factor*, and z is the *zero-point* [17].

Mixed-precision quantization is a more accurate form of quantization, where the various layers of the network are not all quantized with the same bit-width. Quantization can be more aggressive for less sensitive layers (representing them with fewer bits), while more sensitive layers require the maintenance of higher precision. This quantization allows a more optimal trade-off between the accuracy drop and the computational cost of the network compared to the case of uniform quantization.

C. NSNet2

NSNet2 is a neural network originally designed for real-time audio de-noising applications [18]. Its architecture consists of one fully connected embedding layer, followed by two gated recurrent unit (GRU) [19] layers, and finishing with four fully connected layers.

As input, the network takes pre-processed audio as input, i.e., a short-time Fourier transform (STFT) frame of 512 points, with an overlap of 50%. The output represents a mask for the element-wise multiplication with the original input to obtain the enhanced audio. Finally, the process converts the result to the original time domain using an inverse STFT.

D. Perceptual Evaluator of Speech Quality

Perceptual Evaluation of Speech Quality (PESQ) is a metric developed by ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) to evaluate the quality of speech signals after audio processing, such as noise suppression [20]. For this reason, PESQ is a widely used metric in the telecommunications sector and other audio-related contexts. It represents a prediction of the quality and clarity of the speech signal heard by a human.

The metric compares the processed speech signal with the unprocessed one (tainted by noise), and the score varies, ranging from 0.5 to 4.5, where a higher score indicates a better quality of the speech signal after processing.

III. Experimental Setup of Neural Architecture Search Algorithm

In this section, we outline the contributions of our project, including the software, pipeline, and experiment setup. In Subsection III-A, we describe the structure and setup of the NAS algorithm developed to acquire the Pareto optimal mixed-precision quantization configurations for the NSNet2. Subsection III-B exposes the different evaluation metrics used for the objective function and their implementation.

A. Evolutionary Algoritm Implementation

The NAS algorithm represents the core of our experiments to estimate the optimal mixed-precision quantization configuration. For the implementation, we used *Pymoo* [21], a widely exploited Python library for optimization problems.

First, the *chromosome* of the genetic algorithm is the sequence of bit-widths used for the different operators of the neural network. In the case of NSNet2, considering both the weights and the activations, the amount is approximately 2.8 million parameters distributed over 93 elementary operations that can be quantized separately. Therefore, the chromosome in the genetic algorithm consists of 93 genes, each capable of assuming one of three values: 8, 16, or 32. These values indicate the number of bits used for the diverse weights and activations of the model. The chromosome includes the various quantization bit-depth for the different elementary operations. For example, fully connected layers include two weights and one activation, while GRUs incorporate twelve weights and

twenty-two activations. The distinct levels of GRU are each internal operation, according to the equation 2.

$$\begin{aligned} \mathbf{z_t} &= \sigma(\mathbf{W_z} x_t + \mathbf{U_z} \mathbf{h_{t-1}} + \mathbf{b_z}) \\ r_t &= \sigma(\mathbf{W_r} x_t + \mathbf{U_r} h_{t-1} + \mathbf{b_r}) \\ \tilde{h}_t &= \tanh(\mathbf{W_h} x_t + r_t \odot \mathbf{U_h} \mathbf{h_{t-1}} + \mathbf{b_h}) \\ \mathbf{h_t} &= (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + z_t \odot \tilde{\mathbf{h_t}} \end{aligned} \tag{2}$$

Where $\mathbf{z_t}$ is the update gate, $\mathbf{r_t}$is the reset gate, $\tilde{\mathbf{h_t}}$ is the new hidden state, $\mathbf{h_t}$ is the final hidden state, $\mathbf{W}$ and $\mathbf{U}$ are weight matrices, $\mathbf{b}$ is the bias vector, concatenation of $(\mathbf{b_z}; \mathbf{b_r}; \mathbf{b_h})$, and $\odot$ denotes element-wise multiplication.

Similarly, regarding fully connected layers, we referred to the equation 3.

$$y = f(\mathbf{W}x + b) \tag{3}$$

Where f is the activation function, $\mathbf{W}$ is the weights matrix, and b is the bias vector.

The genetic algorithm's structure follows the classic structure, i.e., an initialization phase followed by a series of iterations (named *generations*). The generations improve the population's genome, eliminating the less suitable solutions and rewarding those most suited to solving the optimization problem.

The algorithm starts with a random population initialization, with the genome of each solution randomly assuming values 8, 16, and 32 for each of the 93 genes. The random initialization of the integers occurs via the defined *IntegerRandomSampling* function in Pymoo.

Subsequently, the parents' selection and the crossover phases occur, i.e., mixing the parents' genome to create a new solution or offspring. We used the *TwoPointCrossover* function defined in Pymoo. This function implements the *two points crossover* strategy, which consists of defining two random points on the parental genome, swapping the central part, and leaving the external parts unchanged. This technique is suitable for optimization problems on integer values.

The algorithm introduces a mutation method to vary the offspring's genome and increase genetic variability. We opted for a *polynomial mutation*, implemented in Pymoo through the *PolynomialMutation* method. The algorithm was performed on a population of size 25, considering 50 generations, without defining an alternative stopping criteria.

At the end of each generation, the process evaluates each solution via an *objective function*. This function receives the chromosome as input and returns a vector of metrics described in detail in section III-B.

B. Objective Function Metrics

1) Inference Time: To turn the NAS algorithm hardware-aware, we implemented a version of NSNet2 in C code. We selected C due to its ability to provide extensive control over memory, enabling the optimal mixed-precision quantization configuration on diverse hardware. Inference time is one of the metrics for evaluating mixed-precision quantization configurations. Therefore, it is part of the objective function of the NAS algorithm.

The input is the chromosome, i.e., the mixed-precision quantization configuration. Consequently, the algorithm generates a header file in C called *mpq.h*, containing mixed-precision quantization information. Primarily, it includes information on the bit-depth, the scaling factor, and the zero point for each of the quantized elementary operations of the NSNet2.

Subsequently, the compilation pipeline moves several pre-implemented C files into the same folder. These files are:

- *main.c*: an entry point where the benchmark of the average inference time takes place, over 10,000 iterations for statistical reliability.
- *npy_parser.c* and *npy_parser.h*: allow reading the weights saved in numpy format.
- *q_nsnet2.c* and *q_nsnet2.h*: the C implementation of NSNet2.

The C files are compiled into an executable to assess the inference time. This approach ensures accurate measurements of inference time by providing control over memory management and computational processes during evaluation. The average of the measurements is finally normalized. Normalization is essential to ensure the different metrics are on the same scale, avoiding privileging those with a higher magnitude. We normalized the inference time between the minimum inference (assessed with 8-bit uniform quantization) and the maximum inference (evaluated with 32-bit uniform quantization).

2) Memory Footprint: The memory footprint metric is the model's size, including weights and activations. It is an essential metric to discern whether the model is suitable for embedded systems, which are often memory-bound architectures. The evaluation occurs by computing the number of bits using Equation 4.

$$mf = \sum_{i=1}^{L} mpqc_i \cdot size\left(layer_i\right) \tag{4}$$

Where mf is the memory footprint, $mpqc_i$ is the configuration of the mixed-precision quantization, L is the number of elementary operations (i.e., 93 in the NSNet2), and $size(layer_i)$ is the number of parameters contained in the different quantization levels of the network.

The metric is normalized between a minimum value, assumed for uniform 8-bit quantization of the entire network, and a maximum value, inferred for uniform 32-bit quantization.

3) PESQ: Alongside hardware-related metrics, we introduced an audio quality metric to reduce the model's complexity and maintain a high denoising quality. This metric for the objective function represents a novelty introduced in this project. Generic parameters minimize the impact of quantization specifically on single layers or globally on the whole network (e.g., the accuracy). We opted for a metric tailored for applications in the audio field, specifically for

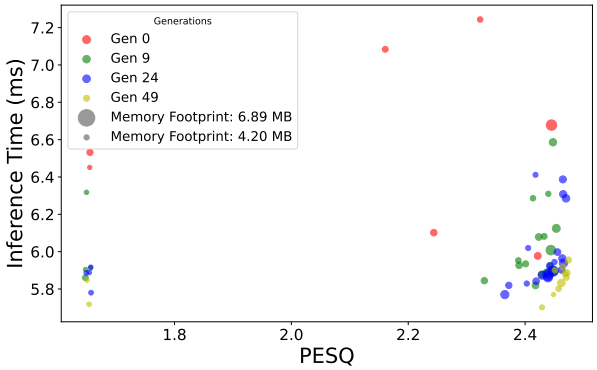

Fig. 1. Scatter plot showing the Pareto optimal evolution across selected generations (0, 9, 24, 49), with PESQ on the x-axis and inference time (ms) on the y-axis. Point color indicates generation, while point size reflects memory footprint.

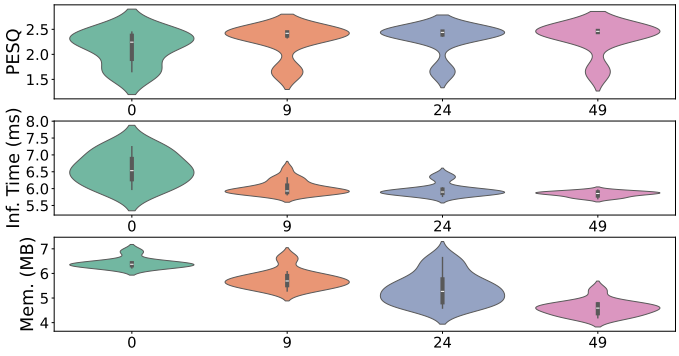

Fig. 2. Violin plot showing the distributions of PESQ, Inference Time (ms), and Memory Footprint (MB) inside the Pareto optimal across selected generations (0, 9, 24, 49) of the evolutionary algorithm.

denoising and speech enhancement. The PESQ metric can provide a reliable indicator regarding audio quality.

To evaluate PESQ on a quantized model, we developed an NSNet2 in PyTorch to simulate mixed-precision quantization. The model simulates all quantized operations such as *quantize* for quantization, *dequantize* for de-quantization, *quantize_mul* for element-wise multiplication, *quantize_add* for element addition-wise, *quantize_matmul* for matrix multiplication, and so on. Creating an instance of the model requires the chromosome of the genetic algorithm.

We used the Python *pypesq* [22] library to evaluate the audio quality. First, the clean audio file signal (or reference signal) and the noise-degraded audio file are loaded. Subsequently, a pre-processing phase is indispensable to run the inference via the NSNet2. The pre-processing consists of applying a STFT on the degraded signal using a window size equal to 512 and an overlap of 50%, as the NSNet2 trained from dataset [23] using the same configuration. Consequently, the process generates a quantized instance of NSNet2 using the contents of the chromosome to define the mixed-precision quantization bit-depths, and the inference occurs on each of the frames generated by the STFT, generating a mask that is multiplied element-wise with the inference input to apply denoising, resulting in a new frame in the frequency domain. Re-combining the generated frames and leveraging the inverse short-time Fourier transform (iSTFT) to return to the time domain leads to the de-noised audio file. Finally, comparing the de-noised audio file with the reference audio file allows for the evaluation of the PESQ metric. The normalization of PESQ occurs between 0.5 and 4.5, which are the minimum and maximum PESQ values, respectively.

IV. Evaluation

The outcomes of the genetic algorithm highlight a clear improvement in performance across generations regarding the optimization of hardware performance (inference time, memory footprint) and audio quality (PESQ score). Figure 1

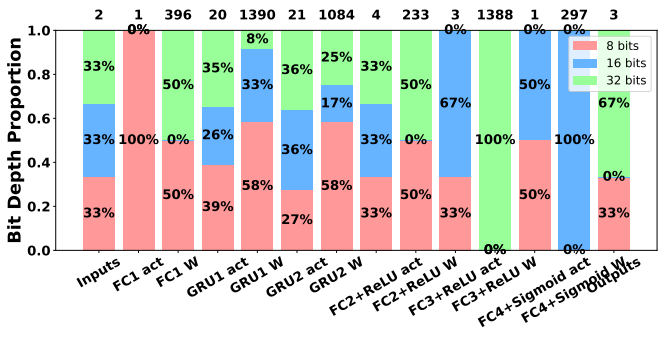

Fig. 3. Bar chart showing the bit depth for each activation (act) and weights (W) of the NSNet2, with the related memory footprint (in KBytes) above each bar.

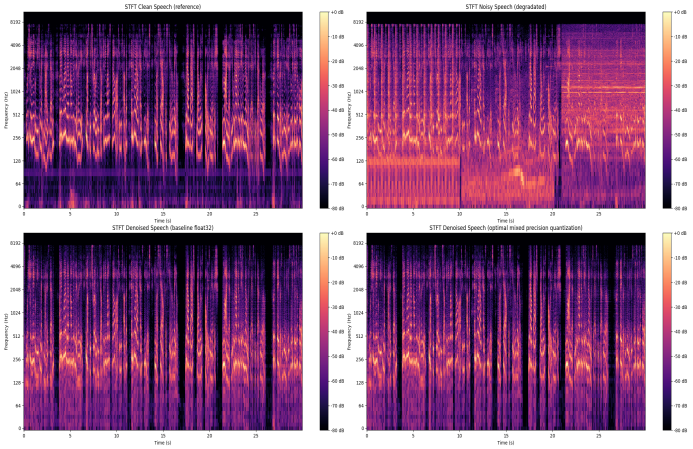

Fig. 4. Top left the spectrogram of the clean speech (reference). At the top right, the spectrogram of the speech is degraded by the noise (degraded). Bottom left, the spectrogram of de-noised speech with the NSNet2 *float32* baseline. The spectrogram of de-noised speech on the bottom right with optimal mixed-precision quantized NSNet2.

illustrates the evolution of the Pareto optimal across selected generations (0, 9, 24, 49), showing the relationship between PESQ and inference time. The color and size of the scatter plot points indicate the generation number and memory footprint, respectively.

We have selected the Pareto optimal solution with the best PESQ (Figure 3 illustrates the bit depth distribution for weights and activations in NSNet2) and reported the metrics in the table I where we compare with the same metrics obtained for the baselines considered, i.e., uniform quantization at 8-bit integers, uniform quantization at 32-bit integers, and full-precision *float32*. As expected, we got the best audio quality for the full-precision baseline but with poor memory footprint and inference time. Conversely, the int-8 baseline is fast and compact but with poor PESQ. Finally, our mixed-precision quantization model presents a PESQ almost equivalent to the *float32* baseline with a reduction of the inference time by 29.93% and the memory footprint by 57.88%.

Metric	int8	int32	float32	MPQ
Inference Time (ms)	5.45	7.87	8.32	5.83
Memory Footprint (MB)	2.81	11.23	11.23	4.73
PESQ	1.6224	2.4560	2.4955	2.4803

TABLE I
METRICS COMPARISON BETWEEN BEST MIXED-PRECISION SOLUTION AND BASELINES.

Figure 2 shows the trends in PESQ, inference time, and memory footprint metrics across generations of the evolutionary algorithm. At the first generation, initial solutions show audio quality (PESQ between 1.6 and 2.4) but high variability, with suboptimal hardware performance (memory footprint: 6.22–6.88 MB; inference time: 6.1–7.2 ms), indicating room for model optimization. By the tenth generation, audio quality stabilizes near 2.4, with reduced inference time (5.8–6.3 ms) and memory footprint (5.28 MB). At the twenty-fifth generation, PESQ reaches 2.46 (close to the *float32* baseline of 2.49), with stable inference time (5.8 ms) and reduced memory (4.59 MB). By the final generation, solutions achieve near-baseline PESQ (2.47), a stable inference time of 5.8 ms, and a further reduced memory footprint of 4.19 MB.

Overall, the optimal mixed-precision quantization configuration via the genetic algorithm considerably reduced the size and inference time of the neural network while maintaining a high denoising quality, with a PESQ score similar to the *float32* baseline. Figure 4 shows the spectrogram of the audio with clean speech, the speech deteriorated by noise with a high SNR equal to 3.0, the de-noised audio with *float32* baseline NSNet2, and the de-noised audio with the mixed-precision quantized model. The mixed-precision quantization configuration used belongs to the Pareto optimal of the latest generation. The spectrograms visually confirm the equivalence of the baseline and quantized model PESQ score. The experiments use audio files containing speech signals, to which background noise is manually added to simulate real-world noisy environments. These features make the new model suitable for environments with limited resources and edge devices.

A. Comparison with ONNX Runtime Quantization

We compared our results with those of ONNX Runtime [24], a framework developed by Microsoft. It supports the ONNX format, ensuring efficient and optimized inference ad hoc for different hardware architectures. It also allows the quantization of neural networks using specific units designed for this purpose, providing an extremely effective calibration system starting from the dataset. However, ONNX Runtime's quantization only supports 8-bit and 16-bit uniform quantization without enabling mixed-precision quantization. Furthermore, 8-bit quantization is implemented in QFormat, while 16-bit quantization is via quantize-dequantize-quantize (QDQ). Therefore, in the case of 16-bit quantization, the operations are still developed in float32. As a result, we report the results obtained via 8-bit uniform quantization: the NSNet2 model quantized via ONNX Runtime provides a PESQ equal to 1.9473, a higher value than that obtained with our framework (1.6224), probably due to a more sophisticated calibration system on the representative dataset compared to the most basic one used by us. However, the quantized ONNX model occupies 11.2 MB in memory (against the 2.81 MB of our C model), probably due to internal metadata. The need for additional memory space may reduce the feasibility of deployment on the edge device.

V. RELATED WORK

In this section, we describe the different mixed-precision quantization frameworks currently representing the state-of-the-art and compare them with our framework. One of the distinctive features of our research is a mixed-precision quantization framework tailored for audio applications, using PESQ as a metric.

One of the most used approaches for estimating the optimal configuration for mixed-precision quantization is Stochastic Layer-Wise precision (DQ) [25], which consists of stochastically testing the sensitivity of the different layers, evaluating which can be quantized more aggressively without introducing a significant error into the network output. Therefore, this method considers the introduced error a metric, while our framework assesses the output quality (i.e., denoising).

Another mixed-precision quantization strategy is HAQ (Hardware-Aware Quantization) [26], which contemplates hardware constraints to optimize accuracy, memory footprint, and energy consumption on different architectures. Similarly, On-Chip Hardware-Aware Quantization (OHQ) directly measures the metrics using the actual edge devices instead of simulating them [27]. This approach is remarkably impactful in deploying the network on embedded systems. Our work measured the inference time by compiling C code, which efficiently optimizes the network on different architectures. In fact, by compiling the C code for diverse architectures and carrying out the benchmark, it is possible to establish the optimal configuration of mixed-precision quantization on distinct hardware.

HAWQ (Hessian-Aware Quantization) [28] is a cornerstone of the state-of-the-art of mixed-precision quantization. This framework evaluates the information from the Hessian, i.e., from the second-order derivatives, to establish the sensitivity of the different layers and consequently establish which ones tolerate quantization more. This approach exploits generic metrics while we explore an effective quantization technique for denoising and speech enhancement applications.

Other approaches are Bayesian Bit [29], based on Bayesian optimization techniques to quantize the network, and EvoQ [30], a method to define the optimal mixed-precision quantization configuration using genetic algorithms. EvoQ presents an approach similar to the one we adopted, using evolutionary algorithms consisting of mutation, cross-over, and evaluation phases. The fundamental difference is in the metrics used: EvoQ presents a generic approach with metrics applicable to different neural networks. Instead, we propose a specific framework for audio applications such as denoising, considering specialized metrics such as PESQ.

Finally, one of the most widespread commercial frameworks for deploying neural networks on embedded systems is TFLite [31]. However, it provides mixed-precision limited to only 8 and 16 bits, with the possibility of performing mixed-precision quantization but limiting the choice to weights and activations, with restrictions in selecting the layers to quantize.

In the audio field, mixed-precision quantization guarantees substantial gain in memory footprint and processing requirements on NsNet2, as reported in [32]. However, the bit-depth was assigned based on apriori knowledge with 8-bit precision applied to the GRU layers to yield maximum memory savings. That study did not leverage any automatic optimization.

Although excellent mixed-precision quantization techniques exist, they often use general-purpose metrics. With our project, we intend to introduce a framework for optimal mixed-precision quantization of a neural network in the audio field.

VI. Conclusion

In conclusion, we provide a framework for estimating the optimal configuration of mixed-precision quantization through genetic algorithms on neural networks in the audio field. Experiments demonstrate that mixed-precision quantization significantly reduces the model's hardware-related metrics, such as memory footprint (57.88%) and inference time (29.93%), maintaining a remarkable de-noising quality. Furthermore, the PESQ score effectively ensures that the optimization algorithm converges to high-quality solutions.

This method leads to efficient neural networks without requiring further fine-tuning and with only post-training quantization.

Acknowledgements

This work has received funding from the European Union's Horizon Research and Innovation program under grant agreement No 101070374.

VII. Data Availability

The code, data, and documentation relating to the experiments are available via the open-source repository: https://github.com/alecerio/nas-nsnet2.

References

[1] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[2] D. Michelsanti, Z.-H. Tan, S.-X. Zhang, Y. Xu, M. Yu, D. Yu, and J. Jensen, "An overview of deep-learning-based audio-visual speech enhancement and separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1368–1396, 2021.

[3] S. Braun, H. Gamper, C. K. Reddy, and I. Tashev, "Towards efficient models for real-time deep noise suppression," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 656–660.

[4] A. Bansal and N. K. Garg, "Environmental sound classification: A descriptive review of the literature," *Intelligent systems with applications*, vol. 16, p. 200115, 2022.

[5] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE access*, vol. 7, pp. 19 143–19 165, 2019.

[6] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A comprehensive survey on tinyml," *IEEE Access*, 2023.

[7] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[8] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[9] M. Rakka, M. E. Fouda, P. Khargonekar, and F. Kurdahi, "A review of state-of-the-art mixed-precision neural network frameworks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[10] R. Miccini, A. Cerioli, C. Laroche, T. Piechowiak, J. Sparsø, and L. Pezzarossa, "Towards a tailored mixed-precision sub-8bit quantization scheme for gated recurrent units using genetic algorithms," *arXiv preprint arXiv:2402.12263*, 2024.

[11] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

[12] Y. Jaafra, J. L. Laurent, A. Deruyver, and M. S. Naceur, "Reinforcement learning for neural architecture search: A review," *Image and Vision Computing*, vol. 89, pp. 57–66, 2019.

[13] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE transactions on neural networks and learning systems*, vol. 34, no. 2, pp. 550–570, 2021.

[14] X. Zhou, A. K. Qin, Y. Sun, and K. C. Tan, "A survey of advances in evolutionary neural architecture search," in *2021 IEEE congress on evolutionary computation (CEC)*. IEEE, 2021, pp. 950–957.

[15] S. Santra, J.-W. Hsieh, and C.-F. Lin, "Gradient descent effects on differential neural architecture search: A survey," *IEEE Access*, vol. 9, pp. 89 602–89 618, 2021.

[16] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in *International conference on machine learning*. PMLR, 2019, pp. 7603–7613.

[17] U. Kulkarni, A. S. Hosamani, A. S. Masur, S. Hegde, G. R. Vernekar, and K. S. Chandana, "A survey on quantization methods for optimization of deep neural networks," in *2022 international conference on automation, computing and renewable systems (ICACRS)*. IEEE, 2022, pp. 827–834.

[18] S. Braun and I. Tashev, "Data augmentation and loss normalization for deep noise suppression," in *International Conference on Speech and Computer*. Springer, 2020, pp. 79–86.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[20] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[21] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[22] BaiCai, "python-pesq: A python package for calculating the pesq," 2019, accessed: 2024-10-24. [Online]. Available: https://github.com/vBaiCai/python-pesq

[23] Microsoft, "Dns challenge datasets," 2020, accessed: 2024-11-06. [Online]. Available: https://github.com/microsoft/DNS-Challenge/tree/interspeech2020/master/datasets

[24] Microsoft, "Onnx runtime," https://github.com/microsoft/onnxruntime, 2024.

[25] X. Huang, Z. Shen, S. Li, Z. Liu, H. Xianghong, J. Wicaksana, E. Xing, and K.-T. Cheng, "Sdq: Stochastic differentiable quantization with mixed precision," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9295–9309.

[26] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, "Hawq-v3: Dyadic neural network quantization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.

[27] W. Huang, H. Qin, Y. Liu, J. Liang, Y. Ding, Y. Li, and X. Liu, "Ohq: On-chip hardware-aware quantization," *arXiv preprint arXiv:2309.01945*, 2023.

[28] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq-v2: Hessian aware trace-weighted quantization of neural networks," *Advances in neural information processing systems*, vol. 33, pp. 18 518–18 529, 2020.

[29] M. Van Baalen, C. Louizos, M. Nagel, R. A. Amjad, Y. Wang, T. Blankevoort, and M. Welling, "Bayesian bits: Unifying quantization and pruning," *Advances in neural information processing systems*, vol. 33, pp. 5741–5752, 2020.

[30] Y. Yuan, C. Chen, X. Hu, and S. Peng, "Evoq: Mixed precision quantization of dnns via sensitivity guided evolutionary search," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[31] T. L. M. Team, "Tensorflow lite for microcontrollers," https://github.com/tensorflow/tflite-micro, 2024, gitHub repository.

[32] M. Rusci, M. Fariselli, M. Croome, F. Paci, and E. Flamand, "Accelerating rnn-based speech enhancement on a multi-core mcu with mixed fp16-int8 post-training quantization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2022, pp. 606–617.