Neural Architecture Search for Optimized TinyML Applications

Abbas Kassem Zein[1], Rand Abou Diab[1], Mohamad Yaacoub[2], Ali Ibrahim[1]

[1] Computer and Communication Engineering Department, Lebanese International University (LIU), Beirut 1105, Lebanon
[2] Department of Naval, Electrical, Electronics, and Telecommunications Engineering, University of Genoa, Genoa, Italy

Abstract. Integrating machine learning methods on circuits with low power consumption and low hardware complexity is challenging at the different levels of the design process. While designing the network at software level, it is crucial not only to achieve a high classification accuracy, but to also keep monitoring the model complexity to reduce as much as possible the hardware burden. This paper proposes a search method merged with the Neural Architecture Search (NAS) to enhance the performance and reduce the complexity of deep learning models. Accuracy and number of Floating-Point Operations Per Second (FLOPS) are employed as evaluation metrics for the targeted models. Achieved results demonstrate that the proposed method outperforms similar state of the art architectures for the same problem, through showing comparable accuracy with up to 70% of reduction in complexity.

1 Introduction

Designing machine learning models for integrated circuits is a complex task due to the need to balance accuracy and complexity while designing the circuits. The process includes dealing with the dataset, choosing the appropriate model architecture, and setting accuracy requirements taking into consideration the hardware constraints. This latter is the main design issue when low power battery autonomous or even battery less devices are targeted. Design complexity derives from different aspects, such as hardware constraints, optimization techniques and the challenge of maintaining the quality of the results while reducing the circuit complexity. The design of wearable devices faces significant challenges with respect to power consumption and memory footprint constrained by the lightweight and small battery size requirements. This necessitates using optimized neural network models to ensure well functionality within these complicated environments. Addressing such a problem while maintaining the model accuracy adds more burden on the design process. In order to address such limitations, optimization techniques such as quantization and model compression are usually employed [1].

Designing efficient Deep Neural Networks (DNNs) manually requires huge computational resources. This complexity comes from the significant trial-and-error iterations required to fine-tune and determine the best network architecture. The process involves exploring various architectural parameters, layer setups, and hyperparameters,

making it a resource-heavy task, in addition to the extensive experimentation to achieve satisfactory performance. The solution to the aforementioned problems could be by adopting an automated process of model architecture design. Neural Architecture Search (NAS) is a cutting-edge technique which aims to automate the designing process of neural network architectures, but usually these searches aim to only find models with good accuracy without taking into consideration model complexity. NAS systematically explores a space of potential architectures through making use of reinforcement learning, evolutionary algorithms, or other search strategies.

2 Related Work

Many works have been presented recently focusing on the NAS architectures and implementation. In [2], authors introduced novel advancements in neural architecture search and edge device deployment. Dynamic Channel Scaling is proposed to adjust channel numbers during training, showing their impact on accuracy and efficiency. Also, they used progressive spacy shrinking to address the large search space through efficiently pruning and shrinking it to reduce the search burden. The framework in [3] involves two stages: the first aims to search for architectures satisfying latency constraints at the macro level, this was denoted as LightNets. The second proposes a cost-effective evolutionary scheme for micro-level channel exploration. Notable contributions include a latency predictor for accurate on-device latency estimation, integrated into LightNAS to eliminate the need for extensive on-device measurements, and a lightweight differentiable search method that reduces optimization complexity. The paper in [4] tackles the limitations of existing techniques by introducing a generic search space pruning algorithm, a fast differentiable search for mixed precision, a joint search for architecture and precision, and a combined search for accelerator, architecture, and mixed precision. Authors applied these methods on MobileNetV2 model, which showed a significant improvement in efficiency in both latency and accuracy when compared to manually designed and existing approaches. In [5], authors introduce a hybrid network design that significantly reduces multiplication operations, which caused an improvement in computational efficiency and maintained high accuracy. Jiang et al. [6] explore the synergistic approach of simultaneously optimizing neural network architectures and their corresponding hardware accelerators. The work provided insights into co-search methodologies, highlighting the importance of joint optimization to achieve both high performance and efficient hardware implementation. The research done in [7] delves into the imperative need for error-resilient applications within the realm of Machine Learning (ML). It introduces Approximate Computing Techniques (ACTs) as a viable solution, emphasizing their role in facilitating efficient, real-time, and low-power implementations of ML algorithms. The paper in [8] proposes a novel NAS methodology, called S3NAS, which integrates the latest research results into a three-step process: supernet design, Single-Path NAS, and scaling/post-processing. The methodology includes designing a supernet structure to optimize network accuracy on specific hardware, performing a rapid NAS with modified loss functions, and scaling the network until the desired latency is achieved. Experimental results show that S3NAS offers better accuracy-latency tradeoffs on various hardware platforms compared to existing state-of-the-art models. Authors in [9] introduces

SurgeNAS, a methodology aimed at designing efficient Convolutional Neural Networks (CNNs) by challenging three common assumptions in NAS: the effectiveness of theoretical complexity metrics, the transferability of architectures searched on proxy tasks, and the impact of search space on performance. Results demonstrate that SurgeNAS achieves superior performance and efficiency on various hardware platforms by addressing these assumptions.
In this research, NAS was applied on Touch Modality Classification problem. This field had been a place on interest to many researchers lately. The research in [10] presented the challenges posed by the computational demands of Convolutional Neural Networks (CNNs) for resource-limited devices, acknowledging their potential but also recognizing the hurdles they present. In response, the authors proposed a Mixed-Precision Architecture that combines 32-bit floating-point representation for input and output layers with binarization in hidden layers. This strategic design choice aims to balance classification accuracy and computational efficiency. Innovative hardware architecture was introduced in [11] for Tensorial Support Vector Machines (TSVM) based on Shallow Neural Networks (NN) designed for Single Value Decomposition (SVD) computation, specifically tailored for touch modality classification. In comparing performance, the proposed NN demonstrates similar Mean Squared Error and Cosine Similarity to the widely used one-sided Jacobi algorithm, with FPGA implementation showcasing a remarkable 324× speedup, along with reductions in hardware resources by 58% and power consumption by 67%.

3 Methodology

The NNI [12] library has been adopted in this work with the main objective is to find an optimized neural architecture as a tradeoff between accuracy and complexity. The process includes searching through parameters like number of layers, neurons, learning rate, number of epochs, etc. referred to as hyper-parameters. The search is guided by a searching algorithm which is Tree-structured Parzen Estimator (TPE) [13] which uses the Bayesian optimization technique at which it maintains probability distributions for promising and less promising parameter configurations, to sample new configurations accordingly.
As illustrated in Fig. 1, the system is composed of two main components: 1) the controller and 2) the trial script. Concerning the controller, it is used to define the search space, searching algorithm and number of trials. Moreover, it is responsible for sampling parameters from the search space using the searching algorithm and sending them to the trials script. The search space defines the variables for the trials existing of: model types, hyper-parameters e.g., filter size, kernel size, pool size for convolutional

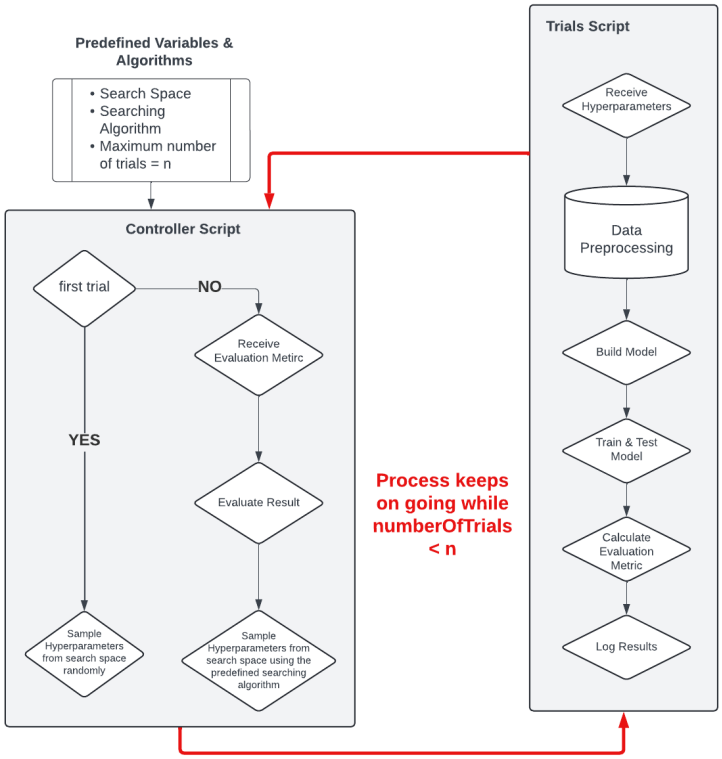

Fig. 1: High level flow chart of the proposed system

models, and number of neurons for MLP models. Besides, it manages the general parameters such as batch size, number of epochs, and learning rate. On the other hand, the trial script is called by the controller in every experiment: the controller passes to the script the sampled parameters to be used in different stages. The first step consists of triggering a function to preprocess data and prepare it for training, then it uses the "Model Type" from the sent parameters to choose which model to build. After that, the received hyper-parameters are used to set the model architecture and configuration. When the training is done, the accuracy-based evaluation is sent to the controller using a callback function to tune new parameters from the search space.

4 Implementation

In order to find the most efficient system for the targeted problem, the search has been applied following three different methods: 1) Static Search, 2) Dynamic Search, and 3) updated Dynamic Search. Starting with the static search, the number of layers has been fixed to a minimum of two trying to get an optimum model while tuning the other hyperparameters. Hence, the number of trials has been reduced from 1000

(with 17 hours search) to 100 trial (with 2 hours search). On the other hand, dynamic search is a shrank version of the static one which consists of modifying the number of layers to be variable according to the search space. This strategy provides the ability to tune the number of layers choosing a model among different architectures during the search process.

In both static and dynamic searches, the evaluation metric was only the model testing accuracy. The updated dynamic search introduces an additional evaluation metric as a trade-off between complexity and accuracy.

$$Eval = acc \times Wacc + (1\text{-}nFlops) \times Wcomp \quad (1)$$

Where Eval is the evaluation metric, acc is the testing accuracy, Wacc is the accuracy weight chosen to be 85%, nFlops is the normalized number of flops to be between 0 and 1, and Wcomp is the weight of the complexity chosen to be 15%.

5 Experimental Results

The proposed method has been applied for the touch modality classification problem presented in [14]. The dataset includes 26 alphabetical patterns at which five subjects were involved to collect 50 samples of each letter. Extensive trials have been done resulting in different results highlighting the evaluation metric mentioned above complexity. The results of the three different phases Static, Dynamic and updated Dynamic along with a comparison with the state of the art are presented in Table 1, Table 2, and Table 3.

Table 1. Static Search Top Three Models

Model type	Accuracy	KFlops	Epochs	Learning Rate	Conv Layer	Number of Neurons in Dense layers
MLP	.8846	389.9	65	.000776	-	{19,19}
Conv	.8769	266.9	68	.000284	{32,32} {10,8}	-
MLP	.873	246.3	63	.000610	-	{12,16}

Table 1 highlights the results obtained in the static search indicating their accuracy, complexity, and architectural details. The name static stems from the fact that the number of layers in each model remains constant. Moreover, the framework was fixed by limiting the number of variables in each model, such as having two layers for MLP and two layers for CONV 1D. This approach makes it simpler to measure and compare the model performance. By limiting the search space, it was possible to monitor how various parameters and configurations, such as kernel size, filter size, batch size, epochs, learning rate, and number of neurons, affect the model output. This can be referred to as hyper-parameters search rather than architectural search since the search is conducted on both the entire model (MLP & CONV 1D) and the neuron level, along with the other mentioned parameters.

Table 2. Results of Five different dynamic searches

Model type	Accuracy	KFlops	Epochs	Learning Rate	Conv Layer	Dense Layers
MLP	.9153	780.6	95	.000577	-	{38,37}
MLP	.9076	800.8	94	.000512	-	{39,32}
MLP	.9038	678.1	116	.000531	-	{33}
MLP	.9269	884.8	178	.000496	-	{43,38,40}
MLP	.92307	534.6	124	.00028	-	{26,42}
MLP	.92307	577.4	122	.000209	-	{28,41,43}
Conv1D	**.9538**	**696.3**	**161**	**.00609**	**{4}{2}**	**-**
Conv1D	.9423	860.1	174	.006833	{4}{4}	-
Conv1D	.9346	860.1	195	.00451	{4}{4}	-
Conv1D	**.9423**	**696.3**	**181**	**.008651**	**{4}{2}**	**-**
Conv1D	.9384	696.3	198	.00802	{4}{2}	-
Conv1D	.9346	348.1	166	.004877	{2}{2}	-
MLP	**.9307**	**226.1**	**122**	**.000286**	**-**	**{11}**
MLP	.9076	554.1	180	.000659	-	{27,22,14}
Conv1D	.8961	3030.4	102	.00049	{8,4} {10,8}	-

The dynamic search results are reported in Table 2. It performs the search on three different levels: model type, number of neurons, and number of layers. The top three results of five different searches are reported. On the other hand, a comparison with [14] is done since the same dataset and networks were adopted. Moreover, the same splitting ratio of the dataset at an 80-20 train-test has been adopted for fair comparison. The top result achieved around 95.38% accuracy with 696.3 KFlops. Compared to CNN13 in [14], having an accuracy of 95.5% and 2730.5 KFlops, the proposed system produced a model having the same accuracy with a 74.5% of complexity reduction. Another significant reduction in complexity occurred with the MLP model trained by the system, achieving 93.07% accuracy with a number of FLOPS equal to 226.1 KFlops. Comparing this to CNN3 in [14], which achieved 93.8% accuracy with a number of FLOPS equal to 2935 KFlops, the system produced a model with a 92.28% reduction in FLOPS while reducing accuracy by less than 1%.

Table 3. Top Three Results from Updated Dynamic Search

Model type	Evaluation Metric	Accuracy	KFlops	Number Epochs	Learning Rate	Conv Layer
Conv1D	**.936**	**.9723**	**436.872**	**199**	**.0053**	**{2}{2}**
Conv1D	.934	.9692	430.08	199	.0033	{2}{2} {2}{2}
Conv1D	. 933	.9692	436.87	178	.0089	{2}{2}

The updated dynamic search results are reported in Table 3. The evaluation metric score explained by Equation (1) is highlighted in the table. It is shown that the best achieved result is with a score of 0.936/1 having an accuracy of 97.23% with a model complexity equals to 436.872 KFlops. When compared to CNN7 in [14] corresponding to the model with the highest accuracy of 97.7% and having a complexity of 2704.2 KFlops, a high reduction in model complexity from 2704.2 KFlops to 436.872 KFlops i.e. 83.84% with a slight loss in accuracy in the range of 0.5% has been achieved.

6 Conclusion

The aim of this work is to explore the potential of using NAS in optimizing deep learning models for resource constrained devices focusing on a touch modality classification dataset. The objective behind is to select the optimized model targeting integrated circuit design. Achieved results demonstrated the effectiveness of the proposed approach in reducing the complexity of the neural network model by 83.84% in terms of floating-point operations. Comparative accuracy is achieved showing a slight loss of only 0.5% from 97.7% to 97.2% which is considered negligeable when compared to the major advantage gained with the complexity reduction. This work pave the way towards low power integrated circuit design of optimized neural network models.

Acknowledgments

This project has received funding from the European Union's Horizon Europe research and innovation program under the Marie Sklodowska-Curie grant agreement No 101086359.

References

1. Sakr F, Berta R, Doyle J, Capello A, Dabbous A, Lazzaroni L, Bellotti F. CBin-NN: An Inference Engine for Binarized Neural Networks. Electronics. 2024; 13(9):1624.

2. X. Luo, D. Liu, S. Huai, H. Kong, H. Chen and W. Liu, "Designing Efficient DNNs via Hardware-Aware Neural Architecture Search and Beyond," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 6, pp. 1799-1812, June 2022.
3. X. Luo, D. Liu, H. Kong, S. Huai, H. Chen and W. Liu, "LightNAS: On Lightweight and Scalable Neural Architecture Search for Embedded Platforms," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 6, pp. 1784-1797, June 2023.
4. K. T. Chitty-Venkata, Y. Bian, M. Emani, V. Vishwanath and A. K. Somani, "Differentiable Neural Architecture, Mixed Precision and Accelerator Co-Search," in IEEE Access, vol. 11, pp. 106670-106687, 2023.
5. Shi, H., You, H., Wang, Z., & Lin, Y. (2023). NASA+: Neural Architecture Search and Acceleration for Multiplication-Reduced Hybrid Networks. IEEE Transactions on Circuits and Systems. I, Regular Papers, 70(6), 2523–2536.
6. Sekanina, L. (2021). Neural Architecture Search and Hardware Accelerator Co-Search: A Survey. IEEE Access, 9, 151337–151362.
7. H. Younes, A. Ibrahim, M. Rizk and M. Valle, "Algorithmic Level Approximate Computing for Machine Learning Classifiers," 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 2019, pp. 113-114.
8. Lee, J., Rhim, J., Kang, D., & Ha, S. (2022). SNAS: Fast Hardware-Aware Neural Architecture Search Methodology. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 41(11), 4826–4836.
9. Luo, X., Liu, D., Kong, H., Huai, S., Chen, H., & Liu, W. (2023). SurgeNAS: A Comprehensive Surgery on Hardware-Aware Differentiable Neural Architecture Search. I.E.E.E. Transactions on Computers/IEEE Transactions on Computers, 72(4), 1081–1094.
10. H. Younes, A. Ibrahim, M. Rizk and M. Valle, "A Mixed-Precision Binary Neural Network Architecture for Touch Modality Classification," SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME, online, 2021, pp. 1-4.
11. H. Younes, A. Ibrahim, M. Rizk and M. Valle, "A Shallow Neural Network for Real-Time Embedded Machine Learning for Tensorial Tactile Data Processing," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 10, pp. 4232-4244, Oct. 2021.
12. Microsoft NNI Team, "Microsoft NNI: An open-source AutoML toolkit for neural architecture search, model compression and hyper-parameter tuning," GitHub, https://github.com/microsoft/nni, Accessed May 20, 2024.
13. Ozaki, Y., Tanigaki, Y., Watanabe, S., & Onishi, M. (2020). Multiobjective tree-structured parzen estimator for computationally expensive optimization problems.
14. Al Haj Ali, H., Gianoglio, C., Ibrahim, A., Valle, M. (2023). Resource-Constrained Implementation of Deep Learning Algorithms for Dynamic Touch Modality Classification. Lecture Notes in Networks and Systems, vol 546. Springer, Cham.