End-to-end evolutionary neural architecture search for microcontroller units

René Groh, Andreas M Kist
Department Artificial Intelligence in Biomedical Engineering, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU),
Werner-von-Siemens Street 61, 91052 Erlangen, Bavaria, Germany

Abstract

Smart wearable devices require accurate, fast, and energy-efficient neural networks to allow for optimal application performance. To advance the field of neural architecture search (NAS), we introduce our end-to-end evolutionary NAS (EvoNAS) for microcontroller units that optimize both, pre-processing and neural network architectures. Each neural network architecture is assessed using the multi-objective accuracy, memory footprint, inference time, and energy consumption, to derive a common performance measure to be maximized. To ensure immediate use of all potential solutions on the microcontroller environment, we create a software-hardware chain in which each neural network is deployed to measure the inference time and power consumption directly. In a proof-of-concept study, we focused on the analysis of audio-based speech commands. Our experiments suggest that 2D convolutional layers with automatically set pre-processing (short-time Fourier transforms) outperform 1D convolutional layers with raw audio signals. We show that our end-to-end EvoNAS scales with the complexity of the classification task and is still able to find constraint-preserving, and thus deployable, Pareto-optimal neural network architectures even when the classification task is more complex. Our proposed EvoNAS approach is dataset and hardware-agnostic, allowing a universal use across a wide range of applications.

Introduction

In this work, we propose an end-to-end software-hardware chain based on an evolutionary neural architecture search (EvoNAS) to optimize 1D and 2D preprocessing and neural network architectures in a single step. Found combinations are contained in a single model file and directly tested on the target hardware allowing for immediate use in a production environment. In our proof-of-concept study we utilized the Speech Commands [1] dataset (see Figure 1).

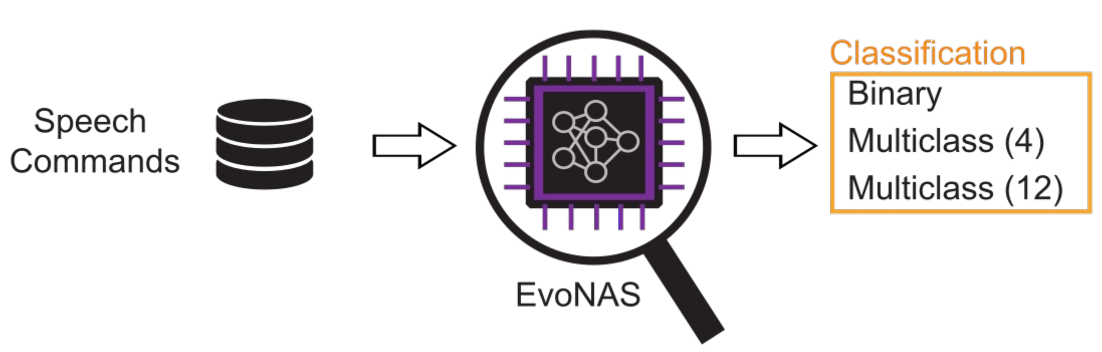

Figure 1: EvoNAS for several classification tasks

Main contributions:

- We propose a fully functional, scalable, software-hardware end-to-end pipeline for evolutionary NAS.
- We included 2D audio preprocessing (short-term Fourier transform) as part of the evolutionary optimization. This enables direct deployment and allows for end-to-end design.
- We found that 2D convolutional neural networks outperform 1D networks in terms of overall fitness achieved.
- We show that our optimization algorithm is able to find Pareto-optimal architectures.

Methods

Hardware/software setup:

- nRF52840 (32-bit ARM Cortex-M4) Development Kit Board from Nordic Semiconductor
- Power Profiler Kit II (PPK2) from Nordic Semiconductor
- Zephyr RTOS
- TensorFlow and TensorFlow Lite Micro
- Kapre Python library [2]

To find neural network architectures we utilized evolutionary optimization as shown in Figure 2.

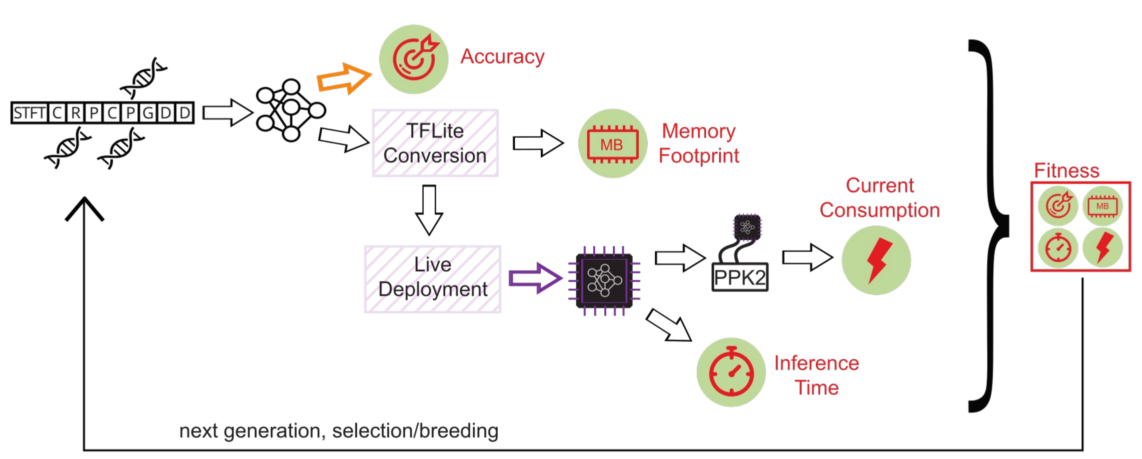

Figure 2: Overview of EvoNAS. Throughout the chain, each of the four objective measures is measured to obtain a fitness score for each candidate neural network architecture. The best individuals are selected and bred to create the population for the next generation.

Each of the sampled preprocessing and neural network architecture combinations (see Figure 3A) is evaluated against a common multi-objective function to determine its fitness. We combine four constrained target measures as a weighted sum into a fitness function that penalizes models that violate constraints and favors models that are better than the given constraint. The fitness function F is described by the following equation:

$$F = \alpha \cdot A + \beta \cdot \frac{M}{M_{\max}} + \gamma \cdot \frac{T}{T_{\max}} + \delta \cdot \frac{W}{W_{\max}},$$

where A is the accuracy of the validation dataset, M the obtained memory footprint in Bytes (B), T the measured inference time (see Figure 3B) in Milliseconds (ms), and W the energy consumption in Millijoule (mJ).

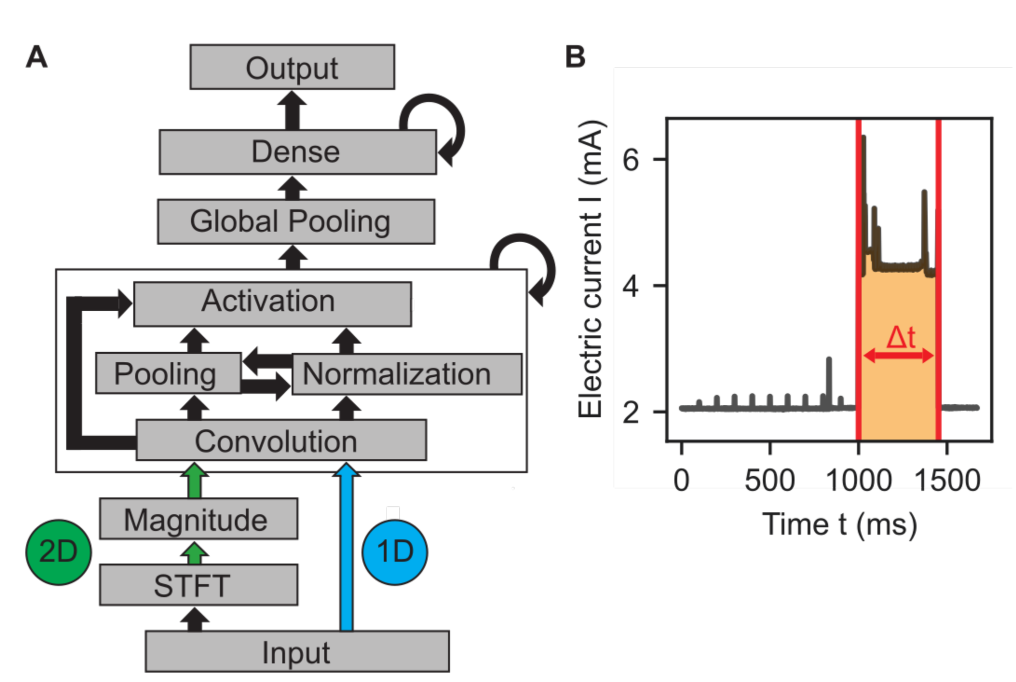

Figure 3: The evolutionary algorithm creates, evaluates, and selects neural network architectures defined by a gene pool and a rule set by also incorporating physical hardware measurements. A) The used rule set. B) Example of a power measurement during inference. The yellow shaded area under the curve between the start and end of inference (Δt = 452ms) is the consumed energy during that time (W = 6.4mJ, with a supply voltage of 3.3V).

Results

Computations in 2D space outperform 1D neural networks

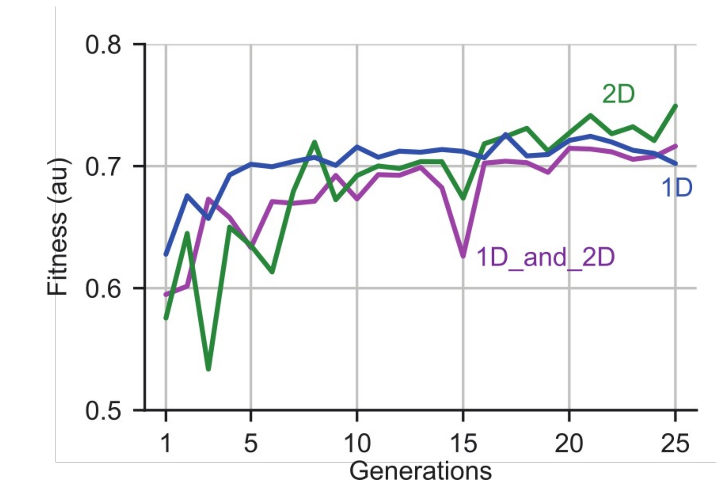

Figure 4: Maximum fitness of the best model in each generation.

Regardless of the gene pool, our algorithm is able to converge to a maximum fitness value with our given hyperparameter set (see Figure 4). Using 1D instead of 2D convolutional layers leads to lower fitness values. All of the best optimized models satisfy previously defined constraints regarding inference time, memory footprint and energy consumption (M_{max} = 0.8 MB, T_{max} = 200 ms, W_{max} = 2 mJ).

EvoNAS yields Pareto-optimal neural architectures

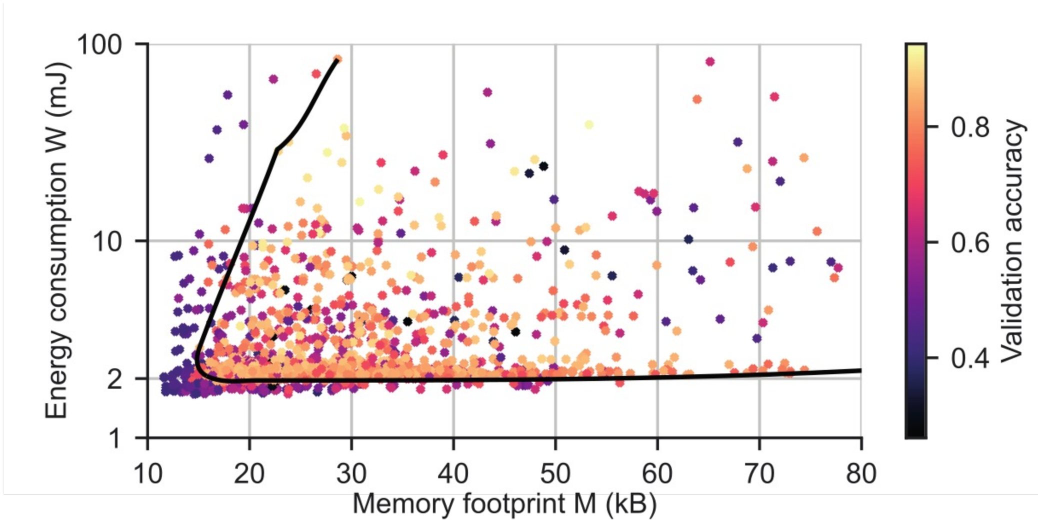

Figure 5: All evaluated neural architectures of the EvoNAS with their respective objective measures (2D space).

Our end-to-end EvoNAS is able to find Pareto-optimal neural architectures (see Figure 5). If a given neural architecture has a memory footprint of less than 15kB and an energy consumption per inference of less than 1.8mJ, the accuracy drops drastically. Increasing both objective measures leads vice versa to higher classification accuracies. Restricting ourselves to neural architectures that performed best on a given objective measurement (Figure 6), we observe architectures that perform very well in terms of energy consumption, while others have high performance in terms of accuracy. However, our algorithm finds a variety of Pareto-optimal neural networks driven by the pre-defined fitness function F.

Considering the model with the best Fitness, assuming we use a 200mAh battery and execute our EvoNAS architecture twice per second (i.e. at a 2 Hz rate), we can run our deployed network continuously for 89.8 h without the need for a recharge (on average I_{sleep} = 2mA, $I_{inference}$ = 4.5mA).

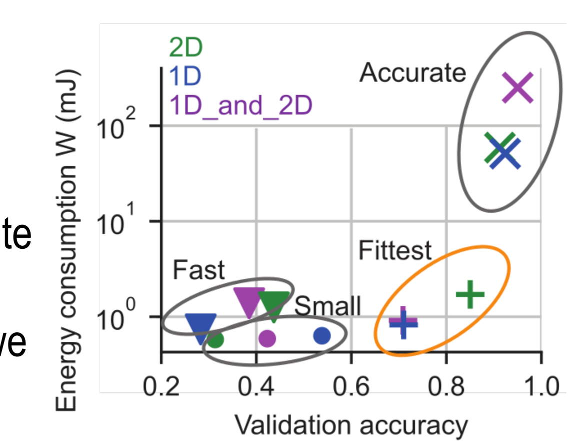

Figure 6 Generated neural architectures that achieve the best values in their respective objective measure (circle: memory footprint, triangle: inference time, cross: validation accuracy, plus: fitness).

Our EvoNAS algorithm adapts for classification problems with varying complexity

Classification task	Model	Validation Accuracy	Memory Footprint (kB)	Inference Time (ms)	Energy consumption per inference (mJ)	Fitness	Test Accuracy	Baseline Test Accuracy (ResNet-50)
Binary	Fittest	0.93	25.4	106	1.55	0.82	0.921	0.969
	Most accurate	0.97	86.1	9439	146.17	-11.06	0.975	
Multiclass (4)	Fittest	0.85	22.4	115	1.71	0.749	0.846	0.966
	Most accurate	0.91	68.9	4096	59.03	-4.07	0.904	
Multiclass (12)	Fittest	0.73	27.4	91	1.33	0.7	0.476	0.914
	Most accurate	0.85	46.0	1087	15.52	-0.43	0.714	
	DS-CNN [3]	-	38.6	-	-	-	0.944	
	res15 [4]	-	237	-	-	-	0.984	

Table 1: Results EvoNAS for different classification problems (2D space).

Our EvoNAS approach yields appropriate neural architectures for each classification problem and complexity. The performance in terms of accuracy is slightly worse compared to our ResNet-50 baseline results where we used STFT preprocessing as well. However, we are an order of magnitude smaller and faster and can ensure that our proposed network architectures are actually deployable to our target environment because they are fulfilling every given constraint. Our found architectures for all three classification problems have an up to 99, 992% lower memory footprint compared to our ResNet-50 baseline (280 MB).

Discussion

We are able to find neural networks that don't violate any of the given constraints and that are directly deployable on the hardware they were optimized for. However, especially the test accuracy of the 12-classes Speech Commands classification task is highly affected compared to other state-of-the-art networks. This could be due to the downsampling step we performed, as the SRAM peak memory consumption of the neural network would be too high when used in the microcontroller. Additionally, the used implementation of the TFLM-compatible STFT layer in the kapre python library leads to too large memory footprints if parameters are outside of a very confined range, e.g. *n_fft*.

In future work, we also plan to include Mel spectrogram/MFCC preprocessing instead of only STFT in our gene pool. We will also increase the search space of EvoNAS by expanding the gene pool to find more complex, elaborate neural network architectures.

References

1. P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

2. K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," in Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning. ICML, 2017.

3. Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," arXiv preprint arXiv:1711.07128, 2017.

4. R. Vygon and N. Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss," in Speech and Computer: 23rd International Conference, SPECOM 2021, St. Petersburg, Russia, September 27–30, 2021, Proceedings 23. Springer, 2021, pp. 773–785.

Acknowledgements

We receive funding from the Bavarian State Ministry of Science and the Arts (StMWK) and Fonds de recherche du Québec (FRQ) under the Collaborative Bilateral Research Program Bavaria – Québec managed by WKS at Bavarian Research Alliance (BayFOR) and Fonds de recherche du Québec – Santé (FRQS). The presented content is solely the responsibility of the authors and does not necessarily represent the official views of the above funding agencies.