Surrogate Neural Architecture Codesign Package (SNAC-Pack)

Jason Weitz
University of California San Diego
La Jolla, CA 92093, USA
jdweitz@ucsd.edu

Dmitri Demler
University of California San Diego
La Jolla, CA 92093, USA
ddemler@ucsd.edu

Benjamin Hawks
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA
bhawks@fnal.gov

Nhan Tran
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA
ntran@fnal.gov

Javier Duarte
University of California San Diego
La Jolla, CA 92093, USA
jduarte@ucsd.edu

Abstract

Neural Architecture Search is a powerful approach for automating model design, but existing methods struggle to accurately optimize for real hardware performance, often relying on proxy metrics such as bit operations. We present Surrogate Neural Architecture Codesign Package (SNAC-Pack), an integrated framework that automates the discovery and optimization of neural networks focusing on FPGA deployment. SNAC-Pack combines Neural Architecture Codesign's multi-stage search capabilities with the Resource Utilization and Latency Estimator, enabling multi-objective optimization across accuracy, FPGA resource utilization, and latency without requiring time-intensive synthesis for each candidate model. We demonstrate SNAC-Pack on a high energy physics jet classification task, achieving 63.84% accuracy with resource estimation. When synthesized on a Xilinx Virtex UltraScale+ VU13P FPGA, the SNAC-Pack model matches baseline accuracy while maintaining comparable resource utilization to models optimized using traditional BOPs metrics. This work demonstrates the potential of hardware-aware neural architecture search for resource-constrained deployments and provides an open-source framework for automating the design of efficient FPGA-accelerated models.

1 Introduction

Machine learning models are applied across diverse domains, achieving state-of-the-art performance. Yet, deploying these models in resource-constrained environments, such as edge devices, is challenging due to strict hardware and latency requirements. In order to meet these constraints, researchers have developed methods such as automated architecture design and model compression. Neural Architecture Search (NAS) can automate model design, but existing approaches typically optimize for accuracy alone or use simplistic computational metrics like bit operations (BOPs) that approximate actual hardware performance.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

To address this challenge, we introduce Surrogate Neural Architecture Codesign Package (SNAC-Pack), an integrated software framework that automates the discovery and optimization of neural network architectures specifically tailored for hardware deployment. SNAC-Pack combines two tools: Neural Architecture Codesign [1], which performs a multi-stage neural architecture search to discover optimal models, and the Resource Utilization and Latency Estimator [2], which uses a surrogate model to predict how an architecture will perform when synthesized for implementation on an FPGA.

By integrating a fast and accurate surrogate model for resource estimation directly into the search stage, SNAC-Pack can perform a more effective multi-objective optimization for accuracy, hardware resource usage, and latency while avoiding time-intensive full hardware synthesis for every candidate model. This work explores the use of SNAC-Pack on a jet classification task [3], from a search space to a synthesized model. The SNAC-Pack software is available here.

2 Related Work

Neural Architecture Search (NAS) [4, 5] is an automated process that explores a search space of model architectures with a search strategy based on a set of evaluation objectives, such as accuracy. Neural Architecture Codesign (NAC) [1] is a multi-stage NAS, consisting of a global and local search. The global search explores a wide range of architectures, resulting in a Pareto front of well-performing models. Selecting an architecture along this front, local search performs model compression, including quantization-aware-training (QAT) [6] and pruning [7]. With this further refinement, the model is then synthesized with hls4ml [8, 9].

Separately, surrogate models [10, 11] have been developed to accelerate the design cycle by providing rapid feedback on hardware performance without time-consuming synthesis runs. The rule4ml library [2], for instance, introduces a method to accurately predict the resource utilization, including block ram (BRAM), digital signal processors (DSPs), flip flops (FFs), lookup tables (LUTs), initiation interval (II), and latency of a neural network on an FPGA.

3 Method

The SNAC-Pack tool builds upon NAC by introducing the additional objectives that can be estimated with rule4ml. With NAC integration, the global search stage begins with a user-defined search space that specifies the range of possible architectures, such as the types of layers, number of neurons, activations, and other hyperparameters. A multi-objective search algorithm then explores this space, samples candidate architectures, and performs evaluation. Any combination of the metrics estimated by rule4ml, BOPs, and accuracy can be used as objectives in the search. In this work, SNAC-Pack utilizes an average of the resource utilization estimation, average clock cycles, and accuracy. With a Pareto front produced by global search, an optimal architecture is selected based on the user's specific accuracy and resource constraints. This model then proceeds to the local search stage, which focuses on further refinement with QAT and iterative magnitude pruning. Another Pareto front is produced from which a model with a specific bit precision and sparsity. Finally, the fully optimized model is synthesized using the hls4ml library to generate high-level synthesis (HLS) code for FPGA deployment.

4 Jet Classification Implementation

To show the effectiveness of SNAC-Pack, we apply it to jet classification, a common and challenging task in high energy physics at the Large Hadron Collider (LHC). The goal is to accurately classify collision-created jets into one of five categories (light quark, gluon, W boson, Z boson, top quark) based on their kinematic properties. This is showcased with the hls4ml LHC dataset [3]. The 8 constituents with the greatest transverse momentum are used per jet.

For this task, we configure SNAC-Pack to search for an optimal multi-layer perceptron (MLP) architecture, with an 8 constituent MLP as a comparative baseline [12] with the data processed and normalized as done there. This baseline is chosen, as it is one of the state-of-the-art architectures for this task. The global search uses the Non-dominated Sorting Generic Algorithm II (NSGA-II) [13], exploring a search space of varying number of layers, hidden units per layer, activations, and

Table 1: Comprehensive parameter space for the multilayer perceptron (MLP) search.

Parameter	**Space**
Number of layers	{4, 5, 6, 7, 8}
Hidden units per layer	
Layer 1	{64, 120, 128}
Layer 2	{32, 60, 64}
Layer 3	{16, 32}
Layer 4	{32, 64}
Layer 5	{32, 64}
Layer 6	{32, 64}
Layer 7	{16, 32}
Layer 8	{32, 44, 64}
Activation function	{ReLU, Tanh, Sigmoid}
Batch normalization	{True, False}
Learning rate	{0.0010, 0.0015, 0.0020}
L1 regularization	$\{0.0, 10^{-6}, 10^{-5}, 10^{-4}\}$
Dropout rate	{0.0, 0.05, 0.1}

Table 2: Comparison of model accuracy, BOPs, and estimated hardware metrics from global search. Note that while all metrics are reported here for consistency, the Baseline was optimized for accuracy, NAC for accuracy and BOPs, and SNAC-Pack for accuracy, estimated average resources and clock cycles. The best values are reported in bold.

Model	Accuracy [%]	BOPs	Est. average resources	Est. clock cycles
Baseline [12]	63.77	25,916	7.10	183.74
Optimal NAC [1]	63.81	**7,904**	3.60	**62.69**
Optimal SNAC-Pack	**63.84**	8,352	**3.12**	72.24

batch normalization [14], as seen in Table 1. The objectives used are estimated average hardware utilization, estimated clock cycles, and accuracy. All training is performed with a batch size of 128. Global search is performed for 500 trials, 5 epochs per trial, and an evolutionary population size of 20.

The result is depicted as three Pareto fronts, seen in Figs. 1, 2, 3. For comparison the search is also ran with NAC, optimizing solely for BOPs and accuracy, with the same number of trials and epochs. The resulting Pareto front is shown in Fig. 4. The Pareto-optimal architectures with an accuracy greater than 0.638 are selected for local search and compared to the baseline [12], seen in Table 2. This accuracy value is selected as the threshold to ensure the accuracy meets or exceeds that of the baseline.

For local search, each selected architecture has a 5 epoch warm-up, followed by 10 iterations of iterative magnitude pruning [15, 16], each 10 epochs, with 20% pruned per iteration, with QAT at 8-bit precision.

With hls4ml, the resulting architectures are then synthesized on the Xilinx Virtex UltraScale+ VU13P FPGA, with io_parallel io_type, latency strategy, a reuse factor of 1. Resource utilization and latency is shown in Table 3.

Seen in Table 2, the model found with the SNAC-Pack framework performs similarly to the NAC model and baseline in terms of accuracy, and it is comparable to the NAC model in the other criteria. Table 3 shows that the SNAC-Pack model matches metrics of the other models, but can improve in terms of latency. This is an indicator of a need to improve the estimation of resources themselves.

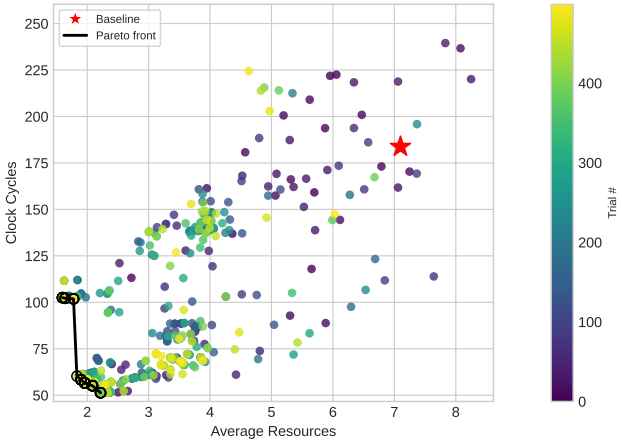

Figure 1: SNAC-Pack Pareto front of estimated average resources versus estimated clock cycles. Each point represents a unique architecture sampled.

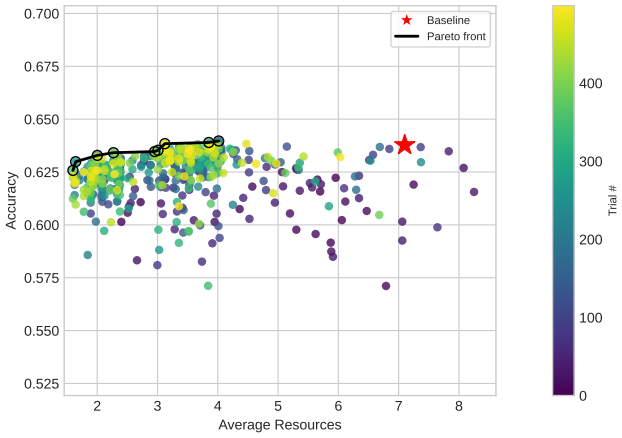

Figure 2: SNAC-Pack Pareto fronts of estimated average resources versus accuracy. Each point represents a unique architecture sampled.

5 Conclusion

This work introduced the Surrogate Neural Architecture Codesign Package (SNAC-Pack), a framework that extends NAC by incorporating resource-aware objectives estimated with rule4ml. Applied to the jet classification task, the optimal model produced by SNAC-Pack performed comparably to the NAC method and baseline reference.

With slight under performance, this new framework establishes the potential of hardware-awareness in NAS. As a functioning pipeline, there is an opportunity for extensions. Incorporating additional surrogate models trained on large datasets [17] in future work can be an enhancement to improve SNAC-Pack, realtive to the BOPs proxy. With this, the estimation of resources can be refined in order to discover lower latency architectures that utilize fewer true resources.

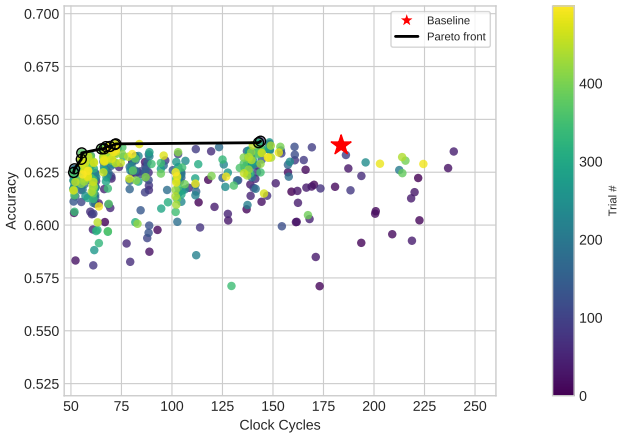

Figure 3: SNAC-Pack Pareto fronts of estimated clock cycles versus accuracy. Each point represents a unique architecture sampled.

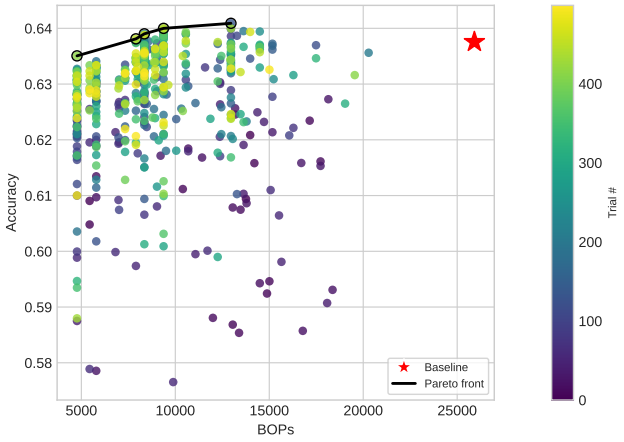

Figure 4: NAC Pareto front of BOPs versus accuracy. Each point represents a unique architecture sampled.

Table 3: Hardware resource utilization and latency estimates for the selected models. The baseline is pruned by 50% and quantized to 8 bits. NAC and SNAC-Pack models are both synthesized after their respective local searches. CC is the number of clock cycles. The selected architectures are the result of pruning to approximately 50% and quantization aware training at 8 bit precision. The best values are reported in bold.

Model	Lat. [ns] (cc)	II [ns] (cc)	DSP	LUT	FF	BRAM
Baseline [12]	**105 (21)**	**5** (1)	262 (2.1%)	155080 (9.0%)	25714 (0.7%)	4 (0.1%)
Optimal NAC [1]	125 (25)	60 (12)	**0**	**54075 (3.13%)**	**12016 (0.35%)**	8 (0.3%)
Optimal SNAC-Pack	140 (24)	70 (12)	**0**	57728 (3.34%)	12605 (0.36%)	**0**

6 Acknowledgments

This manuscript has been authored by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. BH and NT are supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy (DOE), Office of Science, Office of High Energy Physics. BH and NT are also supported under the DOE Early Career Research program under Award No. DE-0000247070. BH, JD, and NT are supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research under the "Real-time Data Reduction Codesign at the Extreme Edge for Science" Project(DE-FOA-0002501). JD is also supported by the DOE, Office of Science, Office of High Energy Physics Early Career Research program under Grant No. DE-SC0021187, and the U.S. National Science Foundation (NSF) Harnessing the Data Revolution (HDR) Institute for Accelerating AI Algorithms for Data Driven Discovery (A3D3) under Cooperative Agreement No. PHY-2117997. JW is supported by a WATCHEP fellowship sponsored by the DOE, Office of High-Energy Physics under Award No. DE-SC-0023527.

References

[1] Jason Weitz, Dmitri Demler, Luke McDermott, Nhan Tran, and Javier Duarte. Neural architecture codesign for fast physics applications. *Machine Learning: Science and Technology*, 6(3): 035009, jul 2025. doi: 10.1088/2632-2153/adede1. URL https://dx.doi.org/10.1088/2632-2153/adede1.

[2] Mohammad Mehdi Rahimifar, Hamza Ezzaoui Rahali, and Audrey C Therrien. rule4ml: an open-source tool for resource utilization and latency estimation for ml models on fpga. *Machine Learning: Science and Technology*, 6(1):015009, jan 2025. doi: 10.1088/2632-2153/ada71c. URL https://dx.doi.org/10.1088/2632-2153/ada71c.

[3] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. Hls4ml lhc jet dataset (150 particles), January 2020. URL https://doi.org/10.5281/zenodo.3602260.

[4] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):550–570, 2023. doi: 10.1109/TNNLS.2021.3100554.

[5] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.

[6] Olivia Weng. Neural network quantization for efficient inference: A survey. *arXiv preprint arXiv:2112.06126*, 2021.

[7] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2016.

[8] Javier Duarte et al. Fast inference of deep neural networks in FPGAs for particle physics. *JINST*, 13(07):P07027, 2018. doi: 10.1088/1748-0221/13/07/P07027.

[9] FastML Team. fastmachinelearning/hls4ml, 2024. URL https://github.com/fastmachinelearning/hls4ml.

[10] Nan Wu, Hang Yang, Yuan Xie, Pan Li, and Cong Hao. High-level synthesis performance prediction using gnns: benchmarking, modeling, and advancing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 49–54, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391429. doi: 10.1145/3489517.3530408. URL https://doi.org/10.1145/3489517.3530408.

[11] Kenneth O'Neal, Mitch Liu, Hans Tang, Amin Kalantar, Kennen DeRenard, and Philip Brisk. Hlspredict: cross platform performance prediction for fpga high-level synthesis. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359504. doi: 10.1145/3240765.3264635. URL https://doi.org/10.1145/3240765.3264635.

[12] Patrick Odagiu, Zhiqiang Que, Javier Duarte, Johannes Haller, Gregor Kasieczka, Artur Lobanov, Vladimir Loncar, Wayne Luk, Jennifer Ngadiuba, Maurizio Pierini, Philipp Rincke, Arpita Seksaria, Sioni Summers, Andre Sznajder, Alexander Tapper, and Thea K Årrestad. Ultrafast jet classification at the hl-lhc. *Machine Learning: Science and Technology*, 5(3): 035017, jul 2024. doi: 10.1088/2632-2153/ad5f10. URL https://dx.doi.org/10.1088/2632-2153/ad5f10.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. Evol. Comput.*, 6(2):182, 2002. doi: 10.1109/4235.996017.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[15] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. arXiv preprint arXiv:1903.01611, 2019.

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635, 2018.

[17] Benjamin Hawks, Jason Weitz, Dmitri Demler, Karla Tame-Narvaez, Dennis Plotnikov, Mohammad Mehdi Rahimifar, Hamza Ezzaoui Rahali, Audrey C. Therrien, Donovan Sproule, Elham E Khoda, Keegan A. Smith, Russell Marroquin, Giuseppe Di Guglielmo, Nhan Tran, Javier Duarte, and Vladimir Loncar. wa-hls4ml: A benchmark and surrogate models for hls4ml resource and latency estimation, 2025. URL https://arxiv.org/abs/2511.05615.