
Apollo: Transferable Architecture Exploration

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 The looming end of Moore’s Law and ascending use of deep learning drives the
2 design of custom accelerators that are optimized for specific neural architectures.
3 Architecture exploration for such accelerators forms a challenging constrained opti-
4 mization problem over a complex, high-dimensional and structured input space with
5 a costly to evaluate objective function. Existing approaches for accelerator design
6 are sample-inefficient and do not transfer knowledge between related optimizations
7 tasks with different design constraints (e.g. area budgets) or neural architecture
8 configurations. In this work, we propose a transferable architecture exploration
9 framework, dubbed APOLLO, that leverages recent advances in black-box function
10 optimization for sample-efficient accelerator design. We use this framework to
11 optimize accelerator configurations of a diverse set of neural architectures with
12 alternative design constraints. We show that our framework finds high reward
13 design configurations (up to 24.6% speedup) more sample-efficiently than a baseline
14 black-box optimization approach. We further show that by transferring knowledge
15 between target architectures with different design constraints, APOLLO is able to
16 find optimal configurations faster and often with better objective value (up to 25%
17 improvements). This encouraging outcome portrays a promising path forward to
18 facilitate generating higher quality accelerators.

19 1 Introduction

20 The ubiquity of customized accelerators demands efficient architecture exploration approaches,
21 especially for the design of neural network accelerators. However, optimizing the parameters of
22 accelerators is daunting optimization task that generally requires expert knowledge [11, 28] since
23 the search space is exponentially large while the objective function is a black-box and costly to
24 evaluate. Constraints imposed on parameters further complicate the identification of valid accelerator
25 configurations. Constrains can arise from hardware limitations or if the evaluation of a configuration
26 is impossible or too expensive [29].

27 To address the aforementioned challenges, we introduce a general architecture exploration framework,
28 dubbed APOLLO, that leverages the recent advances in black-box optimization to facilitate finding opti-
29 mal design configurations under different design constraints. We demonstrate how leveraging tailored
30 optimization strategies for complex and high-dimensional space of architecture exploration yields large
31 improvements (up to 24.6%) with a reasonably small number of evaluations ($\approx 0.0004\%$ of the search
32 space). Finally, we study the impact of transfer learning between architecture exploration tasks with dif-
33 ferent design constraints in further reducing the number of hardware evaluations. The following outlines
34 the contributions of APOLLO, making the first transferable architecture exploration infrastructure:

- 35 • **End-to-end architecture exploration framework.** We introduce and develop APOLLO, an
36 end-to-end and highly configurable framework for architecture exploration. The proposed
37 framework tunes accelerator configurations for a target set of workloads with a relatively small
38 number of hardware evaluations. As hardware simulations are generally time-consuming, reducing

39 the number of these simulations not only shortens the design cycle for accelerators, but also provides
 40 an effective way to adapt the accelerator itself to various target workloads.

- 41 • **Supporting various optimization strategies.** APOLLO introduces and employs a variety of
 42 optimization strategies to facilitate the analysis of optimization performance in the context of
 43 architecture exploration. Our evaluations results show that evolutionary and population-based
 44 black-box optimization strategies yield the best accelerator configurations (up to 24.6% speedup)
 45 compared to a baseline black-box optimization with only $\approx 2\text{K}$ number of hardware evaluations
 46 ($\approx 0.0004\%$ of search space).
- 47 • **Transfer learning for architecture exploration.** Finally, we study and explore transfer learning
 48 between architecture exploration tasks with different design constraints showing its benefit in
 49 improving the optimization results and sample-efficiency. Our results show that transfer learning
 50 not only improves the optimization outcome (up to 25%) compared to independent exploration,
 51 but also reduces the number of hardware evaluations.

52 2 Methodology

53 **Problem definition.** The objective in APOLLO (architecture exploration) is to discover a set of feasible
 54 accelerator parameters (h) for a set of workloads (w) such that a desired objective function (f), e.g.
 55 weighted average of runtime, is minimized under an optional set of user-defined constraints, such
 56 as area (α) and/or runtime budget (τ).

$$\begin{aligned} \min_{h,w} \quad & f(h,w) \\ \text{s.t.} \quad & \text{Area}(h) \leq \alpha \\ & \text{Latency}(h,w) \leq \tau \end{aligned} \tag{1}$$

57 The manifold of architecture search generally contains infeasible points [28], for example due to
 58 impractical hardware implementation for a given set of parameters or impossible mapping of workloads
 59 to an accelerator. As such, one of the main challenges for architecture exploration is to effectively
 60 sidestep these infeasible points. We present and analyze the performance of optimization strategies to
 61 reduce the number of infeasible trials in Section 3.

62 **Neural models.** We evaluate APOLLO on two variations of MobileNet [33, 15] and five in-house neural
 63 models with distinct accelerator resource requirements. The neural model configurations, including
 64 their target domain, number of layers, and total filter sizes are detailed in Table 1. In the multi-model
 study, the workload contains MobileNetV2 [33], MobileNetEdge [15], M3, M4, M5, M6.

Table 1: The detailed description of the neural models, their domains, number of layers, parameter size in megabytes, and number of MAC operations in million.

Name	Domain	# of layers	Params (MB)	# of MACs
MobileNetV2 [33]	Image Classification	76	3.33	301 M
MobileNetEdge [16]	Image Classification	93	3.88	991 M
M3	Object Detection	93	2.19	464 M
M4	Object Detection	111	0.42	107 M
M5	Object Detection	60	6.29	1721 M
M6	Semantic Segmentation	62	0.37	591 M
M7	OCR	56	0.30	5.19 M

65

66 **Accelerator search space.** In this work, we use an in-house and highly parameterized edge accelerator.
 67 The accelerator contains a 2D array of processing elements (PE) with multiple compute lanes and
 68 dedicated register files, each operating in single-instruction multiple-data (SIMD) style with multiply-
 69 accumulate (MAC) compute units. There are distributed local and global buffers that are shared across
 70 the compute lanes and PEs, respectively. We designed a cycle-accurate simulator that faithfully models
 71 the main microarchitectural details and enables us to perform architecture exploration. Table 2 outlines
 72 the microarchitectural parameters (e.g. compute, memory, or bandwidth) and their number of discrete
 73 values in the search space. The total number of design points explored in APOLLO is nearly 5×10^8 .

74 2.1 Optimization Strategies

75 In APOLLO, we study and analyze the performance of following optimization methods.

Table 2: The microarchitecture parameters, their type, and number of discrete values per parameter. The total number of design points per each study is 452,760,000.

Accelerator Parameter	# discrete values	Accelerator Parameter	# discrete values
# of PEs-X	10	# of PEs-Y	10
Local Memory	7	# of SIMD units	7
Global Memory	11	# of Compute lanes	10
Instruction Memory	4	Parameter Memory	5
Activation Memory	7	I/O Bandwidth	6

76 **Evolutionary.** Performs evolutionary search using a population of K individuals, where the genome
 77 of each individual corresponds to a sequence of discretized accelerator configurations. New individuals
 78 are generated by selecting for each individual two parents from the population using tournament
 79 selecting, recombining their genomes with some crossover rate γ , and mutating the recombined
 80 genome with some probability μ . Following Real et al. [31], individuals are discarded from the
 81 population after a fixed number of optimization rounds (‘death by old age’) to promote exploration.
 82 In our experiments, we use the default parameters $K = 100$, $\gamma = 0.1$, and $\mu = 0.01$.

83 **Model-Based Optimization (MBO).** Performs model-based optimization with automatic model se-
 84 lection following [2]. At each optimization round, a set of candidate regression models are fit on the data
 85 acquired so far and their hyper-parameter optimized by randomized search and five fold cross-validation.
 86 Models with a cross-validation score above a certain threshold are ensembled to define an acquisition
 87 function. The acquisition is optimized by evolutionary search and the proposed accelerator configu-
 88 rations with the highest acquisition function values are used for the next objective function evaluation.

89 **Population-Based black-box optimization (P3BO).** Uses an ensemble of optimization methods,
 90 including Evolutionary and MBO, which has been recently shown to increase sample-efficiency
 91 and robustness [3]. Acquired data are exchanged between optimization methods in the ensemble,
 92 and optimizers are weighted by their past performance to generate new accelerator configurations.
 93 Adaptive-P3BO is an extension of P3BO which further optimizes the hyper-parameters of optimizers
 94 using evolutionary search, which we use in our experiments.

95 **Random.** Samples accelerator configurations uniformly at random from the defined search space.

96 **Vizier.** An alternative approach to MBO based on Bayesian optimization with a Gaussian process
 97 regressor and the expected improvement acquisition function, which is optimized by gradient-free
 98 hill-climbing [14]. Categorical variables are one-hot encoded.

99 We use the Google Vizier framework [14] with the optimization strategies described above for perform-
 100 ing our experiments. We use the default hyper-parameter of all strategies [14, 3]. Each optimization
 101 strategy is allowed to propose 4096 trials per experiment. We repeat each experiment five times with
 102 different random seeds and set the reward of infeasible trials to zero. To parallelize hardware simu-
 103 lations, we use 256 CPU cores each handling one hardware simulation at a time. We further run each
 104 optimization experiment asynchronously with 16 workers that can evaluate up to 16 trials in parallel.

105 3 Evaluation

106 **Single model architecture search.** For the first experiment, we define the optimization problem
 107 as maximizing throughput per area (e.g. $\frac{1}{latency} \times \frac{1}{area}$) for each neural model without defining
 108 any design constraints. Figure 1 depicts the cumulative reward across various number of trials.
 109 Compared to Vizier, Evolutionary and P3BO improve the throughput per area by 4.3% (up to 12.2% in
 110 MobileNetV2), on average. In addition, both Evolutionary and P3BO yield lower variance across multiple
 111 runs suggesting a more robust optimization method for architecture search.

112 **Multi-model architecture search.** For multi-model architecture search, we define the optimization as
 113 maximizing $geomean(speedup)$ across all the evaluated models (See Section 2) while imposing area
 114 budget constraints of 6.8 mm^2 , 5.8 mm^2 , and 4.8 mm^2 . Note that, as the area budget becomes stricter,
 115 the number of infeasible trials increases. The baseline runtime numbers are obtained from a production-
 116 ized edge accelerator. Figure 2 demonstrates the cumulative reward (e.g. $geomean(speedup)$) across
 117 various number of sampled trials. Across the studied optimization strategies, P3BO delivers the highest
 118 improvements across all the design constraints. Compared to Vizier, P3BO improves the speedup by

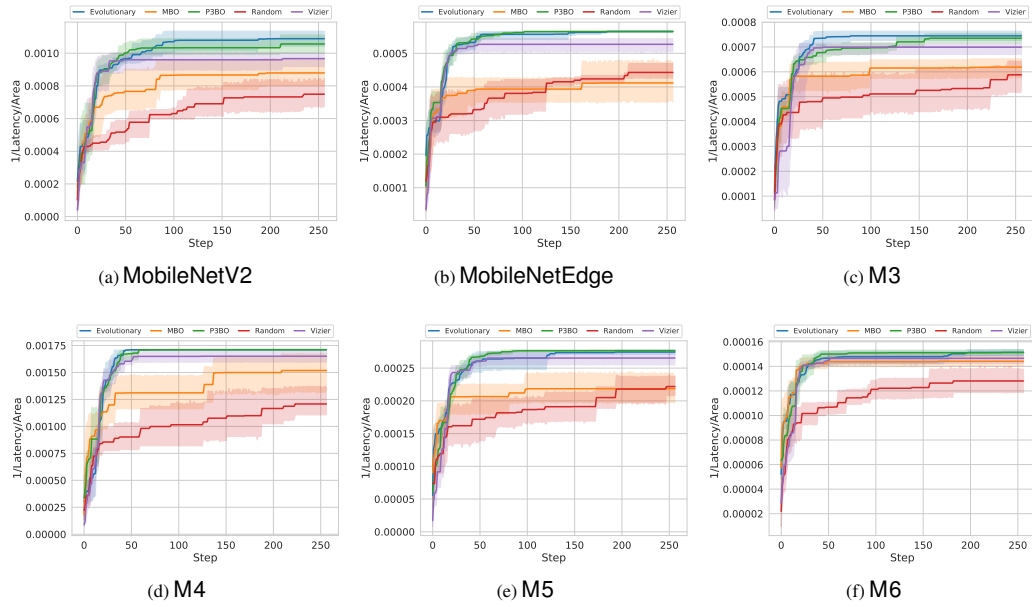


Figure 1: Performance of optimization strategies across various neural models in maximizing the throughput per area ($\frac{1}{\text{Latency}} \times \frac{1}{\text{Area}}$) (\uparrow is better). The shaded area depicts the 95% bootstrap confidence interval over five runs. Evolutionary and P3BO find high reward accelerator configurations faster than alternative optimization strategies.

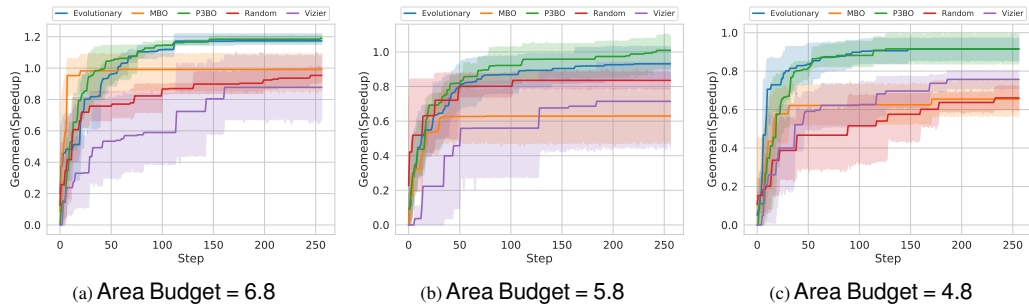


Figure 2: Performance of optimization strategies in maximizing $\text{geomean}(\text{speedup})$ (\uparrow is better) under alternative area budget constrains. The shaded area depicts the 95% bootstrap confidence interval over five runs. The baseline latency numbers are from a productionized edge accelerator. As the area constraint becomes tighter (more infeasible points), the improvement by P3BO increases.

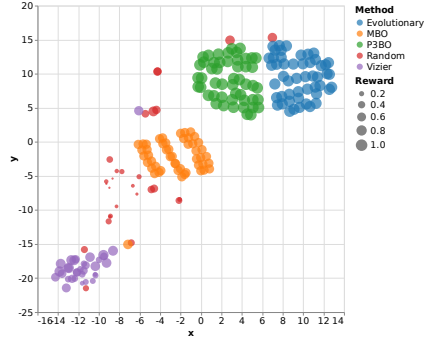
119 6.2%, 16.6%, and 24.6% for area budget 6.8 mm², 5.8 mm², and 4.8 mm², respectively. These results
 120 demonstrate that as the design space becomes more constrained (e.g. more infeasible points), the
 121 improvement by P3BO increases, showing its performance in navigating the search space better.

122 **Analysis of infeasible trials.** To better understand the effectiveness of each optimization strategy in
 123 selecting feasible trials and unique trials, we define two metrics *feasibility ratio* and *uniqueness ratio*,
 124 respectively. The feasibility (uniqueness) ratio defines the fraction of feasible (unique) trials over
 125 the total number of sampled trials. Higher ratios generally indicate improved exploration of feasible
 126 regions. Table 3 summarizes the feasibility and uniqueness ratio of each optimization strategy for area
 127 budget 6.8 mm², averaged over multiple optimization runs. MBO yields the highest avg. feasibility
 128 ratio of ≈ 0.803 while Random shows the lowest ratio of ≈ 0.009 . While MBO features a high feasibility
 129 ratio, it underperforms compared to other optimization strategies in finding accelerator configurations
 130 with high performance. The key reason attributed to this behavior for MBO is its low performance
 131 (0.236) in identifying unique accelerator parameters compared to other optimization strategies.

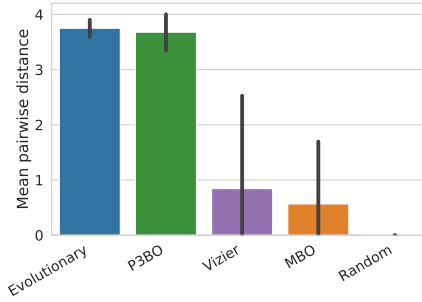
132 **Diversity of architecture configurations.** A desired property of optimizers is to not only find a single
 133 but a diverse set of architecture configurations with a high reward that can be tested downstream. We
 134 quantified the ability of optimizers to find diverse configurations qualitatively by visualizing the 50 best
 135 unique trials found by each method using tSNE. Figure 3a shows that Evolutionary and P3BO find both
 136 higher-reward and more diverse configurations compared to alternative methods with the exception

Table 3: The average feasibility and uniqueness ratio across five runs for architecture search with an area budget of 6.8 mm^2 (see Figure 2a).

	Evolutionary	MBO	P3BO	Random	Vizier
Avg. Feasibility Ratio (\uparrow better)	0.362	0.803	0.347	0.009	0.012
Avg. Uniqueness Ratio (\uparrow better)	0.891	0.236	0.848	1.0	0.979



(a) tSNE of the 50 best configurations found by different methods.



(b) Mean pairwise Euclidean distance of all configuration with a reward above the 75th percentile of the maximum reward. Error bars show the variance across 5 optimization runs.

Figure 3: Diversity quantification of architecture configurations found by different methods for an area budget of 4.8 mm^2 .

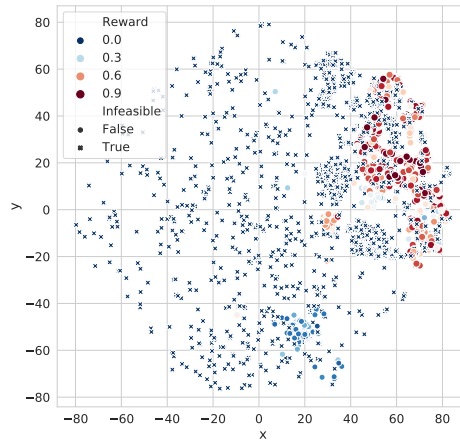


Figure 4: tSNE of all trials (including infeasible ones) proposed by the Evolutionary algorithm for an area budget of 4.8 mm^2 . Shows the large fraction of infeasible trials (crosses) vs feasible (circles) trials.

137 of Random. This finding is supported quantitatively by Figure 3b, which shows the mean pairwise
 138 Euclidean distance of configurations with a reward above the 75th percentile of the maximum reward.
 139 The mean pairwise distance of Random is zero since it did not find any configurations with a reward
 140 above the 75th percentile. To further visualize the search space in architecture exploration, Figure 4
 141 shows the tSNE visualization of all trials proposed by the Evolutionary method for an area budget of
 142 4.8 mm^2 . This figure shows the large number of infeasible trials in the space and the proximity of
 143 low- and high-performing trials, which renders identifying high-performing trials challenging.

144 **Transfer learning between optimizations with different constraints.** We analyze the effect of
 145 transfer learning between architecture search tasks with different area budgets. To create the source
 146 tasks, we select 100 unique trials from optimization studies with area budget constraint of 6.8 mm^2
 147 (See Fig. 2a) under two criteria. First, the area consumption of the selected trials must satisfy the area
 148 budget (4.8 mm^2) of the target task. Second, the objective function value (reward) of the selected
 149 trials must be below a predefined threshold. In our experiments, we create two source tasks with an
 150 objective value of 0.8 and 0.4, respectively, which we chose to better understand the impact of low-

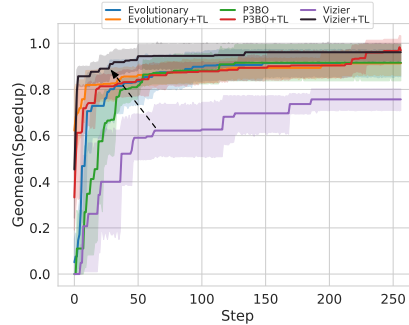


Figure 5: Comparing optimization strategies in maximizing the `geomean(speedup)` (\uparrow is better) without transfer learning (top row in the legend) and with transfer learning (bottom row in the legend) for area budget 4.8 mm^2 . Transfer learning enables finding higher performance accelerator configurations in fewer steps.

151 and high-value rewards. We use the selected trials to seed the optimization of the target task, which has
 152 an area budget of 4.8 mm^2). Figure 5 shows the results. All the optimization strategies find high reward
 153 trials in fewer steps with transfer learning than without. The improvement is most pronounced for
 154 Vizier, which finds trials with a reward of ≈ 1.0 with transfer learning compared to only ≈ 0.8 without
 155 transfer learning. This suggests that Vizier uses the selected trials from the source task more efficiently
 156 than Evolutionary and P3BO for optimizing the target task.

157 In our implementation, Evolutionary and P3BO simply use the 100 unique and feasible trails from the
 158 source task to initialize the population of evolutionary search. Instead, Vizier uses a more advanced
 159 transfer learning approach based on a stack of Gaussian process regressors (see Section 3.3 of Golovin
 160 et al. [14]), which may account for the performance improvement. We leave extending Evolutionary
 161 and P3BO by more advanced transfer learning approaches as future work.

162 **Comparison to exhaustive exploration.** To understand the optimal design point, we perform a
 163 semi-exhaustive search within the search space. Since the search space has almost 5×10^8 design
 164 points, it is merely not practical to perform a fully-exhaustive search. As such, we manually prune
 165 the search space using domain knowledge where the design points are within a typical edge accelerator
 166 configuration (e.g. total memory size within 4–16 MB, total number of PEs within 2–16, etc.).
 167 Additionally, we perform a cheaper area estimation to reject design points before performing expensive
 168 cycle-level simulations. Using this pruning approach, we reduced the size of search space to around
 169 3K samples. We observe that P3BO can reach the best configurations found by the semi-exhaustive
 170 search by performing far fewer evaluations ($1.36 \times$ less). Another interesting observation is that
 171 for the multi-model experiment targeting 6.8 mm^2 , P3BO actually finds a design slightly better than
 172 semi-exhaustive with 3K-sample search space. We observe that the design uses a very small memory
 173 size (3MB) in favor of more compute units. This leverages the compute-intensive nature of vision
 174 workloads, which was not included in the original semi-exhaustive search space. This demonstrates
 175 the need of manual search space engineering for semi-exhaustive approaches, whereas learning-based
 176 optimization methods leverage large search spaces reducing the manual effort.

177 4 Related Work

178 While inspired by related work, APOLLO is fundamentally different from classic methodologies in
 179 design space exploration: (1) we develop a platform to compare the effectiveness of a wide range of
 180 optimization algorithms; and (2) we are the first work, to the best of our knowledge, that leverages
 181 transfer learning between architecture exploration tasks with different design constraints showing
 182 how transfer learning slashes the time for discovering new accelerator configurations. Related work to
 183 APOLLO embodies three broad research categories of black-box optimization, architecture exploration,
 184 and transfer learning. Below, we overview the most relevant work in these categories.

185 **Black-box optimization.** Black-box optimization has been broadly applied across different domains
 186 and appeared under various optimization categories, including Bayesian [37, 3, 24, 34, 42, 36, 6, 8],
 187 evolutionary [1, 39, 20], derivative-free [23, 32, 12], and bandit [7, 25, 38, 13]. APOLLO benefits
 188 from advances in black-box optimization and establishes a basis for leveraging this broad range of
 189 optimization algorithms in the context of accelerator design. In this work, we extensively studied

190 the effectiveness of some of these black-box optimization algorithms, namely random search [14],
191 Bayesian optimization [14], evolutionary algorithms [3], and ensemble methods [3] in discovering
192 optimal accelerator configurations under different design objectives and constraints.

193 **Design space exploration.** Design space exploration in computer systems has been always
194 an active research and has become even more crucial due to the surge of specialized hard-
195 ware [30, 18, 40, 28, 10, 21, 5, 4]. Hierarchical-PABO [30] and FlexiBO [18] use multi-objective
196 Bayesian optimization for neural network accelerator design. In order to reduce the use of computa-
197 tional resources, Sun et al. [40] apply genetic algorithm to design CNN models without modifying the
198 underlying architecture. HyperMapper [28] uses a random forest in the automatic tuning of hardware ac-
199 celerator parameters in a multi-objective setting. HyperMapper optionally uses continuous distributions
200 to model the search space variables as a means to inject prior knowledge into the search space.

201 **Transfer learning.** Transfer learning exploits the acquired knowledge in some tasks to facilitate
202 solving similar unexplored problems more efficiently, e.g. consuming a fewer number of data samples
203 and/or outperforming previous solutions. Transfer learning has been explored extensively and applied
204 to various domains [27, 44, 43, 17, 19, 9, 35, 26, 22, 41]. Due to the expensive-to-evaluate nature of
205 hardware evaluations, transfer learning seems to be a practical mechanism for architecture exploration.
206 However, using transfer learning for architecture exploration and accelerator design is rather less
207 explored territory. APOLLO is one of the first methods to bridge this gap between transfer learning
208 and architecture exploration.

209 5 Conclusion

210 In this paper, we propose APOLLO, a framework for sample-efficient architecture exploration for large
211 scale design spaces. The benefits of APOLLO are most noticeable when architecture configurations
212 are costly to evaluate, which is a common trait in various architecture optimization problems. Our
213 framework also facilitates the design of new accelerators with different design constraints by leveraging
214 transfer learning. Our results indicate that transfer learning is effective in improving the target archi-
215 tecture exploration, especially when the optimization constraints have tighter bounds. Finally, we show
216 that the evolutionary algorithms used in this work yield more diverse accelerator designs compared
217 to other studied optimization algorithms, which can potentially discover overlooked architectures.
218 Architecture exploration is just one use case in the accelerator design process that is bolstered by
219 APOLLO. The evolution of accelerator architectures mandates broadening the scope of optimizations
220 to the entire computing stack, including scheduling and mapping, that potentially yields higher benefits
221 at the cost of handling more complex optimization problems. We argue that such co-evolution between
222 the cascaded layers of the computing stack is inevitable in designing efficient accelerators honed for
223 a diverse category of applications. This is an exciting path forward for future research directions.

224 References

- 225 [1] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue,
226 and Mengjie Zhang. A survey on evolutionary machine learning. *Journal of the Royal Society*
227 *of New Zealand*, 49(2):205–228, 2019.
- 228 [2] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy,
229 and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In
230 *International Conference on Learning Representations*, 2019.
- 231 [3] Christof Angermueller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin
232 Murphy, Lucy Colwell, and D Sculley. Population-based black-box optimization for biological
233 sequence design. *arXiv preprint arXiv:2006.03227*, 2020.
- 234 [4] Jason Ansel, Shoaib Kamil, Kalyan Veeramachaneni, Jonathan Ragan-Kelley, Jeffrey Bosboom,
235 Una-May O’Reilly, and Saman Amarasinghe. OpenTuner: An extensible framework for program
236 autotuning. In *Proceedings of the 23rd international conference on Parallel architectures and*
237 *compilation*, pp. 303–316, 2014.
- 238 [5] Prasanna Balaprakash, Ananta Tiwari, Stefan M Wild, Laura Carrington, and Paul D Hovland.
239 AutoMOMML: Automatic multi-objective modeling with machine learning. In *International*
240 *Conference on High Performance Computing*, pp. 219–239. Springer, 2016.

- 241 [6] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-
242 parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554,
243 2011.
- 244 [7] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and
245 contextual bandits. *arXiv preprint arXiv:1904.10040*, 2019.
- 246 [8] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of
247 expensive cost functions, with application to active user modeling and hierarchical reinforcement
248 learning. *arXiv preprint arXiv:1012.2599*, 2010.
- 249 [9] T. Chugh, M. Singh, S. Nagpal, R. Singh, and M. Vatsa. Transfer learning based evolutionary
250 algorithm for composite face sketch recognition. In *2017 IEEE Conference on Computer Vision
251 and Pattern Recognition Workshops (CVPRW)*, pp. 619–627, 2017.
- 252 [10] Jason Cong, Peng Wei, Cody Hao Yu, and Peng Zhang. Automated accelerator generation and
253 optimization with composable, parallel and pipeline architecture. In *2018 55th ACM/ESDA/IEEE
254 Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2018.
- 255 [11] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free
256 optimization*. SIAM, 2009.
- 257 [12] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free
258 optimization*. SIAM, 2009.
- 259 [13] Josep Ginebra and Murray K Clayton. Response surface bandits. *Journal of the Royal Statistical
260 Society: Series B (Methodological)*, 57(4):771–784, 1995.
- 261 [14] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley.
262 Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD
263 international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.
- 264 [15] Suyog Gupta and Berkin Akin. Accelerator-aware neural network design using automl. *arXiv
265 preprint arXiv:2003.02838*, 2020.
- 266 [16] Andrew Howard and Suyog Gupta. Introducing the Next Generation of On-Device Vision
267 Models: MobileNetV3 and MobileNetEdgeTPU. [https://ai.googleblog.com/
268 2019/11/introducing-next-generation-on-device.html](https://ai.googleblog.com/2019/11/introducing-next-generation-on-device.html), 2020.
- 269 [17] Kotthoff Lars Vanschoren Joaquin Hutter, Frank (ed.). *Automated Machine Learning: Methods,
270 Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer
271 International Publishing, 2019.
- 272 [18] Md Shahriar Iqbal, Jianhai Su, Lars Kotthoff, and Pooyan Jamshidi. Flexibo: Cost-aware
273 multi-objective optimization of deep neural networks. *arXiv preprint arXiv:2001.06588*, 2020.
- 274 [19] Min Jiang, Zhongqiang Huang, Liming Qiu, Wenzhen Huang, and Gary G Yen. Transfer
275 learning-based dynamic multiobjective optimization algorithms. *IEEE Transactions on
276 Evolutionary Computation*, 22(4):501–514, 2017.
- 277 [20] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of
278 expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.
- 279 [21] David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben
280 Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, et al. Spatial: A language
281 and compiler for application accelerators. In *Proceedings of the 39th ACM SIGPLAN Conference
282 on Programming Language Design and Implementation*, pp. 296–311, 2018.
- 283 [22] Barış Koçer and Ahmet Arslan. Genetic transfer learning. *Expert Systems with Applications*,
284 37(10):6997–7002, October 2010.
- 285 [23] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods.
286 *arXiv preprint arXiv:1904.11585*, 2019.

- 287 [24] Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. Constrained bayesian
288 optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.
- 289 [25] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar.
290 Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of*
291 *Machine Learning Research*, 18(1):6765–6816, 2017.
- 292 [26] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning
293 using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, May 2015.
- 294 [27] Alan Tan Wei Min, Yew-Soon Ong, Abhishek Gupta, and Chi-Keong Goh. Multiproblem
295 Surrogates: Transfer Evolutionary Multiobjective Optimization of Computationally Expensive
296 Problems. *IEEE Transactions on Evolutionary Computation*, 23(1):15–28, February 2019.
- 297 [28] Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. In
298 *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer*
299 *and Telecommunication Systems (MASCOTS)*, pp. 347–358. IEEE, 2019.
- 300 [29] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying,
301 Anurag Mukkara, Rangharajan Venkatesan, Bruce Khailany, Stephen W Keckler, and Joel Emer.
302 Timeloop: A systematic approach to dnn accelerator evaluation. In *ISPASS*. IEEE, 2019.
- 303 [30] Maryam Parsa, John P Mitchell, Catherine D Schuman, Robert M Patton, Thomas E Potok,
304 and Kaushik Roy. Bayesian multi-objective hyperparameter optimization for accurate, fast, and
305 efficient neural network accelerator design. *Frontiers in Neuroscience*, 14:667, 2020.
- 306 [31] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image
307 classifier architecture search. In *Proceedings of the aai conference on artificial intelligence*,
308 volume 33, pp. 4780–4789, 2019.
- 309 [32] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms
310 and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293,
311 2013.
- 312 [33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen.
313 Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference*
314 *on computer vision and pattern recognition*, pp. 4510–4520, 2018.
- 315 [34] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking
316 the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104
317 (1):148–175, 2015.
- 318 [35] Alistair Shilton, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Regret bounds for transfer
319 learning in bayesian optimisation. In *Machine Learning Research: Proceedings of the 20th*
320 *Artificial Intelligence and Statistics International Conference*, pp. 1–9. Journal of Machine
321 Learning Research (JMLR), 2017.
- 322 [36] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine
323 learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.
- 324 [37] Artur Souza, Luigi Nardi, Leonardo B Oliveira, Kunle Olukotun, Marius Lindauer, and Frank
325 Hutter. Prior-guided bayesian optimization. *arXiv preprint arXiv:2006.14608*, 2020.
- 326 [38] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian
327 process optimization in the bandit setting: No regret and experimental design. *arXiv preprint*
328 *arXiv:0912.3995*, 2009.
- 329 [39] Chaoli Sun, Yaochu Jin, and Ying Tan. Semi-supervised learning assisted particle swarm
330 optimization of computationally expensive problems. In *Proceedings of the Genetic and*
331 *Evolutionary Computation Conference*, pp. 45–52, 2018.
- 332 [40] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. Automatically designing
333 cnn architectures using the genetic algorithm for image classification. *IEEE Transactions on*
334 *Cybernetics*, 2020.

- 335 [41] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In
336 *Advances in neural information processing systems*, pp. 2004–2012, 2013.
- 337 [42] Ke Tang, Fei Peng, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios with
338 automated constituent algorithms selection. *Information Sciences*, 279:94–104, 2014.
- 339 [43] Tinu Theckel Joy, Santu Rana, Sunil Gupta, and Svetha Venkatesh. A flexible transfer learning
340 framework for Bayesian optimization with convergence guarantee. *Expert Systems with
341 Applications*, 115:656–672, January 2019.
- 342 [44] Michael Volpp, Lukas P. Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter,
343 and Christian Daniel. Meta-Learning Acquisition Functions for Transfer Learning in Bayesian
344 Optimization. *arXiv:1904.02642 [cs, stat]*, February 2020.