Carbon-Efficient Neural Architecture Search

Yiyang Zhao
Worcester Polytechnic Institute
yzhao10@wpi.edu

Tian Guo
Worcester Polytechnic Institute
tian@wpi.edu

ABSTRACT

This work presents a novel approach to neural architecture search (NAS) that aims to reduce energy costs and increase carbon efficiency during the model design process. The proposed framework, called carbon-efficient NAS (CE-NAS), consists of NAS evaluation algorithms with different energy requirements, a multi-objective optimizer, and a heuristic GPU allocation strategy. CE-NAS dynamically balances energy-efficient sampling and energy-consuming evaluation tasks based on current carbon emissions. Using a recent NAS benchmark dataset and two carbon traces, our trace-driven simulations demonstrate that CE-NAS achieves better carbon and search efficiency than the three baselines.

CCS CONCEPTS

• **Social and professional topics** → **Sustainability**; • **Computing methodologies** → **Neural networks**; *Search methodologies.*

KEYWORDS

Sustainability, carbon aware, neural architecture search

ACM Reference Format:
Yiyang Zhao and Tian Guo. 2023. Carbon-Efficient Neural Architecture Search. In *2nd Workshop on Sustainable Computer Systems (HotCarbon '23), July 9, 2023, Boston, MA, USA.* ACM, Boston, MA, USA, 7 pages. https://doi.org/10.1145/3604930.3605708

1 INTRODUCTION

Deep Learning (DL) has become an increasingly important field in computer science, with applications ranging from healthcare to transportation to energy management. However, DL training is notoriously energy-intensive and significantly contributes to today's carbon emissions [20, 31]. The main culprit comes down to the iterative nature of training, which requires evaluating and updating model parameters based on a large amount of data.

Neural architecture search (NAS) has emerged as a means to automate the design of DL models. At the high level, NAS often involves leveraging search algorithms to explore a massive architecture design space, ranging from hundreds of millions to trillions of candidates [15, 19, 23, 29–31], by training and evaluating a subset of architectures. In searching for the best architecture for different application domains, many NAS works have reported using thousands of GPU-hours [19, 22, 23, 30, 31].

The environmental impact of NAS, if left untamed, can be substantial. While recent works have significantly improved the search efficiency of NAS [19, 22, 23, 29, 31], e.g., reducing the GPU-hours to tens of hours without sacrificing the architecture quality, there still lacks *conscious efforts in reducing carbon emissions.* As noted in a recent vision paper by Bashir et al. [2], energy efficiency can help reduce carbon emissions but is not equivalent to carbon efficiency.

Table 1: Comparison of energy-efficient NAS evaluation methods. *Eval. cost* refers to the cost of obtaining the evaluation results. *Init. cost* describes additional dataset preparation and the time required for training the model (e.g., supernet or predictor). *Accuracy* measures the rank correlation between the evaluation method and the actual rank. Predictor-based methods require *Extra data* as a training set to construct the prediction model.

Method	Eval. cost	Init. cost	Accuracy	Extra data
One-shot [4, 15, 18, 26, 29]	Low	Low	Intermediate	No
Predictor [9, 14, 23]	Low	High†	High†	Yes
Low-fidelity [12, 15, 19, 23, 31]	High	None	Intermediate‡	No
Gradient Proxy [25]	Low	Low	Intermediate	No

† It depends on the size of extra data.
‡ It depends on the extent of the fidelity.

This paper aims to bridge the gap between carbon and energy efficiency with a new NAS framework designed to be carbon-aware from the outset.

The proposed framework, termed CE-NAS, will tackle the high carbon emission problem from two main aspects. First, CE-NAS will regulate the model design process by *deciding when to use different NAS evaluation strategies based on the varying carbon intensity.* To elaborate, given a fixed amount of GPU resources, CE-NAS will allocate more resources to energy-efficient NAS evaluation strategies, e.g., one-shot NAS [3–5, 15, 18, 29], during periods of high carbon intensity. Conversely, during periods of low carbon intensity, CE-NAS will shift the focus to running energy-intensive but more effective NAS evaluation strategies, e.g., vanilla NAS [19, 22, 23, 31]. Second, the CE-NAS framework will support energy and carbon-efficient DL model design via multi-objective optimization. Specifically, we will leverage a recent learning-based multi-objective optimizer LaMOO [30] and integrate it to CE-NAS to achieve search efficiency.

Based on these two design guidelines, we sketch out the basis of the proposed CE-NAS framework in Figure 1 and implement a trace-driven simulator to investigate the promise of CE-NAS in improving carbon and search efficiencies. Using carbon emission traces from electricityMap [17] and a new NAS dataset called HW-NASBench [13], we show that CE-NAS has the least relative carbon emissions and only marginally lower search efficiency compared to vanilla LaMOO [30]. Based on our investigation, we believe there are many fruitful directions in the context of CE-NAS which we outline in §5. We hope this discussion will serve as the blueprint and a baseline for building a carbon-efficient NAS framework.

2 NAS AND ITS CARBON IMPACT

Neural architecture search (NAS) is a technique for automating the design of neural network architectures. NAS aims to find an optimal network architecture that performs well on a specific task without

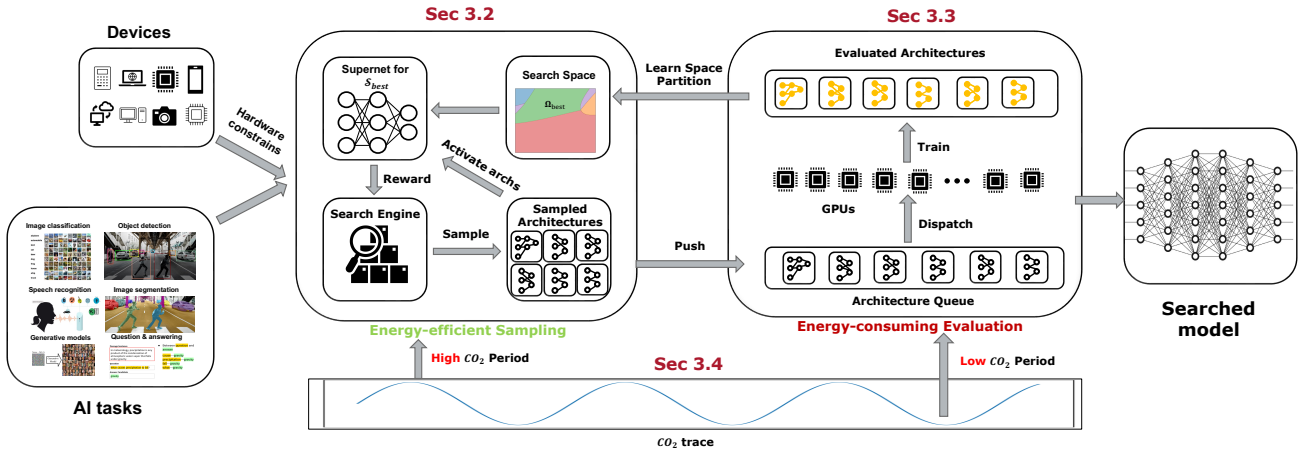


Figure 1: An overview of CE-NAS. The sampling and evaluation tasks will be dispatched with different GPU resources based on carbon emission intensity during the neural architecture search.

human intervention. NAS-designed neural architectures achieved state-of-the-art performances on many AI applications, such as image classification, object detection, and image segmentation [11, 19, 23, 24, 31].

However, NAS typically requires significant computational resources (e.g., GPUs) to find the optimal architecture, with most of these resources being used for architecture evaluation. For example, Zoph et al. [31] used 800 GPUs for 28 days, equivalent to 22,400 GPU hours, to obtain the final architectures. Strubell et al. [20] found that a single NAS solution can emit as much carbon as five cars during its lifetime. These findings highlight the need for energy and carbon-efficient NAS methods to reduce the environmental impact of AI research.

Existing works on energy-efficient NAS often focus on improving the evaluation phase, e.g., via weight-sharing [4, 6, 15, 18, 26], performance predictor [4, 14, 23, 28], low-fidelity NAS evaluation [12, 15, 19, 22, 23, 31], and gradient proxy NAS [25]. A comparison of these methods can be found in Table 1. Weight-sharing leverages the accuracy estimated by a *supernet* as a proxy for the true architecture performance, while gradient proxy NAS uses the gradient as a proxy. These proxy-based methods, although incurring smaller search costs in terms of energy, can have lower search efficiency because their estimated architecture accuracy may have poor rank correlation [29]. Performance predictors provide a more accurate performance prediction than weight-sharing and gradient proxy NAS. Still, their accuracy heavily relies on the volume and quality of the training data, which can be very expensive to create [27, 28]. Low-fidelity evaluation still requires training each searched architecture, leading to limited energy savings.

In short, utilizing existing energy-efficient NAS methods requires careful consideration of the search quality and efficiency trade-offs; however, naively applying these methods may not even lead to energy savings, not to mention lower carbon emissions. In this work, we achieve the goals of search efficiency, search quality, and carbon efficiency by leveraging a generic multi-objective optimizer [30], a mix of energy-efficient [3, 15, 18, 29] and energy-consuming [19, 22, 23, 31] evaluation methods, and a carbon-aware GPU resource allocation strategy.

3 RESEARCH ROADMAP

In this section, we present an overview of the proposed CE-NAS framework (Figure 1) and sketch the basis for each component. We hope this discussion will serve as the blueprint and a baseline for designing a carbon-efficient NAS framework.

3.1 CE-NAS Overview

As observed in [2], grid carbon emissions vary geographically and temporally based on the mix of active generators. Consequently, different carbon emissions arise even when consuming the same electricity at different locations or times. Operating the NAS framework without considering costs across every carbon period will lead to carbon waste when utilizing carbon-consuming but effective NAS methods. Conversely, employing carbon-saving yet sample-inefficient NAS methods may deteriorate search performance.

To address this issue, we propose a carbon-aware adaptive NAS search strategy that balances energy consumption during high-carbon and low-carbon periods. Our strategy decouples the two parts of a NAS search process—evaluation (energy-consuming) and sampling (energy-saving)—and handles them independently across different carbon periods. The basic idea involves leveraging the energy-efficient one-shot NAS [3, 15, 18, 29] to effectively estimate the accuracy of architectures in the sampling process during periods of high carbon intensity. Meanwhile, we will run the expensive evaluation part primarily during low-carbon periods. In the following sections, we will provide a detailed explanation of the carbon-aware NAS strategy.

3.2 Search Initialization

Similar to other optimization problems [7, 19, 21, 30], the first step in our proposed carbon-efficient NAS framework involves initializing the search process by randomly selecting several architectures, $\mathbf{a}$, from the search space, Ω, and evaluating their accuracy, $E(\mathbf{a})$,

carbon emissions, $C(\mathbf{a})$, and inference energy, $I(\mathbf{a})$. The resulting samples are then added to the observed samples set, $\mathcal{P}$.

Here, we define two types of methods for evaluating the accuracy of architectures. One is actual training, which trains the architecture a from scratch until convergence and evaluates it to obtain its true accuracy, $E(a)$. Another method is called *one-shot evaluation* [3, 15, 18], which leverages a trained *supernet* to estimate the accuracy of the architecture, denoted as $E^{'}(a)$. Note that obtaining $E^{'}(a)$ is energy-efficient; however, due to the co-adaption among operations [29], $E^{'}(a)$ is often not as accurate as $E(a)$. We train all the sampled architectures in the initialization stage to obtain their true accuracy for further search.

3.3 Energy-Efficient Architecture Sampling

To search for architectures with high inference accuracy and low inference energy, we formulate the search problem as a multi-objective optimization (MOO).

Primer. Mathematically, in multi-objective optimization we optimize M objectives $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})] \in \mathrm{r}^M$:

$$\begin{aligned} \min \quad & f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_M(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega, \end{aligned} \tag{1}$$

where $f_i(\boldsymbol{x})$ denotes the function of objective i. Modern MOO methods aim to find the problem's entire *Pareto frontier*, the set of solutions that are not *dominated* by any other feasible solutions. Here we define *dominance* $\boldsymbol{y} \prec_f \boldsymbol{x}$ as $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ for all functions f_i, and there exists at least one i s.t. $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$, $1 \leq i \leq M$. If the condition holds, a solution $\boldsymbol{x}$ is always better than solution $\boldsymbol{y}$.

Multi-objective search space partition. We leverage the recently proposed multi-objective optimizer called LaMOO [30] that learns to partition the search space from observed samples to focus on promising regions likely to contain the Pareto frontier. LaMOO is a general optimizer; we can extend it to NAS as follows.

We utilize LaMOO [30] to partition the search space, Ω, into several sub-search spaces. This partitioning will be based on the architectures and their true accuracy ($E(\mathbf{a})$) and inference energy ($I(\mathbf{a})$) as observed in the sample set, $\mathcal{P}$. Specifically, LaMOO recursively divides the search space into promising and non-promising regions. Each partitioned region can then be mapped to a node in a search tree. Using Monte-Carlo Tree Search (MCTS), LaMOO selects the most promising sub-space (i.e., tree node) for further exploration based on their UCB values [1]. This optimal sub-space is denoted as Ω_{best}.

Next, we will construct and train a supernet [3, 29], $\mathcal{S}_{best}$, for Ω_{best}. We then use a NAS search algorithm to identify new architectures that will be used to refine the search space. In this work, we employ the state-of-the-art multi-objective Bayesian optimization algorithm qNEHVI [8]. This algorithm will generate new architectures, denoted as $\boldsymbol{a_n}$, from Ω_{best}, and estimate their approximate accuracy, $E^{'}(\boldsymbol{a_n})$, using $\mathcal{S}_{best}$. At the same time, these architectures $\boldsymbol{a_n}$ are added to a ready-to-train set, $\mathcal{R}$, consisting of architecture candidates for further training.

Currently, to avoid unnecessary training and energy consumption, we define the maximum capacity of $\mathcal{R}$ as $Cap(\mathcal{R})$. When the capacity reaches, i.e., when there are more architectures to train than we have resources for, the sampling process blocks until spaces free up in $\mathcal{R}$. The accuracy of architectures, either estimated by $\mathcal{S}_{best}$ or obtained from training, will be fed back into the search engine as shown in Figure 1 to repeat the process described above.

As mentioned in §3.2, obtaining estimated accuracy through supernet is energy-efficient because these architectures can be evaluated without the time-consuming training. Therefore, during high carbon emission periods, CE-NAS will try to perform this process to save energy and produce as little carbon as possible, as shown in the middle left part of Figure 1.

3.4 Energy-Consuming Architecture Evaluation

If we perform the entire NAS only using the process described in §3.3, CE-NAS essentially is performing one-shot NAS within the sub-space $\mathcal{S}_{best}$. However, it is possible to improve LaMOO's space partition with more observed samples, as Zhao et al. showed [30]. This section describes the process to evolve $\mathcal{S}_{best}$ during low carbon emission periods.

At the high level, we will pick new architectures to train to convergence and use them to refine the search space partition. That is, the architecture $\boldsymbol{a}$, with its true accuracy, $E(\boldsymbol{a})$, will be added to the observed sample set $\mathcal{P}$ to help identify a more advantageous sub-space, Ω_{best}, for the architecture sampling process. In this work, we sort the architectures in the ready-to-train set $\mathcal{R}$ by their *dominance number*. The dominance number $o(\boldsymbol{a})$ of an architecture $\boldsymbol{a}$ is defined as the number of samples that dominate $\boldsymbol{a}$ in search space Ω:

$$o(\boldsymbol{a}) := \sum_{\boldsymbol{a_i} \in \Omega} \mathbb{I}[\boldsymbol{a_i} \prec_f \boldsymbol{a},\ \boldsymbol{a} \neq \boldsymbol{a_i}], \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function. With the decreasing of the $o(\boldsymbol{a})$, $\boldsymbol{a}$ would be approaching the Pareto frontier; $o(\boldsymbol{a}) = 0$ when the sample architecture $\boldsymbol{a}$ locates in the Pareto frontier. The use of dominance number allows us to rank an architecture by considering both the estimated accuracy $E^{'}(\boldsymbol{a})$ and its inference energy cost $I(\boldsymbol{a})$ at the same time. CE-NAS will first train the architectures with lower dominance number values when GPU resources are available. Once an architecture is trained, it is removed from $\mathcal{R}$.

This process is depicted in the middle right part of Figure 1. Note that this process includes actual time-consuming DL training, which is energy-intensive. Hence, CE-NAS will try to prioritize this process during periods of low carbon intensity.

3.5 GPU Allocation Strategy

The carbon impact of the above two processes in a NAS search is materialized through the use of GPU resources. A key decision CE-NAS needs to make is *how* to allocate GPUs among these two interdependent processes. Assigning too many GPUs to the architecture sampling could impact the search efficiency, i.e., the searched architectures are far from the Pareto frontier; assigning too many GPUs to the architecture evaluation could significantly increase energy consumption. CE-NAS must consider these trade-offs under varying carbon intensity and re-evaluate the GPU allocation strategy.

Below we describe a heuristic strategy that automatically allocates GPU resources between the sampling and evaluation processes given the carbon emissions C_t at time t. This allocation is based on the energy characteristics of the processes: architecture

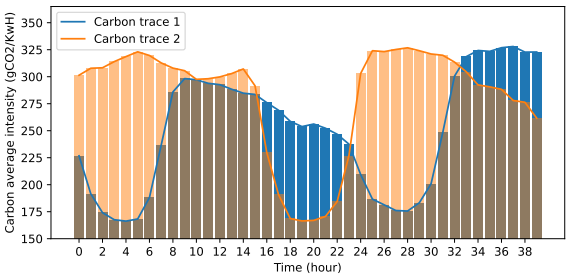
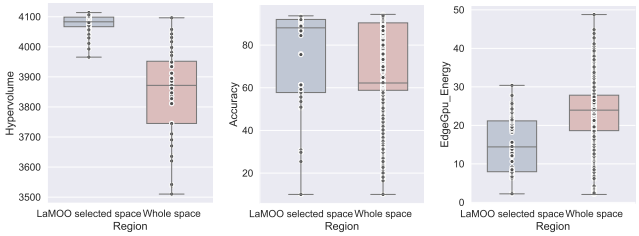
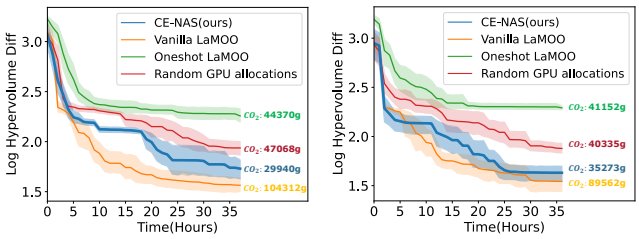


Figure 2: Carbon traces from electricityMap. Trace 1 is based on the US-CAL-CISO data from 2021, specifically covering the period from 0:00, January 1, 2021, to 16:00, January 2, 2021. Trace 2 is also based on the US-CAL-CISO data from 2021, covering the period from 17:00, January 2, 2021, to 9:00, January 4, 2021.



(a) Hypervolume (b) Accuracy (c) Inference energy

Figure 3: Comparisons of architecture qualities between LaMOO-selected region and the entire search space of HW-Nasbench. We ran LaMOO 10 times. For each run, we randomly sampled 50 architectures from the LaMOO-selected space and the whole search space.



(a) With carbon trace 1 (b) With carbon trace 2

Figure 4: Search progress over time. CE-NAS has the lowest relative carbon emission while achieving the second best $HV_{\text{log\_diff}}$.

sampling is often energy-efficient because it does not involve actual training of architectures, while architecture evaluation is often energy-consuming because it does. We assume that the maximum and minimum carbon intensities C_{max} and C_{min} for a future time window are known. G_t denotes the total number of available GPUs. λ_e and λ_s represent the ratio of GPU numbers allocated to the evaluation and sampling processes at a given moment, and $\lambda_e + \lambda_s = 1$. We calculate λ_s as $\frac{C_{cur}-C_{min}}{C_{max}-C_{min}}$, where C_{cur} is the current carbon intensity. The GPU allocations for the sampling and evaluation processes are, therefore, $G_t * \lambda_s$ and $G_t * \lambda_e$. This simple heuristic allocation allows CE-NAS to prioritize more energy-efficient sampling tasks during periods of higher carbon intensity, whereas, during low-carbon periods, CE-NAS will allocate more resources for energy-intensive evaluation tasks.

4 PRELIMINARY RESULTS

We prototype the CE-NAS framework described in §3. This section presents a preliminary analysis of CE-NAS for its carbon and search efficiency based on trace-driven simulations. Specifically, we evaluate LaMOO's performance in partitioning the search space for NAS on HW-NASBench [13].

HW-NASBench was selected due to its inclusion of information on our two search targets: inference energy and accuracy. To assess the search performance and carbon cost of CE-NAS, we have CE-NAS search for optimal architectures on HW-NASBench and compare the searched results to three different NAS search methods. CE-NAS delivers the most effective search results within the same carbon budget.

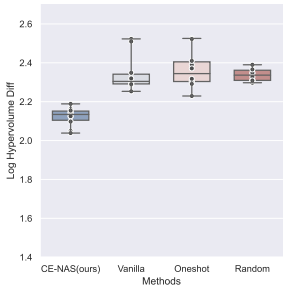
4.1 Setup

We conduct our experiments using CE-NAS and other baselines based on the two carbon traces depicted in Figure 2. We initiate the process with ten samples in the set $\mathcal{P}$ and set the maximum capacity of $\mathcal{R}$ to be 300. Each method is simulated ten times for consistency, and all search processes in the simulation are executed on an Nvidia GeForce RTX 3090.

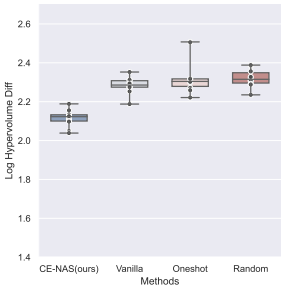
Carbon Traces. We used two carbon traces obtained from ElectricityMap [17], a third-party carbon information service. Both carbon traces span 40 hours and consist of the per-hour average carbon intensity. We chose these two traces because they exhibit varying carbon intensity, as shown in Figure 2, which allowed us to evaluate both the search over time performance and CE-NAS's adaptiveness to carbon intensity.

NAS Dataset. A number of popular open-source NAS datasets, such as NasBench101 [27], NasBench201 [10], and NasBench301 [28] exist. However, none contain information on architecture inference energy, one of our search objectives. We chose the new NAS dataset called *HW-NASBench* [13] due to its inclusion of information on our two search targets: inference energy and accuracy. Specifically, HW-NASBench contains inference performance of all networks in the NasBench201's search space [10] on six hardware devices, including commercial edge devices. In short, we use a combination of architecture information, including inference accuracy, training time, evaluation time, and energy cost in the edge GPU obtained from HW-NASBench and NasBench201 [10].

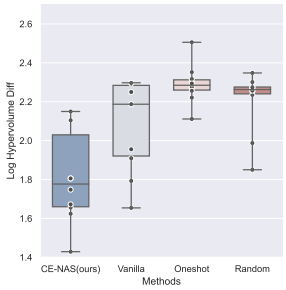
Metrics. We use two main metrics to evaluate the carbon and search efficiency of CE-NAS. First, we use *relative carbon emission* to quantify the amount of CO2 each NAS method is responsible for. The relative carbon emission is calculated by summing the average carbon intensity (in gCO2/KwH) over the search process. We assume that all NAS methods use the same type of GPU whose power



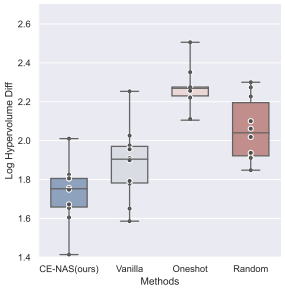
(a) CO_2 cost: 5000g



(b) CO_2 cost: 10000g

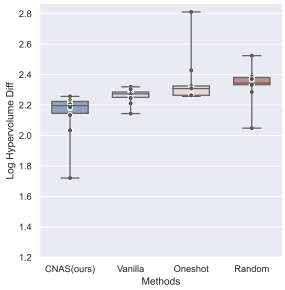


(c) CO_2 cost: 20000g

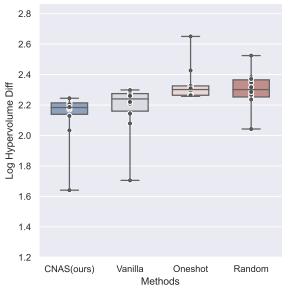


(d) CO_2 cost: 30000g

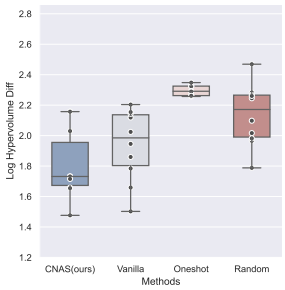
Figure 5: Search efficiency under carbon emission constraints. These results are obtained using carbon trace 1, and we ran each method ten times.



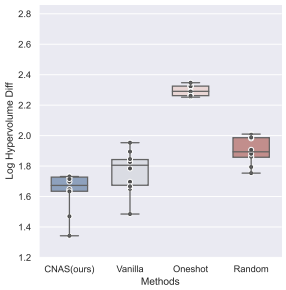
(a) CO_2 cost: 5000g



(b) CO_2 cost: 10000g



(c) CO_2 cost: 20000g



(d) CO_2 cost: 30000g

Figure 6: Search efficiency under carbon emission constraints. These results are obtained using carbon trace 2, and we ran each method ten times.

consumption remains the same throughout the search process. Second, we use the metric *hypervolume* (HV) to measure the "goodness" of searched samples. HV is a commonly used multi-objective optimization quality indicator [7, 8, 30] that considers all dimensions of the search objective. Given a reference point $R \in \mathrm{r}^M$, the HV of a finite approximate Pareto set $\mathcal{P}$ is the M-dimensional Lebesgue measure λ_M of the space dominated by $\mathcal{P}$ and bounded from below by R. That is, $HV(\mathcal{P}, R) = \lambda_M(\cup_{i=1}^{|\mathcal{P}|}[R, y_i])$, where $[R, y_i]$ denotes the hyper-rectangle bounded by the reference point R and y_i. A higher hypervolume denotes better multi-objective results.

Baselines. We chose three types of baselines according to different GPU allocation strategies and NAS evaluation algorithms. During the search process, all search methods employ the state-of-the-art multi-objective optimizer, LaMOO [30]. Specifically, *one-shot LaMOO* is a method that utilizes one-shot evaluations throughout the search process. The *vanilla LaMOO* relies on actual training for architecture evaluation throughout the search. The *random GPU allocations* is a strawman strategy that randomly allocates GPUs between the energy-efficient sampling stage and the more energy-consuming evaluation stage without considering the carbon intensity.

4.2 Effectiveness of LaMOO for NAS

We conducted ten runs of LaMOO (i.e., search space split) with a random search on the HW-NASBench dataset [13]. In addition, we performed random sampling for both the LaMOO-selected region and the entire search space, conducting 50 trials for each. The distribution of accuracy and edge GPU energy consumption of the architectures in both the LaMOO selected region and the entire search space can be seen in Figure 3.

Specifically, our results show that the architectures in the region selected by LaMOO have higher average accuracy and lower average edge GPU energy consumption compared to those in the entire search space. On average, the accuracy of the architectures in the LaMOO selected region is 72.12, while the accuracy in the entire search space is 68.28. The average edge GPU energy for the LaMOO selected region is 16.59 mJ, as opposed to 22.84 mJ for the entire space.

Furthermore, as illustrated in Figure 3(a), we observe that searching within the LaMOO-selected region yielded a tighter distribution, and the median hypervolume demonstrated an improvement compared to searching across the entire search space. These results suggest the efficacy of using LaMOO to partition the search space for NAS.

4.3 Carbon and Search Efficiency

In this section, we evaluate the search performance and carbon costs of our `CE-NAS` framework, comparing it to three other baselines on the HW-NASBench dataset [13]. We use the *log hypervolume* difference, the same as in [7, 8, 30], as our evaluation criterion for HW-NASBench, since the hypervolume difference may be minimal over the search process. Therefore, using log hypervolume allows us to visualize the sample efficiency of different search methods. We

define $HV_{\text{log\_diff}} := \log(HV_{\max} - HV_{\text{cur}})$ where $HV_{\max}$ represents the maximum hypervolume calculated from all points in the search space, and HV_{cur} denotes the hypervolume of the current samples, which are obtained by the algorithm within a specified budget. The $HV_{\max}$ in this problem is 4150.7236. For our simulation, we use the training and evaluation time costs for the architectures derived from NasBench201 [10], and inference energy costs measured on the NVIDIA Edge GPU Jetson TX2 from HW-NASBench [13]. We ran the simulation 10 times with each method.

As depicted in Figure 4, as the search time progresses, vanilla LaMOO demonstrates the highest performance in terms of $HV_{\text{log\_diff}}$. Vanilla LaMOO's superior performance can be attributed to its approach of *training all sampled architectures* to obtain their true accuracy, which effectively steers the search direction. However, when considering the relative carbon emission, vanilla LaMOO consumes 2.22X-3.48X carbon compared to other approaches. This is expected because vanilla LaMOO is an energy-consuming approach and is not designed to be aware of carbon emissions associated with joules.

We show that CE-NAS's search efficiency is only marginally lower than that of vanilla LaMOO while having the least relative carbon emission under both carbon traces. Note that we are plotting the $HV_{\text{log\_diff}}$ in the Y-axis of Figure 4; the actual HV values achieved by CE-NAS and Vanilla LaMOO are about 4100 and 4117, differing only by 0.034%, even though the two lines look far apart. This result suggests that only relying on energy-efficient approaches (e.g., one-shot LaMOO in this case) is insufficient to achieve carbon efficiency. For both traces, one-shot LaMOO has 1.17X-1.48X carbon compared to CE-NAS.

Moreover, we observe that CE-NAS's carbon efficiency is correlated to the time-varying carbon intensity. When the coefficient of variation of carbon intensity is higher, CE-NAS has more opportunity to explore the GPU allocation trade-offs between energy-efficient sampling and energy-consuming evaluation without impacting search quality. The relative carbon emission difference between CE-NAS and *random GPU allocations* represents how well CE-NAS makes such trade-offs. Currently, we are using a heuristic approach, and it is possible to devise more sophisticated strategies to further reduce relative carbon emissions. For example, if the strategy could determine the GPU resources based on the queued architectures and the current carbon intensity, it can better shift the workload to periods of low carbon emission.

Finally, Figure 5 and 6 compare CE-NAS performance with baselines under different carbon budgets. We show that CE-NAS outperforms all baselines in terms of search efficiency. This is because when there is a carbon budget, energy-consuming approaches (e.g., vanilla LaMOO) would exhaust the budget and end the search earlier, as opposed to operating with an unlimited carbon budget. This result suggests CE-NAS's ability to dynamically adjust the search process based on carbon budgets while still producing reasonable search efficiency.

5 CONCLUSION AND FUTURE DIRECTIONS

In this work, we described the design of a carbon-efficient NAS framework CE-NAS by leveraging the temporal variations in carbon intensity. To search for energy-efficient architectures, CE-NAS integrates a state-of-the-art multi-objective optimizer, LaMOO [30], with the one-shot and vanilla NAS algorithms. These two NAS evaluation strategies have different energy requirements, which CE-NAS leverages to schedule when to use each based on average carbon intensity. Our trace-driven simulations show that CE-NAS is a promising approach for reducing relative carbon emission while maintaining search efficiency.

Based on our investigation, we believe there are many fruitful directions in the context of CE-NAS. For example, one can train an agent, e.g., use deep reinforcement learning, to automatically output different GPU allocation strategies based on historical carbon traces. This can replace our current heuristic GPU allocation strategy and will likely lead to better carbon and search efficiency. Another direction is to develop models that are capable of accurately predicting carbon intensity, similar to a recent work [16]. With such predictive models, CE-NAS can better schedule the NAS tasks to a dynamic set of GPUs that can span across geographic locations without adversely impacting the total search time.

ACKNOWLEDGMENTS

This work was supported in part by NSF Grants #2105564 and #2236987, and a VMWare grant. We also thank electricityMap for its carbon intensity dataset.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.
[2] Noman Bashir, Tian Guo, Mohammad Hajiesmaili, David Irwin, Prashant Shenoy, Ramesh Sitaraman, Abel Souza, and Adam Wierman. 2021. Enabling sustainable clouds: The case for virtualizing the energy system. In *Proceedings of the ACM Symposium on Cloud Computing*. 350–358.
[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and Simplifying One-Shot Architecture Search. In *Proceedings of the 35th International Conference on Machine Learning*.
[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*. https://arxiv.org/pdf/1908.09791.pdf
[5] Han Cai, Ligeng Zhu, and Song Han. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*. https://arxiv.org/pdf/1812.00332.pdf
[6] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive DARTS: Bridging the Optimization Gap for NAS in the Wild. *CoRR* abs/1912.10952 (2019). arXiv:1912.10952 http://arxiv.org/abs/1912.10952
[7] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. 2020. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. *arXiv preprint arXiv:2006.05078* (2020).
[8] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. 2021. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems* 34 (2021), 2187–2200.
[9] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. 2015. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*.
[10] Xuanyi Dong and Yi Yang. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=HJxyZkBKDr
[11] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang, and Quoc V. Le. 2019. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. *CoRR* abs/1904.07392 (2019). arXiv:1904.07392 http://arxiv.org/abs/1904.07392
[12] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*. PMLR, 528–536.
[13] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. 2021. {HW}-{NAS}-Bench: Hardware-Aware Neural Architecture Search Benchmark. In *International Conference on Learning Representations*. https://openreview.net/forum?id=

_0kaDkv3dVf

[14] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive Neural Architecture Search. In *European Conference on Computer Vision(ECCV)*.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations(ICLR)*.

[16] Diptyaroop Maji, Prashant Shenoy, and Ramesh K. Sitaraman. 2022. CarbonCast: Multi-Day Forecasting of Grid Carbon Intensity. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (Boston, Massachusetts) *(BuildSys '22)*. Association for Computing Machinery, New York, NY, USA, 198–207. https://doi.org/10.1145/3563357.3564079

[17] Electricity Map. [n. d.]. Electricity Map. https://app.electricitymaps.com/map

[18] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *International Conference on Machine Learning(ICML)*.

[19] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized Evolution for Image Classifier Architecture Search. In *Association for the Advancement of Artificial Intelligence(AAAI)*.

[20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[21] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. 2020. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems* 33 (2020), 19511–19522.

[22] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. 2019. Sample-Efficient Neural Architecture Search by Learning Action Space. *CoRR* abs/1906.06832 (2019). arXiv:1906.06832 http://arxiv.org/abs/1906.06832

[23] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. 2019. AlphaX: eXploring Neural Architectures with Deep Neural Networks and Monte Carlo Tree Search. *CoRR* abs/1903.11059 (2019). arXiv:1903.11059 http://arxiv.org/abs/1903.11059

[24] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. 2020. NAS-FCOS: Fast Neural Architecture Search for Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. 2021. KNAS: green neural architecture search. In *International Conference on Machine Learning*. PMLR, 11613–11625.

[26] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2020. {PC}-{DARTS}: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJlS634tPr

[27] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *Proceedings of the 36th International Conference on Machine Learning*.

[28] Arber Zela, Julien Niklas Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. 2022. Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=OnpFa95RVqs

[29] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. 2021. Few-Shot Neural Architecture Search. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 12707–12718. http://proceedings.mlr.press/v139/zhao21d.html

[30] Yiyang Zhao, Linnan Wang, Kevin Yang, Tianjun Zhang, Tian Guo, and Yuandong Tian. 2022. Multi-objective Optimization by Learning Space Partition. In *International Conference on Learning Representations*. https://openreview.net/forum?id=FlwzVjfMryn

[31] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.