Energy Consumption-Aware Tabular Benchmarks for Neural Architecture Search

Bakhtiarifard, Pedram; Igel, Christian; Selvan, Raghavendra

Publication date:
2022

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
ENERGY CONSUMPTION-AWARE TABULAR BENCHMARKS FOR NEURAL ARCHITECTURE SEARCH

Pedram Bakhtiarifard, Christian Igel & Raghavendra Selvan
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
{pba,igel,raghav}@di.ku.dk

ABSTRACT

The demand for large-scale computational resources for Neural Architecture Search (NAS) has been lessened by tabular benchmarks for NAS. Evaluating NAS strategies is now possible on extensive search spaces and at a moderate computational cost. But so far, NAS has mainly focused on maximising performance on some hold-out validation/test set. However, energy consumption is a partially conflicting objective that should not be neglected. We hypothesise that constraining NAS to include the energy consumption of training the models could reveal a subspace of undiscovered architectures that are more computationally efficient with a smaller carbon footprint. To support the hypothesis, an existing tabular benchmark for NAS is augmented with the energy consumption of each architecture. We then perform multi-objective optimisation that includes energy consumption as an additional objective. We demonstrate the usefulness of multi-objective NAS for uncovering the trade-off between performance and energy consumption as well as for finding more energy-efficient architectures. The updated tabular benchmark, EC-NAS-Bench, is open-sourced to encourage the further exploration of energy consumption-aware NAS.

1 INTRODUCTION

The design of neural architectures is a complex task. While general guidelines for producing suitable neural architectures have been proposed, neural architecture design still requires expert domain knowledge, experience, and not least substantial effort (Philipp, 2021; Zoph & Le, 2016; Ren et al., 2020). This led to an upsurge in research on automated exploration and design of neural architectures cast as an optimisation problem – neural architecture search (NAS) (Baker et al., 2016; Zoph & Le, 2016; Real et al., 2017).

NAS strategies explore neural architectures in a predefined search space relying on model training and evaluation to determine the model’s fitness (i.e., validation/test set score) to adjust the search strategy and extract the best performing architecture (Ren et al., 2020). NAS strategies have shown great promise in discovering novel architecture designs yielding state-of-the-art model performance (Liu et al., 2017; 2018; Lin et al., 2021; Baker et al., 2017). However, it can be prohibitively expensive to perform NAS (Tan & Le, 2019b) due to the demand for large-scale computational resources and the associated carbon footprint of NAS (Schwartz et al., 2019; Anthony et al., 2020).

The introduction of tabular benchmarks for NAS significantly lessened the computational challenges mentioned above by facilitating the evaluation of NAS strategies on a limited search space of architectures (Klein & Hutter, 2019; Dong & Yang, 2020). Predictive models and zero- and one-shot models (Wen et al., 2019; Dong & Yang, 2020) have reduced time-consuming model training and thereby increased the efficiency of NAS strategies. Most recently, surrogate NAS benchmarks (Zela et al., 2022) have been proposed for arbitrary expansion of architecture search spaces for NAS.

https://github.com/PedramBakh/EC-NAS-Bench
Notwithstanding the aforementioned major contributions to the advancement of NAS research, the prime objective of NAS has been maximising a performance objective on some hold-out test/validation test. NAS strategies can be evaluated effectively, yet the search strategies do not intentionally aim to find computationally efficient architectures. That is, the NAS may efficiently determine model performance at a moderate computational cost, but energy efficiency is generally not an objective of NAS.

We hypothesise that adding the energy consumption of training models as a NAS objective could reveal a sub-space of computationally efficient models that also have a smaller carbon footprint. In order to find efficient architectures without sacrificing cardinal performance requirements, we propose the use of NAS strategies that will optimise for multiple objectives.

Our main contributions.

1. We provide an energy consumption-aware tabular benchmark for NAS based on NAS-Bench-101 (Ying et al., 2019). For each architecture, we added its training energy consumption, power consumption and carbon footprint. We hope that the new data set will foster the development of environmentally friendly deep learning systems.
2. To exemplify the use of the new benchmark, we devise a multi-objective optimisation algorithm for NAS and apply it to optimise generalisation accuracy as well as energy consumption. The algorithm is simple, but derived from canonical principles of derivative-free multi-criteria optimisation, such as hypervolume maximisation.
3. We demonstrate the usefulness of multi-objective architecture exploration for revealing the trade-off between performance and energy efficiency and for finding efficient architectures obeying accuracy constraints.

2 E NERGY CONSUMPTION-AWARE BENCHMARKS - EC-NAS-Bench

Our energy consumption-aware tabular benchmark EC-NAS-Bench is based on Nas-Bench-101 (Ying et al., 2019). We closely follow their specification of architectures; however, the search space of architectures that are considered, the evaluation approach and the metrics provided for each architecture is different. This section will briefly present EC-NAS-Bench and its differences to NAS-Bench-101.

2.1 A RCHITECTURE DESIGN

Network Topology. All architectures considered are convolutional neural networks (CNNs) designed for the task of image classification on CIFAR-10 (Krizhevsky, 2009). Each neural network comprises a convolutional stem layer followed by three repeats of three stacked cells and a downsampling layer. Finally, a global pooling layer and a dense softmax layer are used. The space of architectures, $X$, is limited to the topological space of cells, where each cell is a configurable feedforward network.

Cell Encoding. The individual cells are represented as directed acyclic graphs (DAGs). Each DAG, $G(V, M)$, has $N = |V|$ vertices (or nodes) and edges described in the binary adjacency matrix $M \in \{0, 1\}^{N \times N}$. The set of operations (labels) that each node can realise is given by $L' = \{\text{input, output}\} \cup L$, where $L = \{3x3\text{conv}, 1x1\text{conv}, 3x3\text{maxpool}\}$. Two of the $N$ nodes are always fixed as input and output to the network. The remaining $N - 2$ nodes can take up one of the labels in $L$. The connections between nodes of the DAG are encoded in the upper-triangular adjacency matrix with no self-connections (zero main diagonal entries). For a given architecture, $A$, every entry $a_{i,j} \in M_A$ denotes an edge, from node $i$ to node $j$ with operations $i, j \in L$ and its labelled adjacency matrix, $L_{A} \in M_{A} \times L'$.
Search space. The number of DAGs grows exponentially with $N$ and $L$ (Ying et al., 2019). We restrict the search space in EC-NAS-Bench by imposing $|V| \leq 5$ and $|A \neq 0| \leq 9$, referred to as the 5V space. The search space with $|V| \leq 4$ called 4V space is also considered. In contrast, NAS-Bench-101 considers the search space for $|V| \leq 7$. With these imposed restrictions on the search space of EC-NAS-Bench, 91 and 2532 unique architectures are identified from the 4V and 5V space, respectively.

2.2 ENERGY CONSUMPTION-AWARENESS

Resource-constrained NAS for obtaining efficient architectures has been explored mainly by optimising the total number of floating point operations (FPOs) (Tan & Le, 2019a). Optimising for FPOs, however, might not be entirely indicative of the efficiency of models (Henderson et al., 2020). It has been reported that models with fewer FPOs have bottleneck operations that can consume the bulk of the training time (Howard et al., 2017), and some models with high FPOs have lower inference time (Jeon & Kim, 2018). Energy consumption optimised hyperparameter selection outside of NAS settings for large language models has been recently investigated in Puvis de Chavannes et al. (2021).

The energy consumption during the training of a model encapsulates facets of architecture efficiency that are not entirely taken into consideration when using standard resource constraints such as FPOs, computational time and the number of parameters. Energy consumption accounts for both hardware and software variations in the experimental set-ups. To foster a new direction for NAS to find more efficient architectures, we use energy consumption as the additional objective along with standard performance measures.

2.3 QUANTIFYING ENERGY CONSUMPTION

About 75% of the total energy costs during training a neural network are incurred by hardware accelerators such as graphics processing units (GPUs) or tensor processing units (TPUs) (Dodge et al., 2022). The remaining energy consumption is mainly due to the central processing units (CPUs) and dynamic random access memory (DRAM). Additional energy consumed by the supporting infrastructure, such as cooling- and power systems and dissipation, is usually accounted for by the power usage effectiveness (PUE), which is an overhead factor. Several open-source tools have been published in the past couple of years, such as experiment-impact-tracker (Henderson et al., 2020), Carbontracker (Anthony et al., 2020) and CodeCarbon (Schmidt et al., 2021) provide convenient ways to track and log the energy consumption of neural networks by taking these factors into consideration.

In EC-NAS-Bench, the energy consumption of training and evaluating the neural architectures is estimated by modifying the tool Carbontracker (Anthony et al., 2020). Our version of the tool monitors the GPUs, CPUs and DRAM and estimates the total energy costs, $E$ (kWh), aggregate carbon footprint (kgCO$_2$eq) based on the instantaneous carbon intensity of the regions and the total computation time, $T(s)$. The complete set of metrics that are measured and reported in EC-NAS-Bench are listed in Table 1.

2.4 ARCHITECTURE PERFORMANCE AND EFFICIENCY

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Unit of measurement</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parameters</td>
<td>Million (M)</td>
<td>$</td>
</tr>
<tr>
<td>Test/Train/Eval. time</td>
<td>Seconds (s)</td>
<td>T(s)</td>
</tr>
<tr>
<td>Test/Train/Val. Acc.</td>
<td>$R \in [0; 1]$</td>
<td>$P_r$</td>
</tr>
<tr>
<td>Energy consumption</td>
<td>Kilowatt-hour (kWh)</td>
<td>$E(kWh)$</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Joule (J), Watt (W)</td>
<td>$E(\Omega)$</td>
</tr>
<tr>
<td>Carbon footprint</td>
<td>kgCO$_2$eq</td>
<td>$-$</td>
</tr>
<tr>
<td>Carbon intensity</td>
<td>g/kWh</td>
<td>$-$</td>
</tr>
</tbody>
</table>

Table 1: Metrics reported in EC-NAS-Bench.

Training Pipeline. Architectures from the 4V and 5V space are trained on CIFAR-10 (Krizhevsky, 2009) using 40k samples and evaluated on 10k validation and test samples (60k total). Each model is trained on an in-house Slurm cluster on a single NVIDIA Quadro RTX 6000 GPU with 24 GB memory and two Intel CPUs. The training strategy, or hyper-parameter setting, is similar to that of NAS-Bench-101 (Klein & Hutter, 2019). Predicting the energy consumption of longer model runs from a few training epochs has been shown to be robust when performed on the same hardware (Anthony et al., 2020). To refrain from re-
training and re-evaluating all the models in NAS-Bench-101, we train each model for only 4 epochs and then obtain surrogate time and energy measurements by linear scaling. We then tabulate these measurements along with the corresponding mean performance metrics for each model from NAS-Bench-101 and obtain metrics for training and evaluating each model for 12, 36 and 108 epochs.

**Metrics.** We report the operations, no. parameters, and performance metrics in EC-NAS-Bench, as in NAS-Bench-101, and additionally, we include efficiency measures in terms of energy consumption and the carbon footprint for training each model. The primary focus for efficiency metrics is to quantify the resource costs specific to model training; however, we also report the total resource costs, which include computational overhead, e.g., data movements. For completeness, we also provide carbon intensity measures at training time, timestamp, and average energy consumption of computing resources. We have made the metrics of each architecture readily accessible to encourage the development of NAS strategies for exploring efficient architectures. The metrics reported relevant to this work can be seen in Table 1.

### 3 NAS STRATEGIES WITH EC-NAS-Bench

Given a tabular benchmark which can be used to query for model training energy consumption in addition to other standard metrics such as in EC-NAS-Bench, NAS strategies can be used to search for energy-efficient architectures. We next present multi-objective optimisation as a suitable strategy to uncover the trade-off between performance and efficiency, which supports an energy-aware architecture choice.

#### 3.1 MULTI-OBJECTIVE OPTIMISATION

Multi-objective optimisation (MOO), also referred to as multi-criteria or vector optimisation, simultaneously optimises several, potentially conflicting objectives. The goal of MOO is to find or to approximate the set of Pareto-optimal solutions, where a solution is Pareto-optimal if it cannot be improved in one objective without getting worse in another. Formally, let the MOO problem be described by \( f: \mathbb{X} \rightarrow \mathbb{R}^m, \ f(x) \rightarrow (f_1(x), \ldots, f_m(x)). \) Here \( \mathbb{X} \) denotes the search space of the optimization problem and \( m \) refers to the number of objectives. We assume w.l.o.g. that all objectives are to be minimized. For two points \( x, x' \in \mathbb{X} \) we say that \( x' \) dominates \( x \) and write \( x' < x \) if \( \forall i \in \{1, \ldots, m\} \colon f_i(x') \leq f_i(x) \land \exists j \in \{1, \ldots, m\} : f_j(x') < f_j(x). \) For \( X', X'' \subseteq \mathbb{X} \) we say that \( X' \) dominates \( X'' \) and write \( X' < X'' \) if \( \forall x'' \in X'' : \exists x' \in X' : x' < x''. \) The subset of non-dominated solutions in a set \( X' \subseteq \mathbb{X} \) is given by \( \text{ndom}(X') = \{ x \mid x \in X' \land \forall x'' \in X' \setminus \{x\} : x'' < x \}. \) The Pareto front of a set \( X' \subseteq \mathbb{X} \) defined as \( \mathcal{F}(X') = \{ f(x) \mid x \in \text{ndom}(X') \} \) and, thus, the goal of MOO can be formalised as approximating \( \mathcal{F}(\mathbb{X}). \)

In iterative MOO, the strategy is to step-wise improve a set of candidate solutions towards a sufficiently good approximation of \( \mathcal{F}(\mathbb{X}). \) For the design of a MOO algorithm, it is important to have a way to rank two sets \( X' \) and \( X'' \) w.r.t. the overall MOO goal even if neither \( X' < X'' \) nor \( X'' < X'. \) This ranking can be done by the hypervolume measure. The hypervolume measure or \( \mathcal{S}\)-metric (see Zitzler & Thiele, 1999) of a set \( X' \subseteq \mathbb{X} \) is the volume of the union of regions in \( \mathbb{R}^m \) that are dominated by \( X' \) and bounded by some appropriately chosen reference point \( r \in \mathbb{R}^m: \)

\[
\mathcal{S}_r(X') := \Lambda \left( \bigcup_{x \in X'} [f_1(x), r_1] \times \cdots \times [f_m(x), r_m] \right),
\]

where \( \Lambda(\cdot) \) is the Lebesgue measure. The hypervolume is, up to weighting objectives, the only strictly Pareto compliant measure (Zitzler et al., 2003) in the sense that given two sets \( X' \) and \( X'' \) we have \( \mathcal{S}(X') > \mathcal{S}(X'') \) if \( X' \) dominates \( X''. \) As stated by Brüggemann et al. (2013), the worst-case approximation factor of a Pareto front \( \mathcal{F}(X') \) obtained from any hypervolume-optimal set \( X' \) with size \( |X'| = \mu \) is asymptotically equal to the best worst-case approximation factor achievable by any set of size \( \mu \), namely \( \Theta(1/\mu) \) for additive approximation and \( 1+\Theta(1/\mu) \) for relative approximation (Brüggemann & Friedrich, 2013). Now we define the contributing hypervolume of an individual \( x \in X' \) as

\[
\Delta_r(x, X') := \mathcal{S}_r(X') - \mathcal{S}_r(X' \setminus \{x\}).
\]

The value \( \Delta(x, X') \) quantifies how much a candidate solution \( x \) contributed to the total hypervolume of \( X' \) and can be regarded as a measure of the relevance of the point. Therefore, the contributing
Algorithm 1 Multi-Objective optimisation NAS strategy

**Input:** objective $f = (f_1, \ldots, f_m)$, maximum number of iterations $n$

**Output:** set of non-dominated solutions $P$

1: Initialize $P \subset X$ (e.g., randomly) \> Initial random architectures
2: $P \leftarrow \text{ndom}(P)$ \> Discard dominated solutions
3: for $i \leftarrow 1$ to $n$ do \> Loop over iterations
4: $O \leftarrow \text{LinearRankSample}(P, \lambda)$ \> Get $\lambda$ points from $P$
5: $O \leftarrow \text{Perturb}(O)$ \> Change the architectures
6: Compute $f(x)$ for all $x \in O$ \> Evaluate architectures
7: $P \leftarrow \text{ndom}(P \cup O)$ \> Discard dominated points
8: end for
9: return $P$

Procedure 2 Perturb($O$)

**Input:** set of architectures $O$, variation probabilities for edges and nodes $p_{\text{edge}}$ and $p_{\text{node}}$

**Output:** set of modified architecture $O^*$

1: for all $M_A \in O$ do \> Loop over matrices
2: repeat
3: for all $\alpha_{i,j} \in M_A$ do \> Loop over entries
4: With probability $p_{\text{edge}}$ flip $\alpha_{i,j}$
5: end for
6: for all $l \in L_A$ do \> Loop over labels
7: With probability $p_{\text{node}}$ change the label of $l$
8: end for
9: until $M_A$ has changed
10: end for
11: return $O^*$

Hypervolume is a popular criterion in MOO algorithms (e.g. Beume et al., 2007; Igel et al., 2007; Bader & Zitzler, 2011; Krause et al., 2016). If we iteratively optimize some solution set $P$, then points $x$ with low $\Delta(x, P)$ are candidates in an already crowded region of the current Pareto front $F(P)$, while points with high $\Delta(x, P)$ mark areas that are promising to explore further.

In this study, we used a simple MOO algorithm based on hypervolume maximisation outlined in Algorithm 1 inspired by Krause et al. (2016). The algorithm iteratively updates a set $P$ of candidate solutions, starting from a set of random network architectures. Dominated solutions are removed from $P$. Then $\lambda$ new architectures are generated by first selecting $\lambda$ architectures from $P$ and then modifying these architectures according to the perturbation described in Procedure 2. The $\lambda$ new architectures are added to $P$ and the next iteration starts. In Procedure 2, the probability $p_{\text{edge}}$ for changing (i.e., either adding or removing) an edge is chosen such that in expectation, two edges are changed, and the probability $p_{\text{node}}$ for changing a node is set such that in expectation every second perturbation changes the label of a node.

The selection of the $\lambda > m$ architectures from the current solution set is described in Procedure 3. We always select the extreme points in $P$ that minimize a single objective (thus, the precise choice of the reference point $r$ is of lesser importance). The other $m - \lambda$ points are randomly chosen preferring points with higher contributing hypervolume. The points in $P$ are ranked according to their hypervolume contribution. The probability of being selected depends linearly on the rank. We use linear ranking selection (Baker, 1985; Greffenstette & Baker, 1989), where the parameter controlling the slope is set to $\eta^+ = 2$. Always selecting the extreme points and focusing on points with large contributing hypervolume leads to a wide spread of non-dominated solutions.

3.2 Evaluation of NAS strategies

Experimental Setup. We conduct experiments on EC-NAS-Bench by adapting the presented MOO-algorithm to perform both single-objective optimisation (SOO) and MOO. In the former, we will naturally find only one solution when optimising a single objective. In contrast, when optimising multiple, diverse objectives, we will find the empirical Pareto-front in the latter. We
Procedure 3 LinearRankSample($P, \lambda$)

**Input:** set $P \subset X$ of candidate solutions, number $\lambda$ of elements to be selected; reference point $r \in \mathbb{R}^m$, parameter controlling the preference for better ranked points $\eta^+ \in [1, 2]$

**Output:** $O \subset P$, $|O| = \lambda$

1: $O = \emptyset$
2: for $i \leftarrow 1$ to $m$ do
3: $O \leftarrow O \cup \arg\min_{x \in P} f_i(x)$ ▷ Always add extremes
4: end for
5: Compute $\Delta_r(x,P)$ for all $x \in P$ ▷ Compute contributing hypervolume
6: Sort $P$ according to $\Delta(x,P)$
7: Define discrete probability distribution $\pi$ over $P$ where

$$
\pi_i = \frac{1}{|P|} \left( \eta^+ - 2(\eta^+ - 1) \frac{i - 1}{|P| - 1} \right)
$$

is the probability of the element $x_i$ with the $i$th largest contributing hypervolume
8: for $i \leftarrow 1$ to $\lambda - m$ do ▷ Randomly select remaining points
9: Draw $x \sim \pi$ ▷ Select points with larger $\Delta_r$ with higher probability
10: $O \leftarrow O \cup x$
11: end for
12: return $O$

run the algorithm in the 4V and 5V space of models trained for 108 epochs. The optimisation is performed over 100 evolutions with a population size of 20. All the experiments are conducted on a desktop workstation with a single NVIDIA RTX 3090 GPU with 24GB memory and Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz.

**Performance Criteria.** For the multi-objective optimisation, we use the validation accuracy ($P_v$) and the training energy cost, $E$(kWh), as the two objectives to be jointly optimised using the MOO algorithm. For the single-objective optimisation, we only use $P_v$ as the performance objective. We use energy cost rather than, e.g., training time, considering that $E$ is agnostic to parallel computing. We note that it is possible to use any of the provided metrics in Table 1 for the purpose of single- and multi-objective optimisation. As the MOO algorithm minimises the objectives, we simply use the negative of the objectives in cases where the quantities are to be maximised; for instance, we optimise $-P_v$ as accuracy is a maximisation objective.

**Training costs** In aggregate, EC-NAS-Bench had a total estimated training cost of 124,214 GPU days, 2021.02 kWh and 259.047 kgCO$_2$eq for the 5V space. The 4V space had a total estimated training cost of 3.854 GPU days, 63.792 kWh and 5.981 kgCO$_2$eq. The actual training costs for the 5V space were only 3.105 GPU days, 50.525 kWh and 6.476 kgCO$_2$eq. Actual training costs of the 4V space were 0.096 GPU days, 1.594 kWh and 0.149 kgCO$_2$eq.

In total, we saved an estimated compute cost of 121.109 GPU days, 1970.495 kWh and 252.571 kgCO$_2$eq for the 5V space, and 3.758 GPU days, 48.931 kWh and 6.327 kgCO$_2$eq for the 4V space. We obtain $\approx 97\%$ reduction in computing resources and energy consumption in all efficiency measures.

4 RESULTS

**Multi-objective exploration.** The key results from the experiments on EC-NAS-Bench using the multi-objective optimisation of $E$ and $-P_v$ are shown in Figure 2-a),b) and c). The Pareto front obtained from the MOO algorithm for one run is shown in Figure 2-a). It also shows the extrema ($r_0, r_1$) on both ends of the front preferring one of the objectives, whereas the knee point ($r_k$) offers the best trade-off between the two objectives. These three points are shown in different colours and markers. We compute the bend-angles to find the knee point as suggested by Deb & Gupta (2011). Pareto fronts over multiple random initialisations of the MOO algorithm are visualised as an attainment curve in Figure 2-c) which summarises the median solutions attained over the multiple runs(Fonseca et al., 2001).
Figure 2: a) Pareto front obtained for one of the MOO runs shows the family of solutions as discrete points. b) Radar plot showing three solutions from a Pareto front that is close to the median in subfigure a). The three architectures are chosen to correspond to the two extrema ($A_{r_0}$ in blue, $A_{r_1}$ in green) and the knee point ($A_{r_k}$ in yellow). The plot has four axes capturing the performance $P_v$ and resource consumption measured in $E$, $T$, $|\theta|$. c) The attainment curve showing the median solutions and the inter-quartile variations for 100 random initialisations of the MOO algorithm on the EC-NAS-Bench dataset. The two objectives being optimised ($-P_v$, $E$) are shown on the two axes. d) Radar plot for the solution obtained using SOO, which spans a larger area on the resource axes but also achieves the highest accuracy.

The architectures corresponding to the two extrema ($A_{r_0}$, $A_{r_1}$) and the corresponding knee point ($A_{r_k}$) for a single MOO run are visualised in the radar plot in Figure 2-b). The exact performance metrics for the three models in Figure 2-b) are also reported in Table 2. The solution covering the largest area is one of the extremal points ($A_{r_0}$, blue) with high accuracy (0.944) but also a larger footprint in the energy consumption (1.032kWh), computation time (5482.78s) and the number of parameters (21.22M) compared to the other extremum ($A_{r_1}$, green) or the knee point ($A_{r_k}$, yellow). The model corresponding to the knee point ($A_{r_k}$) provides a large reduction in the energy consumption (0.324kWh) at the expense of a small reduction in performance (0.932).

**Single-objective exploration.** We optimise only the validation accuracy, $P_v$, to simulate standard NAS practices. The resulting solution is shown in Figure 2-d) and in the last row of Table 2. This SOO model achieves the highest validation accuracy (0.944). However, the footprint of the solution along the energy consumption, computation time and the number of parameter axes are larger than those from the MOO algorithm.
5 DISCUSSIONS

Single versus multi-objective optimisation. The performance trends of the SOO and MOO solutions are clearly captured in Figure 2 and Table 2. The knee point solution, $A_r^k$, from MOO, yields an architecture that consumes about 70% less energy and has only about 1% degradation in performance. Depending on the downstream tasks, this could be a reasonable trade-off. If the degradation in performance cannot be tolerated, the Pareto front offers other candidate solutions for the practitioners to choose from. For instance, the extremum solution ($A_r^0$) offers basically the same performance as the SOO solution by consuming about 32% less energy.

Training time is not an alternative to energy consumption. The original NAS-Bench-101 already reports the training time (Ying et al., 2019). In single hardware regimes, this could serve as a measure of the energy consumption, as training time mostly correlates with the energy consumption. However, as most neural architecture training is performed on multiple GPUs with large-scale parallelism, training time alone cannot capture the efficiency of models. Aggregate energy consumption can take parallel hardware and the associated overheads into consideration. Even in single GPU training settings, energy consumption could optimise for energy-efficient models. For instance, a small architecture trained on a large GPU still has larger energy consumption due to the under-utilisation of the hardware resources. In such instances, a larger model could (to a certain extent) yield more performance improvements for the total energy consumed ($P_v/E$).

Energy efficient tabular NAS benchmark for obtaining efficient architectures. Tabular benchmarks such as NAS-Bench-101 (Ying et al., 2019) were introduced to reduce the resources required to perform NAS. However, even the one-time cost of generating a tabular benchmark dataset is massive. Surrogate NAS benchmarks are being studied to alleviate these costs, where the models are not exhaustively trained and evaluated. Instead, the performance metrics of architectures are estimated based on smaller training costs. For instance, this is achieved using predictive modelling based on learning curves (Yan et al., 2021), gradient approximations (Xu et al., 2021), or by fitting surrogate models to a subset of architectures (Zela et al., 2022). Similar to these attempts, the proposed EC-NAS-Bench dataset does not train all the models but bases its predictions on training the models only for 4 epochs, as described in Section 3.2. This results in about 97% reduction if the dataset were to be created from scratch, as shown in Table 3. Thus, EC-NAS-Bench is an energy-efficient tabular benchmark that can be used to obtain energy-efficient architectures as demonstrated in Section 4.

Carbon-footprint aware NAS. The EC-NAS-Bench dataset reports several metrics per architecture, as shown in Table 1. Combinations of these metrics and the use of MOO could allow for the exploration of architecture spaces that have interesting properties. For instance, NAS can be performed to directly optimise the carbon footprint of deep learning models. Although instantaneous energy consumption and carbon footprint are linearly correlated, when measured over a longer duration (>5m) these quantities differ due to the fluctuations of the instantaneous carbon intensity (Anthony et al., 2020). These carbon intensity fluctuations are caused by the variations of the power sources to the grid (Henderson et al., 2020). This can have implications when training models for a longer duration or on cloud instances that can be distributed over data centres in different countries (Dodge et al., 2022). By reporting instantaneous and aggregate carbon footprint of model
training in \texttt{EC-NAS-Bench} we facilitate the possibility of carbon footprint aware NAS (Selvan \textit{et al.}, 2022). In this work, we focused only on energy consumption awareness to work around the temporal- and spatial variations of the carbon intensity.

\textbf{Limitations.} Constraining the number of vertices in the DAGs results in sparser search spaces for the optimisation strategy. The optimisation strategy will therefore be more sensitive to initialisation and choice of random seeds, and the empirical Pareto front will appear to be more rigid, as seen in the attainment plot in Figure 2-c, even when multiple initialisations and trials are carried out. We also only demonstrated experiments on the 4V and 5V spaces.

To reduce the computation cost, in \texttt{EC-NAS-Bench} we used the surrogate time and energy measurements that do not model training time variability. We also query the performance metrics from the three repeats of NAS-Bench-101 and update \texttt{EC-NAS-Bench} with their mean performance metrics.

All these limitations are primarily driven by the need to minimise the energy consumption of these experiments. While these are at the expense of variability, we argue that the resulting reduction in the energy consumption justifies these choices. Further, the results from these small-scale experiments have been shown to extend to larger space of architectures (Ying \textit{et al.}, 2019).

\section{Conclusions and Future Work}

In this work, we presented an updated tabular benchmark dataset, \texttt{EC-NAS-Bench}, which tabulates the energy consumption and carbon footprint of training models, in addition to standard performance measures. Using multi-objective optimisation strategies, we showed that Pareto-optimal solutions offer appealing trade-offs between the performance measures and the energy consumption of model training. We qualitatively showed that large reductions (about 70\%) in energy consumption are possible with <1\% reduction in performance.

In addition to providing energy consumption measures, the \texttt{EC-NAS-Bench} benchmark provides metrics such as average carbon footprint and power consumption of CPUs, GPUs and DRAM. We hope this will foster interest in the development of models that are efficient and environmentally friendly by optimising for their energy consumption and carbon footprint.

\textbf{REFERENCES}


A  APPENDIX

A.1  MEASUREMENTS FROM CARBONTRACKER

We modify the open-source tool Carbontracker (Anthony et al., 2020) to measure the additional metrics reported in Table 1. Measurements take into account the energy usage of Graphical Processing Units (GPU), Central Processing Units (CPU), and Dynamic Random Access Memory (DRAM). Note that the energy usage for CPUs will include the power usage of DRAM. Power usage information is monitored, logged every 10 seconds, and reported as the average power usage during model training. Power is measured as the average of total units of a watt (W) over 10-second intervals during model training. The integral power consumed over the time a time interval, energy, is then reported in units of kilowatt-hours (kWh) with 1 kWh = \(3.6 \times 10^6\) Joule (J). Additionally, the emission of greenhouse gasses (GHG) is measured by equivalent units measured in grams of carbon dioxide (CO\(_2\)eq). The CO\(_2\)eq is then estimated by using the carbon intensity - CO\(_2\)eq units necessary to produce one unit of electricity a kilowatt per hour (kWh) - to express the carbon footprint of model training. The quantities for carbon intensity are fetched from the carbon intensity data provider every 15 minutes during model training.

Measurements from the aforementioned components alone do not give an accurate depiction of the carbon footprint model training when taking into account the energy consumption of the supporting infrastructure (e.g., data centre) is not considered. Therefore the quality of energy and carbon footprint estimations is amended by multiplying the power measurements by the PUE of the data centre hosting the compute resources. We use a PUE of 1.59, which is the global average for data centres in 2020 (Ascierto & Lawrence, 2020).

A.2  ADDITIONAL RESULTS

The results in Figure 2 were reported for the 5V space. The EC-NAS-Bench dataset also consists of the complete 4V space. In this section we report the MOO and SOO solutions based on the 4V search space. The trends observed for the 5V space hold for this smaller space as well.

| Model | Strategy | |θ| (Mbs) | T (s) | \(P_r\) | \(E\) (kWh) |
|-------|----------|----------|-------|-------|-------|----------|
| \(A_{r_0}\) (B) | MOO | 21.22 | 5461.66 | 0.944 | 1.015 |
| \(A_{r_1}\) (G) | MOO | 0.88 | 815.93 | 0.83 | 0.153 |
| \(A_{r_2}\) (Y) | MOO | 3.20 | 1782.80 | 0.925 | 0.355 |
| \(A_{r}\) (R) | SOO | 21.22 | 5482.78 | 0.944 | 1.032 |

Table 4: Metrics for models in single- and multi-objective setting seen in Figure 3. For SOO the optimal solution (\(A_{r}\)/Red) is reported. For MOO the two extrema (\(A_{r_0}\)/Blue, \(A_{r_1}\)/Green) and the knee point (\(A_{r}\)/yellow) are reported.
Figure 3: Results for the 4V space. a) Pareto front obtained for one of the MOO runs shows the family of solutions as discrete points. b) Radar plot showing three solutions from a Pareto front that is close to the median in subfigure a). The three architectures are chosen to correspond to the two extrema (\(A_{r0}\) in blue, \(A_{r1}\) in green) and the knee point (\(A_{rk}\) in yellow). The plot has four axes capturing the performance \(P_v\) and resource consumption measured in \(E, T, |\theta|\). c) The attainment curve showing the median solutions and the inter-quartile variations for 100 random initialisations of the MOO algorithm on the EC-NAS-Bench dataset. The two objectives being optimised (\(-P_v, E\)) are shown on the two axes. d) Radar plot for the solution obtained using SOO, which spans larger area on the resource axes but also achieves the highest accuracy.