ENERGY CONSUMPTION-AWARE TABULAR BENCHMARKS FOR NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

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 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; Lin et al., 2021; Zela et al., 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.

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 Energy 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 Architecture 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, $\mathbb{X}$, is limited to the topological space of *cells*, where each cell is a configurable feedforward network.

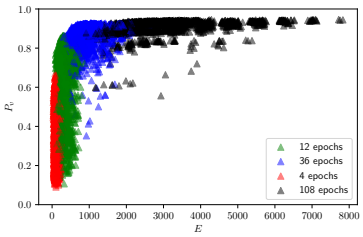

Figure 1: Visualisation of the 5V space of architectures with the validation performance P_v and the corresponding training energy cost (E) for four training epoch budgets (4,12,36,108).

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 $\mathcal{L}' = \{\texttt{input}, \texttt{output}\} \cup \mathcal{L}$, where $\mathcal{L} = \{\texttt{3x3conv}, \texttt{1x1conv}, \texttt{3x3maxpool}\}$. 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 $\mathcal{L}$. 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, $\mathcal{A}$, every entry $\alpha_{i,j} \in M_{\mathcal{A}}$ denotes an edge, from node i to node j with operations $i, j \in \mathcal{L}$ and its labelled adjacency matrix, $L_{\mathcal{A}} \in M_{\mathcal{A}} \times \mathcal{L}'$.

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).

The energy consumption during the training of a model, on the other hand, 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_2eq) 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

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	Unit of measurement	Notation
Model parameters	Million (M)	$\|\theta\|$
Test/Train/Eval. time	Seconds (s)	$T(s)$
Test/Train/Val. Acc.	$\mathbb{R} \in [0; 1]$	P_v
Energy consumption	Kilowatt-hour (kWh)	E(kWh)
Power consumption	Joule (J), Watt (W)	E(J), E(W)
Carbon footprint	kgCO_2eq	–
Carbon intensity	g/kWh	–

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

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 $\boldsymbol{f}\colon \mathbb{X} \to \mathbb{R}^m,\ \boldsymbol{f}(x) \mapsto (f_1(x), \ldots, f_m(x))$. 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 $x' \prec x$ if $\forall i \in \{1, \ldots, m\}\colon f_i(x') \leq f_i(x) \wedge \exists j \in \{1, \ldots, m\}\colon f_j(x') < f_j(x)$. For $X', X'' \subseteq \mathbb{X}$ we say that X' dominates X'' and write $X' \prec X''$ if $\forall x'' \in X'' : \exists x' \in X' : x' \prec X''$. The subset of non-dominated solutions in a set $X' \subseteq \mathbb{X}$ is given by $\mathrm{ndom}(X') = \{x \mid x \in X' \wedge \nexists x' \in X' \setminus \{x\} : x' \prec x\}$. The *Pareto front* of a set $X' \subset \mathbb{X}$ defined as $\mathcal{F}(X') = \{\boldsymbol{f}(x) \mid x \in \mathrm{ndom}(X')\}$ 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 $X' \prec X''$ nor $X'' \prec X'$. 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 $\boldsymbol{r} \in \mathbb{R}^m$:

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

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 Bringmann 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 μ, namely $\Theta(1/\mu)$ for additive approximation and $1 + \Theta(1/\mu)$ for relative approximation (Bringmann & Friedrich, 2013). Now we define the *contributing hypervolume* of an individual $x \in X'$ as

$$\Delta_{\boldsymbol{r}}(x, X') := \mathcal{S}_{\boldsymbol{r}}(X') - \mathcal{S}_{\boldsymbol{r}}(X' \setminus \{x\})\ .$$

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 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 $\mathcal{F}(P)$, 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 λ new architectures are generated by first selecting λ architectures from P and then modifying these architectures according to the perturbation described in Procedure 2. The λ new

Algorithm 1 Multi-Objective optimisation NAS strategy

Input: objective $\boldsymbol{f} = (f_1, \ldots, f_m)$, maximum number of iterations n
Output: set of non-dominated solutions P

```
Initialize P ⊂ 𝕏 (e.g., randomly)                ▷ Initial random architectures
P ← ndom(P)                                       ▷ Discard dominated solutions
for i ← 1 to n do                                 ▷ Loop over iterations
    O ← LinearRankSample(P, λ)                    ▷ Get λ points from P
    O ← Perturb(O)                                ▷ Change the architectures
    Compute f(x) for all x ∈ O                    ▷ Evaluate architectures
    P ← ndom(P ∪ O)                               ▷ Discard dominated points
end for
return P
```

Procedure 2 Perturb(O)

Input: set of architectures O, variation probabilities for edges and nodes p_{edge} and p_{node}
Output: set of modified architecture O^*

```
for all M_A ∈ O do                                ▷ Loop over matrices
    repeat
        for all α_{i,j} ∈ M_A do                  ▷ Loop over entries
            With probability p_edge flip α_{i,j}
        end for
        for all l ∈ L_A do                        ▷ Loop over labels
            With probability p_node change the label of l
        end for
    until M_A has changed
end for
return O*
```

architectures are added to P and the next iteration starts. In Procedure 2, the probability p_{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_{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 $\boldsymbol{r}$ 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 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-

Procedure 3 LinearRankSample(P, λ)

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

$O = \emptyset$
for $i \leftarrow 1$ to m **do**
  $O \leftarrow O \cup \text{argmin}_{x \in P} f_i(x)$ ▷ Always add extremes
end for
Compute $\Delta_{\boldsymbol{r}}(x, P)$ for all $x \in P$ ▷ Compute contributing hypervolume
Sort P according to $\Delta(x, P)$
Define discrete probability distribution π 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 ith largest contributing hypervolume
for $i \leftarrow 1$ to $\lambda - m$ **do** ▷ Randomly select remaining points
  Draw $x \sim \pi$ ▷ Select points with larger $\Delta_{\boldsymbol{r}}$ with higher probability
  $O \leftarrow O \cup x$
end for
return O

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 $kgCO_2eq$ for the 5V space. The 4V space had a total estimated training cost of 3.854 GPU days, 63.792 kWh and 5.981 $kgCO_2eq$. The actual training costs for the 5V space were only 3.105 GPU days, 50.525 kWh and 6.476 $kgCO_2eq$. Actual training costs of the 4V space were 0.096 GPU days, 1.594 kWh and 0.149 $kgCO_2eq$.

In total, we saved an estimated compute cost of 121.109 GPU days, 1970.495 kWh and 252.571 $kgCO_2eq$ for the 5V space, and 3.758 GPU days, 48.931 kWh and 6.327 $kgCO_2eq$ 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 $(\mathbf{r}_0, \mathbf{r}_1)$ on both ends of the front preferring one of the objectives, whereas the knee point $(\mathbf{r}_k)$ 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).

The architectures corresponding to the two extrema ($\mathcal{A}_{\mathbf{r}_0}$,$\mathcal{A}_{\mathbf{r}_1}$) and the corresponding knee point ($\mathcal{A}_{\mathbf{r}_k}$) 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 ($\mathcal{A}_{\mathbf{r}_0}$, 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 ($\mathcal{A}_{\mathbf{r}_1}$, green) or the knee point ($\mathcal{A}_{\mathbf{r}_k}$, yellow). The model corresponding to the knee point ($\mathcal{A}_{\mathbf{r}_k}$) 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

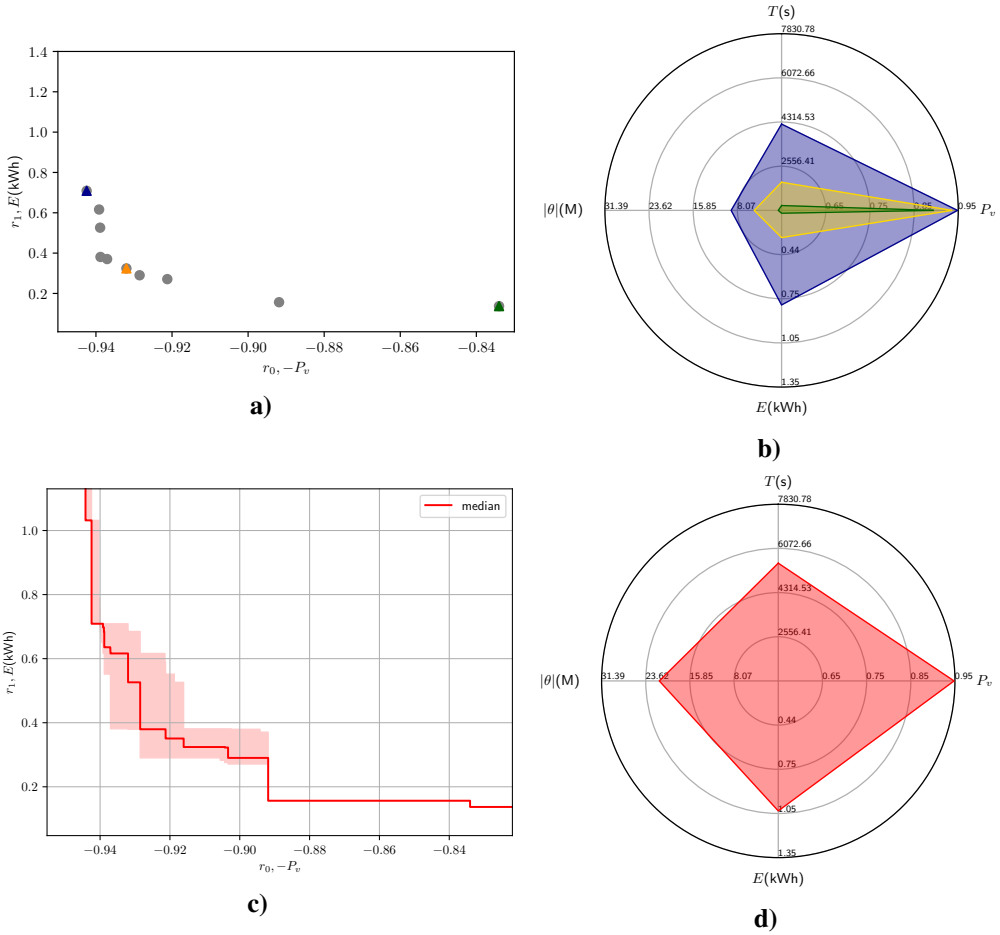

Figure 2: **a)** Pareto front obtained for one of the MOO runs shows the family of solutions as discrete points **b)** Radar plot showing three solutions from a Pareto front that is close to the median in subfigure a). The three architectures are chosen to correspond to the two extrema ($\mathcal{A}_{\mathbf{r}_0}$ in blue, $\mathcal{A}_{\mathbf{r}_1}$ in green) and the knee point ($\mathcal{A}_{\mathbf{r}_k}$ 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.

Model	Strategy	$\lvert\theta\rvert$(M)↓	$T(s)$ ↓	P_v ↑	E(kWh)↓
$\mathcal{A}_{\mathbf{r}_0}$ (B)	MOO	12.08	3878.66	0.942	0.709
$\mathcal{A}_{\mathbf{r}_1}$ (G)	MOO	**0.88**	**818.94**	0.834	**0.137**
$\mathcal{A}_{\mathbf{r}_k}$ (Y)	MOO	5.09	1916.65	0.932	0.324
$\mathcal{A}_{\mathbf{r}^*}$ (R)	SOO	21.22	5482.78	**0.944**	1.032

Table 2: Metrics for models in single- and multi-objective setting seen in Figure 2. For SOO the optimal solution ($\mathcal{A}_{\mathbf{r}^*}$/Red) is reported. For MOO the two extrema ($\mathcal{A}_{\mathbf{r}_0}$/Blue, $\mathcal{A}_{\mathbf{r}_1}$/Green) and the knee point ($\mathcal{A}_{\mathbf{r}_k}$/yellow) are reported.

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, $\mathcal{A}_{\mathbf{r}_k}$, 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 ($\mathcal{A}_{\mathbf{r}_0}$) 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.

Space	Red. GPU days	Red. kWh	Red. $kgCO_2eq$
4V	3.758	48.931	6.327
5V	121.109	1970.495	252.571

Table 3: Estimated reduction in total training costs w.r.t GPU days, kWh and $kgCO_2eq$ of the 4V and 5V space.

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 `EC-NAS-Bench` we facilitate the possibility of carbon footprint aware NAS (Selvan 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.

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 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 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 et al., 2019).

6 Conclusions and Future Work

In this work, we presented an updated tabular benchmark dataset, 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 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.

Acknowledgments

The authors would like to acknowledge a lot of people and some of the funding agencies.

References

Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems, July 2020. arXiv:2007.03051.

Rhonda Ascierto and Andy Lawrence. Uptime Institute global data center survey 2020. Technical report, Uptime Institute, 07 2020.

Johannes Bader and Eckart Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76, 2011.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02167.

Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction, 2017. URL https://arxiv.org/abs/1705.10823.

James Edward Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their Applications*, volume 1, 1985.

Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

Karl Bringmann and Tobias Friedrich. Approximation quality of the hypervolume indicator. *Artificial Intelligence*, 195:265–290, 2013.

Karl Bringmann, Tobias Friedrich, Christian Igel, and Thomas Voß. Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence*, 204:22–29, 2013.

Kalyanmoy Deb and Shivam Gupta. Understanding knee points in bicriteria problems and their implications as preferred solution principles. *Engineering Optimization*, 43(11):1175–1204, November 2011. doi: 10.1080/0305215x.2010.548863. URL https://doi.org/10.1080/0305215x.2010.548863.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. Measuring the carbon intensity of ai in cloud instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1877–1894, 2022.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search, 2020.

Viviane Grunert da Fonseca, Carlos M Fonseca, and Andreia O Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 213–225. Springer, 2001.

John J Greffenstette and James E Baker. How genetic algorithms work: A critical look at implicit parallelism. In *Proceedings of the 3rd International Conference on Genetic algorithms*, pp. 20–27, 1989.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Christian Igel, Nikolaus Hansen, and Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. *Advances in Neural Information Processing Systems*, 31, 2018.

Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization, 2019. URL https://arxiv.org/abs/1905.04970.

Oswin Krause, Tobias Glasmachers, and Christian Igel. Multi-objective optimization with unbounded solution sets. In *NeurIPS Workshop on Bayesian Optimization (BayesOpt 2016)*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition, 2021. URL https://arxiv.org/abs/2102.01063.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search, 2017. URL https://arxiv.org/abs/1712.00559.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018. URL https://arxiv.org/abs/1806.09055.

George Philipp. The nonlinearity coefficient - a practical guide to neural architecture design, 2021. URL https://arxiv.org/abs/2105.12210.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2902–2911. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/real17a.html.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions, 2020. URL https://arxiv.org/abs/2006.02903.

Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing. 2021. doi: 10.5281/zenodo.4658424.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019. URL https://arxiv.org/abs/1907.10597.

Raghavendra Selvan, Nikhil Bhagwat, Lasse F Wolff Anthony, Benjamin Kanding, and Erik B Dam. Carbon footprint of selecting and training deep learning models for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 506–516. Springer, 2022.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019a.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019b. doi: 10.48550/ARXIV.1905.11946. URL https://arxiv.org/abs/1905.11946.

Wei Wen, Hanxiao Liu, Hai Li, Yiran Chen, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search, 2019. URL https://arxiv.org/abs/1912.00848.

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

Shen Yan, Colin White, Yash Savani, and Frank Hutter. Nas-bench-x11 and the power of learning curves. *Advances in Neural Information Processing Systems*, 34:22534–22549, 2021.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/ying19a.html.

Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search, 2020. URL https://arxiv.org/abs/2001.10422.

Arber Zela, Julien Niklas Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OnpFa95RVqs.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2016. URL https://arxiv.org/abs/1611.01578.

A APPENDIX

A.1 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.

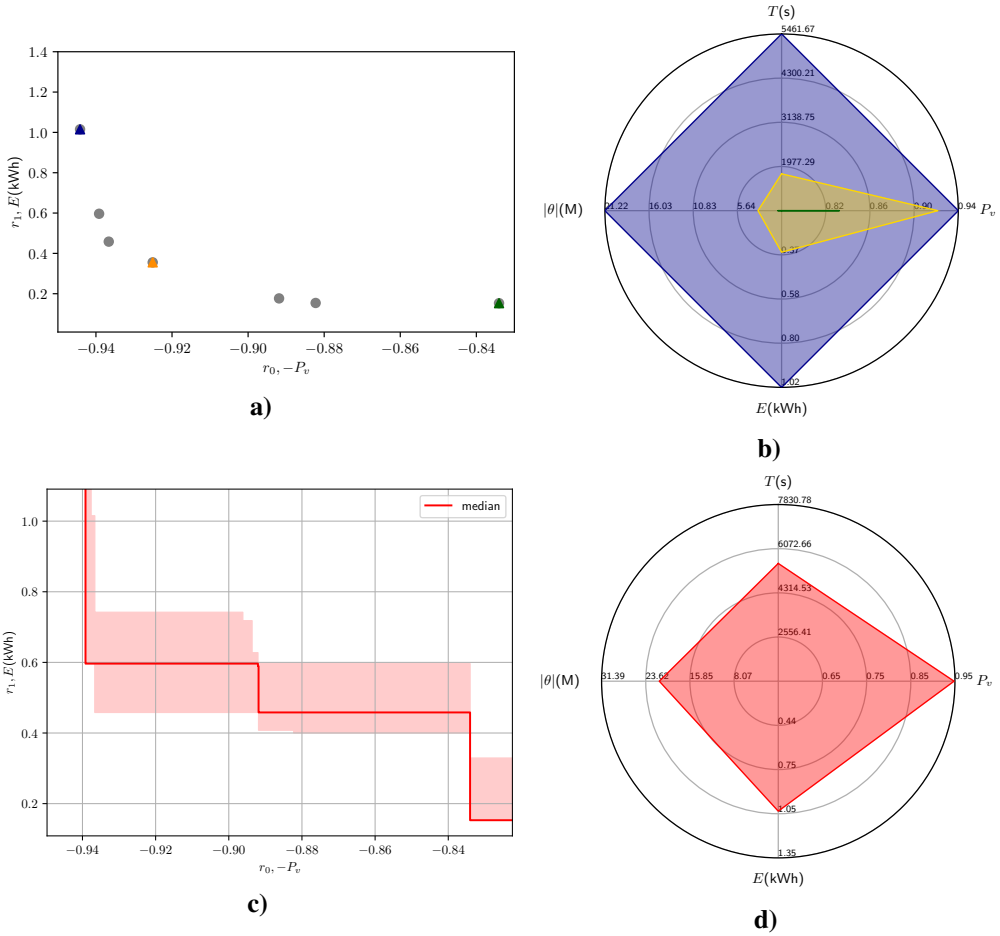

Figure 3: Results for the 4V space. **a)** Pareto front obtained for one of the MOO runs shows the family of solutions as discrete points **b)** Radar plot showing three solutions from a Pareto front that is close to the median in subfigure a). The three architectures are chosen to correspond to the two extrema ($\mathcal{A}_{\mathbf{r}_0}$ in blue, $\mathcal{A}_{\mathbf{r}_1}$ in green) and the knee point ($\mathcal{A}_{\mathbf{r}_k}$ 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 larger area on the resource axes but also achieves the highest accuracy.

A.2 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

Model	Strategy	$\lvert\theta\rvert$(M)↓	$T(s)$ ↓	P_v ↑	E(kWh)↓
$\mathcal{A}_{\mathbf{r}_0}$ (B)	MOO	21.22	5461.66	0.944	1.015
$\mathcal{A}_{\mathbf{r}_1}$ (G)	MOO	**0.88**	**815.93**	0.83	**0.153**
$\mathcal{A}_{\mathbf{r}_k}$ (Y)	MOO	3.20	1782.80	0.925	0.355
$\mathcal{A}_{\mathbf{r}^*}$ (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 ($\mathcal{A}_{\mathbf{r}^*}$/Red) is reported. For MOO the two extrema ($\mathcal{A}_{\mathbf{r}_0}$/Blue, $\mathcal{A}_{\mathbf{r}_1}$/Green) and the knee point ($\mathcal{A}_{\mathbf{r}_k}$/yellow) are reported.

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 1kWh $= 3.6 \cdot 10^6$Joule (J). Additionally, the emission of greenhouse gasses (GHG) is measured by equivalent units measured in grams of carbon dioxide (CO_2eq). The CO_2eq is then estimated by using the carbon intensity - CO_2eq 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).