PARETO RANK-PRESERVING SUPERNETWORK FOR HW-NAS

Anonymous authors
Paper under double-blind review

ABSTRACT

In neural architecture search (NAS), training every sampled architecture is very time-consuming and should be avoided. Weight-sharing is a promising solution to speed up the evaluation process. However, a sampled subnetwork is not guaranteed to be estimated precisely unless a complete individual training process is done. Additionally, practical deep learning engineering processes require incorporating realistic hardware-performance metrics into the NAS evaluation process, also known as hardware-aware NAS (HW-NAS). HW-NAS results a Pareto front, a set of all architectures that optimize conflicting objectives, i.e. task-specific performance and hardware efficiency. This paper proposes a supernetwork training methodology that preserves the Pareto ranking between its different subnetworks resulting in more efficient and accurate neural networks for a variety of hardware platforms. The results show a 97% near Pareto front approximation in less than 2 GPU days of search, which provides x2 speed up compared to state-of-the-art methods. We validate our methodology on NAS-Bench-201, DARTS and ImageNet. Our optimal model achieves 77.2% accuracy (+1.7% compared to baseline) with an inference time of 3.68ms on Edge GPU for ImageNet.

1 INTRODUCTION

A key element in solving real-world deep learning (DL) problems is the optimal selection of the sequence of operations and their hyperparameters, called *DL architecture*. Neural architecture search (NAS) (Santra et al. (2021)) automates the design of DL architectures by searching for the best architecture within a set of possible architectures, called *search space*. When considering hardware constraints, hardware-aware neural architecture search (Benmeziane et al. (2021); Sekanina (2021)) (HW-NAS) simultaneously optimizes the task-specific performance, such as accuracy, and the hardware efficiency computed by the latency, energy consumption, memory occupancy, and chip area. HW-NAS works (Cai et al. (2019); Lin et al. (2021); Wang et al. (2022)) showed the usefulness and discovered state-of-the-art architectures for Image Classification (Lin et al. (2021)), Object detection (Chen et al. (2019)), and Keyword spotting (Busia et al. (2022)).

HW-NAS is cast as a multi-objective optimization problem. Techniques for HW-NAS span evolutionary search, Bayesian optimization, reinforcement learning and gradient-based methods. These require evaluating each sampled architecture on the targeted task and hardware platform. However, the evaluation is extremely time-consuming, especially for task-specific performance, which requires training in the architecture. Many estimation strategies (White et al. (2021)) are used to alleviate this problem, such as neural predictor methods (Benmeziane et al. (2022); Ning et al. (2020)), zero-cost learning (Lopes et al. (2021); Abdelfattah et al. (2021)), and weight sharing (Chu et al. (2021); Chen et al. (2021)). These strategies are evaluated on how well they respect the ground truth ranking between the architectures in the search space.

Weight sharing is an estimation strategy that formulates the search space into a supernetwork. A supernetwork is an over-parameterized architecture where each path can be sampled. At the end of this sampling, a sub-network of the supernetwork is obtained. In each layer, all possible operations are trained. With this definition, we can classify weight-sharing NAS in two categories: (1)**a two-stage NAS** in which we first train the supernetwork on the targeted task. Then, using the pre-trained supernetwork, each sampled sub-network's performance can be estimated using a search strategy, such as an evolutionary algorithm. (2) **a one-stage NAS** in which we simultaneously search and

train the supernetwork. Additional parameters are assigned to each possible operation per layer. These parameters are trained to select which operation is appropriate for each layer.

Both Weight-sharing categories assume that the rank between different sub-networks is preserved. Two architectures with the same rank imply that they have the same accuracy. State-of-the-art works (Zhang et al. (2020); Peng et al. (2021); Zhao et al. (2021)) have highlighted the training inefficiency in this approach by computing the ranking correlation between the architectures' actual rankings and the estimated rankings. Some solutions have been proposed to train the supernetwork with strict constraints on fairness to preserve the ranking for accuracy, such as FairNAS (Chu et al. (2021)). Others train a graph convolutional network in parallel to fit the performance of sampled sub-networks Chen et al. (2021). However, current solutions have two main drawbacks:

1. In the multi-objective context of HW-NAS, different objectives such as accuracy and latency have to be estimated. The result is a Pareto front, a set of architectures that better respects the trade-off between the conflicting objectives. The ranking following one objective is no longer a good metric for the estimator. In this setting, we need to take into account the dominance concept in the ranking. Both estimations hinder the final Pareto front approximation and affect the search exploration when considering the accuracy and latency as objectives.
2. Many works (Chen et al. (2021); Zhao et al. (2021); Guo et al. (2020)) attempt to fix the supernetwork sampling after its training. We believe that this strategy is inefficient because: The supernetwork is pre-trained. Its accuracy-based ranking correlation is bad (0.47 Kendall's tau-b rank correlation coefficient on NAS-Bench-201 (Dong & Yang (2020))). The accuracy estimation is thus non-conclusive and will mislead any NAS search strategy.

To overcome the aforementioned issues, we propose a new training methodology for supernetworks to preserve the Pareto ranking of sub-networks in HW-NAS and avoid additional ranking correction steps. The contributions of this paper are summarized as follows:

- We define the **Pareto ranking as a novel metric** to compare HW-NAS evaluator in the multi-objective context. Our study shows that optimizing this metric while training the supernetwork increases the Kendall rank correlation coefficient from 0.47 to 0.97 for a Vanilla Weight-sharing NAS.
- We introduce a novel **one-stage weight-sharing supernetwork training methodology**. The training optimizes the task-specific loss function (e.g. cross-entropy loss) and a Pareto-ranking listwise loss function to select the adequate operation per layer accurately.
- During training, **we prune the operations that are the least likely to be in the architecture of the optimal Pareto front**. The pruning is done by overlapping the worst Pareto-ranked sub-networks and removing the operations that are only used in these sub-networks.

We demonstrate that using our methodology, we achieve a 97% near Pareto front approximation compared to current state-of-the-art methods (87% for One-Shot- NAS-GCN (Chen et al. (2021))) on three search space: NAS-Bench-201 (Dong & Yang (2020)), DARTS (Liu et al. (2019)), and ProxylessNAS search space (Cai et al. (2019)).

2 BACKGROUND

Given a target hardware platform and a DL task, **Hardware-aware Neural Architecture Search (HW-NAS)** (Benmeziane et al. (2021)) automates the design of efficient DL architectures. HW-NAS is a multi-objective optimization problem with three main components: (1) *the search space* that defines the set of architectures and their hyperparameters, (2) *the evaluation method* that estimates the task-specific performance as well as the hardware efficiency and (3) *the search strategy* that explore the search space to find the best architectures in terms of trade-off between the different objectives.

The main time-consuming component in NAS is architecture evaluation. Evaluating a single architecture requires hours counting its training deployment on embedded devices. State-of-the-art techniques (White et al. (2021)) have been used to alleviate this problem. **Predictor-based methods** (Ning et al. (2020); Benmeziane et al. (2022)) are the most popular strategies where machine

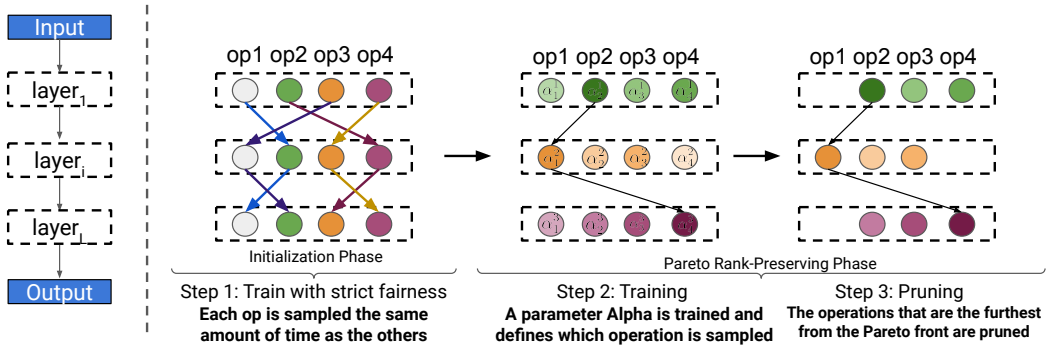

Figure 1: Our Pareto Rank-Preserving Training methodology for Supernetwork. The strongest shades illustrate the most important operations for each layer at each iteration. α_o^l corresponds to the parameter alpha associated with layer l, operation o.

learning models are used to predict the accuracy or latency from the architecture features (e.g. number of convolutions, widening factor, etc.) or its representation using Graph Neural Networks (GNN) (Ning et al. (2020)) or Recurrent Neural Networks (RNN) (Lomurno et al. (2021)). However, this method is not flexible to different search spaces as it requires first training a sampled dataset and then training the predictor. **Weight-sharing approaches** (Chu et al. (2021); Chen et al. (2021); Zhao et al. (2021); Guo et al. (2020)), on the other hand, define the search space as a supernetwork. In each layer, the supernetwork combines the results of possible operations. A sequence of operations from the input to the output is called a sub-network and constitutes a possible architecture.

Training the supernetwork consists on training several paths at once. The input is forwarded through a series of parallel operations which outputs are summed after each layer. The back-propagation adjusts the weights of each operation used during the forward. There are two main issues when training a supernetwork:

1. **The order of the sampled sub-networks matters:** Assume we have two sub-networks A and B. Both A and B start with the same operation op_1 in layer 1. During the first training iteration, A is sampled and op_1 weights are adjusted. The second iteration samples B and adjusts op_1 weights again. If we want now to evaluate A we would use the new adjusted weights of op_1 which degrades the estimation.
2. **Unfair Bias:** Sub-networks with an initial better task-specific performance are more likely to be sampled next and maintain a higher coefficient in one-stage supernetwork. Fairnas (Chu et al. (2021)) defines a strict fairness constraints that ensures that each operation's weights are updated the same amount of times at each stage.

3 METHODS

The core motivation for a novel training methodology is to achieve a valuable sub-networks evaluation for HW-NAS. Its purpose is two preserve the Pareto ranking between different sub-networks while reducing the overall search time. For simplification, we only show one operation selected per layer.

3.1 PARETO RANKING

In the HW-NAS multi-objective context, the result is a Pareto front, a set of architectures that dominates all other architectures in the search space. If A dominates B, then A beats or ties B on every objective. In this paper, we define the Pareto ranking as follows.

Pareto Ranking: Solving the problem on the complete supernetwork result in a Pareto front. The architectures within this front are Pareto ranked 1. We achieve the lower ranks by successfully solving the problem on the supernetwork pruned from the previous solutions. The lowest rank

is assigned to the sub-networks that do not dominate other sub-networks. We formally define the Pareto ranking in equation 1, where S is the entire supernetwork, $F_{k'}$ is a set of sub-networks ranked k', and $\succ$ is the dominance operation.

Using this ranking, multiple architectures may have the same rank. This happens when none of them can dominate the others.

$$\text{a is ranked k} \iff \forall \hat{a} \in S - \bigcup_{s_i \in F_{k'} \wedge k' < k}, \ \hat{a} \succ a \tag{1}$$

Pareto Ranking Correlation. We evaluate the quality of an estimator using ranking correlations such as Kendall's tau-b Correlation or Spearman Correlation. Kendall's tau-b determines whether there is a monotonic relationship between two variables and is suitable when variables contain many tied ranks Benmeziane et al. (2021), which is the case. In the rest of the paper, we compute Kendall's Tau-b correlation between the ground truth ranks (i.e. the Pareto ranks obtained from independently training the sub-networks), and the Pareto ranks obtained by evaluating each architecture with the supernetwork shared weights.

3.2 Pareto rank-preserving training

Our training methodology aims at preserving the Pareto ranking obtained by the weight-sharing evaluation.

Figure 2 shows a representation of the supernetwork definition and the different parameters we aim to learn. A sub-network is a path from the input to the output. All extracted sub-networks are of the same depth. We train the supernetwork with two goals: 1) to enhance the task-specific loss function by adjusting W, the task-specific weights of the original model associated with the neural network operations such as the kernels in convolution, and 2) to improve the Pareto-ranking loss between its different paths by adjusting α, the weights associated with the operation selection. α measures which operation is critical and which is selected.

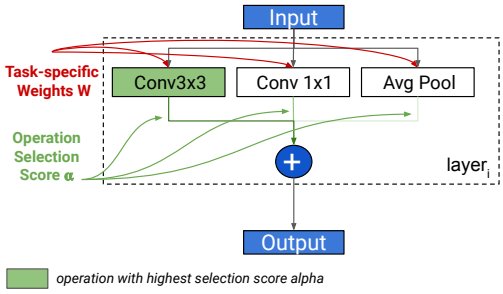

Figure 2: Supernetwork definition when coupling task-specific weights W and operation's score parameters α.

Algorithm 1 and figure 1 summarizes summarizes the training procedure.

- **Step 1: Train with Strict Fairness** We train our supernetwork using FairNAS (Chu et al. (2021)) strict fairness constraint. This step adjusts the weights of all the sub-networks W and gives a good starting point for the Pareto-ranking training.
- **Step 2: Pareto-Ranking training** For each iteration, we apply:
  - **Training to solve the task**
    A mini-batch is sampled from the training set, and a sub-network is chosen according to each operation's highest α.
  - **Pareto-rank training**
    After completing the iteration, we adjust each operation's Pareto-rank scores α. We first randomly sample n sub-networks. We compute each sampled sub-network's latency using a lookup table. We define $P_s = \sum_{op \in a} \alpha_{op}$, i.e., the sum of selected operations' alpha values as the Pareto score. Next, we compute the listwise ranking loss defined by the cross entropy between the ranking scores and the Pareto ranks (ground truth).
- **Step 3: Pruning by Pareto Ranking Sub-networks** We drop sub-networks furthest from the optimal Pareto front to accelerate the training. First, we select the sub-networks belong-

ing to the two first Pareto ranks. Then, based on the hypervolume improvement (HVI) (Emmerich et al. (2011)), we select n sub-networks. The operations never used by any sub-network in this selection are removed for each layer.

Finally, passing over all the layers to select the operations with the highest α would suffice to find the most efficient DNN within the search space.

Figure 3 shows the training results. We compare our methodology to FairNAS (Chu et al. (2021)) strict fairness training. During training, the Pareto ranking correlation increases with the quality of the estimations. When using our training methodology without considering the alpha parameters, the ranking correlation saturates at 0.7. FairNAS achieves the same behaviour with reduced variance among the different training runs. However, if we consider the alpha parameters, the selection is more efficient and the architectures' rankings are well represented with 0.94.

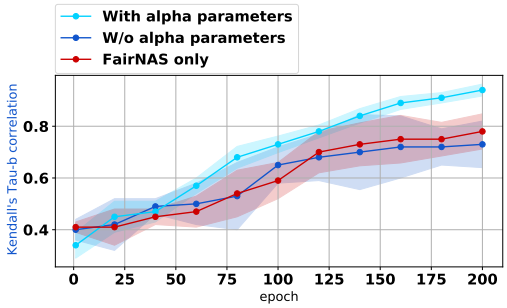

Figure 3: Training performance computed with the Kendall's Tau Correlation between the independently trained Pareto ranks and the estimated Pareto ranks obtained by training the supernetwork.

Algorithm 1 Supernetwork Training Algorithm

Input: Search space S, number of epochs for fairness training N_f, number of epochs for pareto-training N_p, Supernetwork parameters (W, α), training dataloader D, task-specific loss $Loss$, Pareto-raking loss $Loss_{PR}$, number of sampled sub-network n

procedure TRAIN
  Initialize W and α for each operation in Supernetwork
  Strict fairness training for N_f epochs
  for i=1 to N_p **do**
    for data, labels in D **do**
      Build $model$ with $argmax(\alpha)$
      Clear gradients recorder for all W parameters
      Calculate gradients based on $Loss$, data, labels
      Update W by gradients
    end for
    Sample n sub-networks, $models$
    Compute Pareto rank of $models$
    Compute $Loss_{PR}$ between scores and pareto rank.
    Adjust α
    end for
end procedure

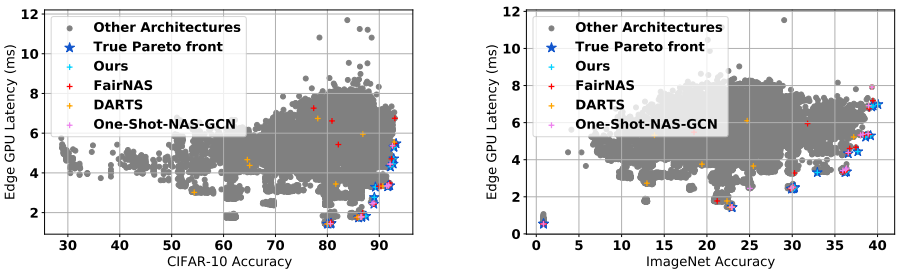

Figure 4: Pareto front approximation comparison on CIFAR-10 and ImageNet.

4 EXPERIMENTS

In this section, we evaluate our training methodology on three search spaces: NAS-Bench-201 (Dong & Yang (2020)), DARTS (Liu et al. (2019)) and ProxylessNAS Search space (Cai et al. (2019)).

4.1 SETUP

Search Spaces: Several search spaces have been used to evaluate our method's performance. NAS-Bench-201 (Dong & Yang (2020)) is a tabular benchmark that contains 15k convolutional neural networks. Each architecture is trained on CIFAR-10, CIFAR-100 and ImageNet-16-120 (Chrabaszcz et al. (2017)). We use the latency values obtained from HW-NAS-Bench (Li et al. (2021)). DARTS Liu et al. (2019) is a supernetwork benchmark that contains 10^{18} architectures. Each architecture is trained on CIFAR-10 and is transferable to ImageNet. Besides, we validates our methodology on ImageNet using ProxylessNAS search space Cai et al. (2019) whose size goes to 6^{19}.

Training Hyperparameters: The training hyperparameters are listed in Table 1. It takes 2, 3.8, 3.8 GPU-days for NAS-Bench-201, DARTS and ProxylessNAS search space to train each supernetwork to fullness. Our training is 5x faster than previous works due to the pruning strategy. To be consistent with previous works, we do not employ data augmentation tricks such as cutout or mixup. We also do not employ any special operations such as squeeze-and-excitation. All these methods can further improve the scores on the test set.

Benchmark	Hyperparameter	Value
NAS-Bench-201	Nf	20
	Np	50
	n	50
	batch_size	128
	lr	0.01
	optim	SGD
	momentum	0.9
DARTS	Nf	30
	Np	150
	n	100
	batch_size	256
	lr	0.025
	optim	SGD
	momentum	0.9
ProxylessNAS Search Space	Nf	30
	Np	150
	n	100
	batch_size	256
	lr	0.025
	optim	SGD
	momentum	0.9

Table 1: Training Hyperparameters

4.2 SEARCH RESULTS

In these experiments, we consider two objectives: accuracy and latency (inference time). The latency is either given by HW-NAS-Bench (Li et al. (2021)), or computed using a lookup table as explained in section 3.

Figure 4 shows the Pareto front approximations obtained using different methods on NAS-Bench-201 for CIFAR-10 and ProxylessNAS Search space for ImageNet. We obtain a 10% hypervolume increase on NAS-Bench-201 and a 43% hypervolume increase on ImageNet compared to the best baselines: One-Shot-NAS-GCN and FairNAS, respectively.

Architecture	Edge GPU			Mobile Phone : Pixel 3			HW Aware	GPU Days
	Top-1 Test Acc.	Params (M)	Latency (ms)	Top-1 Test Acc.	Params (M)	Latency (ms)		
DARTS Liu et al. (2019)	68.3 ± 0.08	3.4	5.36	68.3 ± 0.08	3.4	11.4	No	4
ENAS Pham et al. (2018)	53.89 ± 0.16	4.6	6.32	53.89 ± 0.16	4.6	19.8	No	0.16
GDAS Dong & Yang (2019)	90.89 ± 0.08	3.4	5.21	90.89 ± 0.08	3.4	10.36	No	0.21
FairNAS Chu et al. (2021)	93.23±0.18	3.2	4.68	92.4 ± 0.15	3.6	8.65	Yes	10
PRP-NAS-BL (Ours)	92.34 ± 0.05	3.0	**2.3**	89.54 ± 0.07	2.8	**3.6**	Yes	2
PRP-NAS-BA (Ours)	**94.5 ± 0.02**	4.5	4.35	**94.2 ± 0.03**	4.3	5.6	Yes	2
PRP-NAS-O (Ours)	93.65 ± 0.01	4.3	3.64	93.74 ± 0.00	3.4	4.61	Yes	2

Table 2: Comparison on NAS-Bench-201 CIFAR-10 on Edge GPU (Jetson Nano) and Mobile phone (Pixel 3).

4.2.1 SEARCH ON NAS-BENCH-201

Table 2 shows the results of our methodology on NAS-Bench-201 compared to state-of-the-art methods. PRP-NAS-BL, PRP-NAS-BA and PRP-NAS-O are three sampled architectures from our final Pareto front. BL stands for "Best Latency". BA stands for "Best Accuracy", and O stands for "Optimal". Notably, our architecture obtains highly competitive results. The optimal architecture, *PRP-NAS-O*, outperforms current state-of-the-art methods in accuracy and latency. Including hardware awareness during the search allows us to obtain flexible results according to the targeted hardware platform. Besides, multiple training runs show the stability of our method compared to other baselines. The acceleration in the search cost is mainly due to applying the pruning while training. This cost can vary according to the used GPU. We used GPU V100 to train the supernetwork. Results on other targeted platforms, can be found in Appendix A.

4.2.2 SEARCH ON IMAGENET

Similar conclusions can be extracted when searching on ImageNet. Table 3 summarizes the results. Our optimal model surpasses FairNAS-A (+1.9%) and One-Shot-NAS-GCN (+1.7%) while running faster. Training on Imagenet is time-consuming due to the difference in image resolution, which explains the increase in the search cost. We still surpass most of the methods in terms of search time. We compare two ProxylessNAS architectures; ProxylessNAS-R is specific to Mobile inference.

When using data augmentation and architecture tricks, namely Squeeze-and-excitation and AutoAugment, in the optimal architecture, we achieve 78.6% accuracy on Imagenet. However, this may affect the latency badly. On FPGA ZCU102, the latency increases from 4.63ms to 7.9ms.

4.3 RANKING QUALITY

The ranking preservation measures the quality of the evaluation component in NAS. In HW-NAS, we argue that this measure should consider the Pareto ranking instead of the independent ranks of each objective. We compare different estimators used in HW-NAS using Kendall's Tau Correlation between the predicted Pareto Ranks and the Pareto ranks obtained from independently training the architectures. These latter are extracted from NAS-Bench-201. Figure 5 shows the correlation results. In general, it is more complex to train a supernetwork to respect the Pareto ranks because of the impact of the sub-networks on each other, i.e., the outputs of each layer are summed together. The increase in Kendall's tau correlation of previous weight-sharing methodology is due to the improvement in the accuracy estimation provided by the supernetwork.

Architecture	Edge GPU			Mobile Phone : Pixel 3			HW Aware	GPU Days
	Top-1 Test Acc.	Params (M)	Latency (ms)	Top-1 Test Acc.	Params (M)	Latency (ms)		
DARTS Liu et al. (2019)	73.3 ± 0.03	4.7	5.36	73.3 ± 0.03	4.7	11.5	No	4
ProxylessNAS Cai et al. (2019)	75.1 ± 0.00	7.1	5.1	-	-	-	Yes	8.3
ProxylessNAS-R Cai et al. (2019)	-	-	-	74.6	4.0	6.8	Yes	8.3
FairNAS-A Chu et al. (2021)	75.3 ± 0.05	4.6	5.32	75.1 ± 0.06	4.3	6.84	Yes	12
One-Shot-NAS-GCN Chen et al. (2021)	75.5 ± 0.09	4.4	5.23	75.5 ± 0.09	4.4	9.4	No	4.7
PRP-NAS-BL (Ours)	68.95 ± 0.01	3.7	3.2	67.56 ± 0.05	3.1	4.8	Yes	3.8
PRP-NAS-BA (Ours)	77.5 ± 0.02	4.8	4.68	76.94 ± 0.02	4.4	6.13	Yes	3.8
PRP-NAS-O (Ours)	77.2 ± 0.03	4.6	3.68	75.6 ± 0.02	3.8	5.68	Yes	3.8

Table 3: Comparison on ImageNet on Edge GPU (Jetson Nano) and Mobile Phone (Pixel 3).

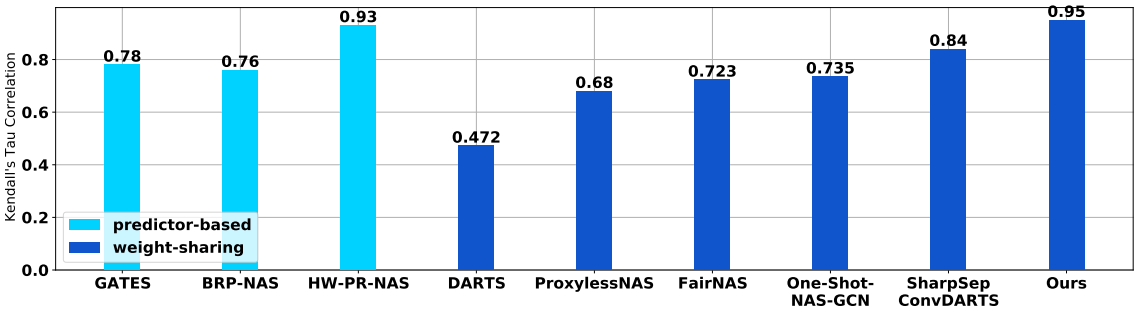

Figure 5: Kendall's Tau-b correlation of different estimators.

Predictor-based evaluators use the learning-to-rank theory and train their predictors only to predict the ranking. Methods such as GATES (Ning et al. (2020)) or BRP-NAS (Dudziak et al. (2020)) train many independent predictors, one for each objective. HW-PR-NAS (Benmeziane et al. (2022)) trains a single predictor to fit the Pareto ranks. However, their methodology is not flexible for supernetwork training.

4.4 Battery Usage Preservation

The amount of energy consumed by each model can be different. It is mainly attributed to the number of multi-adds computed. We take supernetwork usage to another level by adequately scheduling the run of different sub-networks according to the system's battery life. In this experiment, the training is done with two objectives: accuracy and energy consumption. Once the training is done, only the Pareto front solutions are kept in the supernetwork, thanks to the pruning. We further select from the final Pareto front s architectures. In this experiment $s = 5$. The total size of the supernetwork is then reduced to 20.5MB, comparable to MobileNet-V3 Large with 21.11MB. We deploy the model on a smartphone application that is always on. The application repeats the inference classification of one image. The application initially uses the sub-network with the highest accuracy. We switch

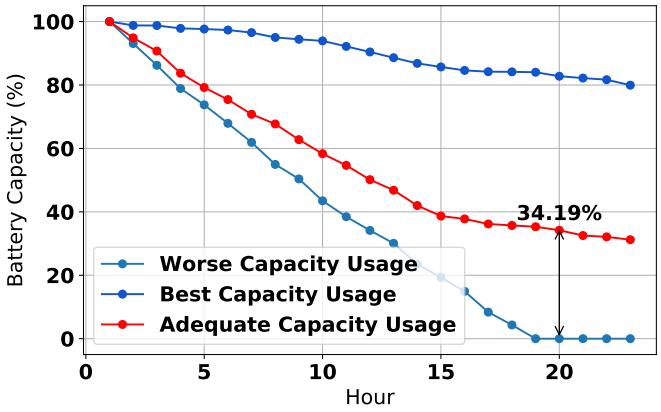

Figure 6: Battery life management.

to a lower accurate model every five hours but better energy preserving. Figure 6 shows the results of the system's battery life while running the application for 24 hours. We use three scenarios:

1. **Worst Battery Usage:** From the Pareto front, we select the most accurate architecture. This is the only architecture the application runs and is the only one loaded in memory.
2. **Best Battery Usage:** Similar to the worst battery usage, we select the most energy-efficient.
3. **Adequate Battery Usage:** We load the complete supernetwork and switch the sub-network every 5 hours.

Using this strategy helps save up to 34% of the battery life while using highly accurate models most of the time. The average accuracy of the five selected sub-networks is 75.2%.

5 Conclusion

This work analyzes Hardware-aware weight-sharing NAS where the multi-objective context requires the estimator to preserve the Pareto rankings between sub-networks accurately. Contrary to standard baselines that independently estimate each objective, we propose a supernetwork training methodology able to preserve the Pareto rankings during the search. Using our methodology, we achieve 97% near Pareto front approximation on NAS-Bench-201, DARTS, and ProxylessNAS Search Space. We find a 77.2% accuracy model on ImageNet while only training the supernetwork for 3.8 days. Using the supernetwork capabilities, we saved up to 34% of the battery capacity with an average accuracy of 75.2%.

REFERENCES

Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight NAS. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0cmMMy8J5q.

Hadjer Benmeziane, Kaoutar El Maghraoui, Hamza Ouarnoughi, Smaïl Niar, Martin Wistuba, and Naigang Wang. A comprehensive survey on hardware-aware neural architecture search. *CoRR*, abs/2101.09336, 2021.

Hadjer Benmeziane, Smaïl Niar, Hamza Ouarnoughi, and Kaoutar El Maghraoui. Pareto rank surrogate model for hardware-aware neural architecture search. In *International IEEE Symposium on Performance Analysis of Systems and Software, ISPASS 2022, Singapore, May 22-24, 2022*, pp. 267–276. IEEE, 2022. doi: 10.1109/ISPASS55109.2022.00040. URL https://doi.org/10.1109/ISPASS55109.2022.00040.

Paola Busia, Gianfranco Deriu, Luca Rinelli, Cristina Chesta, Luigi Raffo, and Paolo Meloni. Target-aware neural architecture search and deployment for keyword spotting. *IEEE Access*, 10: 40687–40700, 2022. doi: 10.1109/ACCESS.2022.3166939. URL https://doi.org/10.1109/ACCESS.2022.3166939.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HylVB3AqYm.

Xin Chen, Lingxi Xie, Jun Wu, Longhui Wei, Yuhui Xu, and Qi Tian. Fitting the search space of weight-sharing NAS with graph convolutional networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021*, pp. 7064–7072. AAAI Press, 2021.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 6638–6648, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/228b25587479f2fc7570428e8bcbabdc-Abstract.html.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *CoRR*, abs/1707.08819, 2017. URL http://arxiv.org/abs/1707.08819.

Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *International Conference on Computer Vision, ICCV 2021*, pp. 12219–12228. IEEE, 2021.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1761–1770. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00186. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Dong_Searching_for_a_Robust_Neural_Architecture_in_Four_GPU_Hours_CVPR_2019_paper.html.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Lukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. BRP-NAS: prediction-based NAS using gcns. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien

Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/768e78024aa8fdb9b8fe87be86f64745-Abstract.html.

Michael T. M. Emmerich, André H. Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, pp. 2147–2154. IEEE, 2011. doi: 10.1109/CEC.2011.5949880. URL https://doi.org/10.1109/CEC.2011.5949880.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVI*, volume 12361 of *Lecture Notes in Computer Science*, pp. 544–560. Springer, 2020. doi: 10.1007/978-3-030-58517-4_32. URL https://doi.org/10.1007/978-3-030-58517-4_32.

Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. Hw-nas-bench: Hardware-aware neural architecture search benchmark. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=_0kaDkv3dVf.

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 image recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 337–346. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00040. URL https://doi.org/10.1109/ICCV48922.2021.00040.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Eugenio Lomurno, Stefano Samele, Matteo Matteucci, and Danilo Ardagna. Pareto-optimal progressive neural architecture search. In Krzysztof Krawiec (ed.), *GECCO '21: Genetic and Evolutionary Computation Conference, Companion Volume, Lille, France, July 10-14, 2021*, pp. 1726–1734. ACM, 2021. doi: 10.1145/3449726.3463146. URL https://doi.org/10.1145/3449726.3463146.

Vasco Lopes, Saeid Alirezazadeh, and Luís A. Alexandre. EPE-NAS: efficient performance estimation without training for neural architecture search. In Igor Farkas, Paolo Masulli, Sebastian Otte, and Stefan Wermter (eds.), *Artificial Neural Networks and Machine Learning - ICANN 2021 - 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14-17, 2021, Proceedings, Part V*, volume 12895 of *Lecture Notes in Computer Science*, pp. 552–563. Springer, 2021. doi: 10.1007/978-3-030-86383-8_44. URL https://doi.org/10.1007/978-3-030-86383-8_44.

Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based NAS. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIII*, volume 12358 of *Lecture Notes in Computer Science*, pp. 189–204. Springer, 2020. doi: 10.1007/978-3-030-58601-0_12. URL https://doi.org/10.1007/978-3-030-58601-0_12.

Jiefeng Peng, Jiqi Zhang, Changlin Li, Guangrun Wang, Xiaodan Liang, and Liang Lin. Pi-nas: Improving neural architecture search by reducing supernet training consistency shift. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 12334–12344. IEEE, 2021. doi: 10.1109/ICCV48922.2021.01213. URL https://doi.org/10.1109/ICCV48922.2021.01213.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4092–4101. PMLR, 2018. URL http://proceedings.mlr.press/v80/pham18a.html.

Santanu Santra, Jun-Wei Hsieh, and Chi-Fang Lin. Gradient descent effects on differential neural architecture search: A survey. *IEEE Access*, 9:89602–89618, 2021. doi: 10.1109/ACCESS.2021.3090918. URL https://doi.org/10.1109/ACCESS.2021.3090918.

Lukas Sekanina. Neural architecture search and hardware accelerator co-search: A survey. *IEEE Access*, 9:151337–151362, 2021. doi: 10.1109/ACCESS.2021.3126685.

Linnan Wang, Chenhan Yu, Satish Salian, Slawomir Kierat, Szymon Migacz, and Alex Fit-Florea. Gpunet: Searching the deployable convolution neural networks for gpus. *CoRR*, abs/2205.00841, 2022. doi: 10.48550/arXiv.2205.00841. URL https://doi.org/10.48550/arXiv.2205.00841.

Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 28454–28469, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ef575e8837d065a1683c022d2077d342-Abstract.html.

Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *CoRR*, abs/2001.01431, 2020.

Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12707–12718. PMLR, 2021. URL http://proceedings.mlr.press/v139/zhao21d.html.

A ADDITIONAL RESULTS

Table 4 shows the results of our training methodology on FPGA ZCU 102 and Raspberry Pi3. Our methodology consistently outperforms state-of-the-art methods on different hardware platforms.

Architecture	FPGA ZCU102			Raspberry Pi 3			HW Aware	GPU Days
	Top-1 Test Acc.	Params (M)	Latency (ms)	Top-1 Test Acc.	Params (M)	Latency (ms)		
DARTS	68.3 ± 0.08	3.4	7.32	68.3 ± 0.08	3.4	45.36	No	4
ENAS	53.89 ± 0.16	4.6	8.91	53.89 ± 0.16	4.6	35.8	No	0.16
GDAS	90.89 ± 0.08	3.4	4.98	90.89 ± 0.08	3.4	41.8	No	0.21
FairNAS	92.9±0.23	3.4	5.12	92.51 ± 0.9	3.3	34.15	Yes	10
PRP-NAS-BL (Ours)	91.35 ± 0.04	3.2	3.6	88.7 ± 0.03	2.4	7.6	Yes	2
PRP-NAS-BA (Ours)	94.37 ± 0.01	4.9	6.8	93.68 ± 0.05	4.68	40.7	Yes	2
PRP-NAS-O (Ours)	93.55 ± 0.04	4.2	4.23	92.54 ± 0.02	3.6	18.5	Yes	2

Table 4: Comparison to baselines on CIFAR-10 on FPGA ZCU-102 and Raspberry Pi3