2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 978-1-6654-5954-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISPASS55109.2022.00040

Pareto Rank Surrogate Model for Hardware-aware Neural Architecture Search

Hadjer Benmeziane*, Smail Niar*, Hamza Ouarnoughi*, Kaoutar El Maghraoui†
*Université Polytechnique Hauts-de-France
LAMIH UMR CNRS, Valenciennes, France
{firstname.lastname}@uphf.fr
†IBM T. J. Watson Research Center, NY, USA
kelmaghr@us.ibm.com

***Abstract*—Hardware-aware Neural Architecture Search (HW-NAS) has recently gained much attention by automating the design of efficient deep learning models with tiny resources and reduced inference time requirements. However, HW-NAS inherits and exacerbates the expensive computational complexity of general NAS due to its significantly increased search spaces and more complex NAS evaluation component. To speed up HW-NAS, existing efforts use surrogate models to predict a neural architecture's accuracy and hardware performance on a specific platform. Thereby reducing the expensive training process and significantly reducing search time. We show that using multiple surrogate models to estimate the different objectives does not achieve the true Pareto front. Therefore, we propose HW-PR-NAS, a novel Pareto Rank-preserving surrogate model. HW-PR-NAS training is based on a new loss function that ranks the architectures according to their Pareto front. We evaluate our approach on seven different hardware platforms, including ASIC, FPGA, GPU and multi-cores. Our results show that we can achieve up to 2.5x speedup while achieving better Pareto-front results than state of the art surrogate models.**

***Index Terms*—Neural Architecture Search, Surrogate Models, Ranking Models, Hardware-aware Neural Architecture Search.**

I. Introduction

With the rise of Automatic Machine Learning (AutoML) techniques, significant progress has been made to automate the design of deep learning (DL) architectures. Among these techniques, Hardware-aware Neural Architecture Search (HW-NAS) [1], [2] has recently gained a lot of attention from the DL community. HW-NAS takes NAS a step further by automating the design of DL architectures on resource-constrained devices.

(2) A **Search algorithm** that is cast as an optimization algorithm such as random search or evolutionary algorithms. Often the search algorithm needs to be formulated as a multi-objective optimization algorithm when more than one hardware constraint needs to be satisfied. (3) **Evaluation methods** that compute the different objectives (e.g. accuracy, latency, energy consumption) given a sampled architecture and a target hardware platform.

NAS evaluation methods [3] involve training and evaluating a large set of architectures sampled from the search space. This requires extensive cloud-scale or large data center GPU computational resources for many days. For instance, if we consider that training a single architecture could take 2 hours to assess its accuracy, about 834 days would be necessary to complete the search for ten thousand sampled architectures. This time complexity is exacerbated further in the case of HW-NAS multi-objective assessments, as additional evaluations need to be performed for each objective or hardware constraint. To get around this expensive problem, researchers have proposed surrogate-assisted evaluation methods [4]–[6]. The key idea behind surrogate models is to use cheap analytical or ML-based algorithms that can quickly estimate the performance of every sample architecture in HW-NAS. The multi-objective context of HW-NAS requires the result to be a Pareto front approximation as an optimal decision needs to be taken in the presence of trade-offs between two or more conflicting objectives. For example, minimizing model inference latency while maximizing model accuracy. In multi-objective optimization, the Pareto front or Pareto frontier refers to the set of all solutions that are Pareto efficient or optimal. In other words, the set of the solutions where no individual objective can be better off without making at least another objective worse. In the context of NAS, the Pareto front represents the architecture where we cannot improve one objective without harming the other one. For example, we cannot improve the accuracy without increasing latency. The Pareto front solutions are essential because (1) They do not require specifying hard constraints or weights on objectives beforehand. For example, there is no need to define a specific threshold for the latency on each hardware platform, (2) They enable adaptability by choosing a different model from the Pareto front according to the platform's hardware specifications. Depending on the user's constraint (e.g., battery life or the total cost), we favour a specific architecture in the Pareto front. Existing HW-NAS research efforts [1], [7], [8] rely on the use of surrogate-assisted evaluations. However, this introduces false solutions as each surrogate model brings its share of approximation error. Thus potentially leading to search inefficiencies and falling into local optimum.

To address this, we propose using a single surrogate model, **HW-PR-NAS**, that directly predicts the Pareto front ranks. In other words, the relative performance of one architecture with respect to another one. To illustrate this further, Figure 1 summarizes the results of using a one-surrogate model NAS versus a two-surrogate model NAS. We use the hypervolume metric

978-1-6654-5954-9/22/$31.00 ©2022 IEEE
DOI 10.1109/ISPASS55109.2022.00040

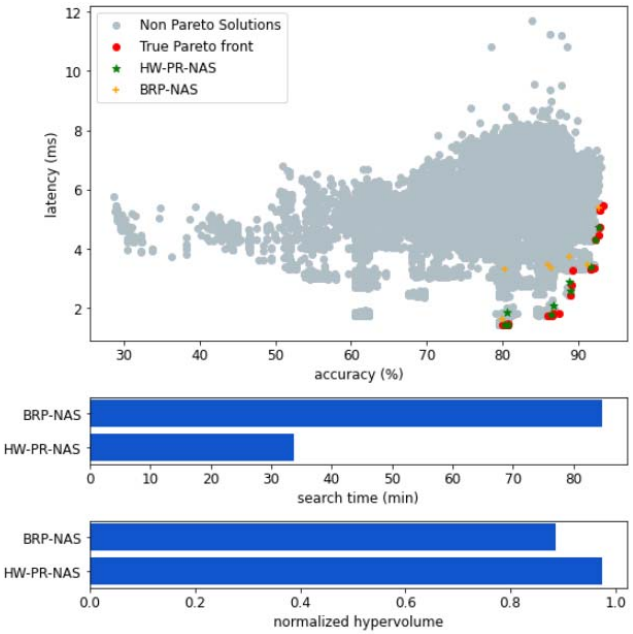

Fig. 1. This figure compares the use of one single Pareto surrogate model (HW-PR-NAS) to the use of two different surrogate models obtained from BRP-NAS [9] using NAS-Bench-201 search space [10] and the CIFAR10 dataset. a) the top figure highlights the Pareto front approximation obtained as well as the true Pareto front. The objectives are network accuracy and latency. The true Pareto front values represent the actual values of model accuracy and actual latency obtained from hardware measurements. b) The middle bar graph shows the search time speedup comparison. c) the bottom bar graph shows the normalized hypervolume comparison.

to compare two Pareto front approximations. The hypervolume computes the coverage and diversity of a Pareto front and is normalized against the true Pareto front. Figure 1 clearly shows that using HW-PR-NAS is more efficient than using two different surrogate models in terms of reducing the search time and improving the quality of the Pareto approximations.

In contrast to existing approaches that build a regression model for each separate objective and subsequently combine the independently predicted objective values within the NAS, we propose a surrogate model trained to precisely predict the Pareto front ranks of architecture for multiple objectives simultaneously. Using the predicted Pareto front ranks, selecting the best architecture represented by the Pareto front approximation can be achieved without expensive model training and evaluations. Our approach, illustrated in figure 2, leads to significant NAS acceleration while ensuring that the search arrives at a near true Pareto front.

The contributions of the paper are summarized as follows:

1) We introduce **a novel training methodology for multi-objective HW-NAS surrogate models** in section III-A. Our surrogate model is trained using a novel ranking loss technique. Its goal is to rank the architectures from dominant to non-dominant by giving high scores to the dominant ones.
2) Using this training methodology, we train our **multi-objective surrogate model, named Hardware-Pareto Rank NAS (HW-PR-NAS)**. HW-PR-NAS comprises two predictors to regress the objectives and enhance the Pareto score prediction. This approach is detailed in section III-B.

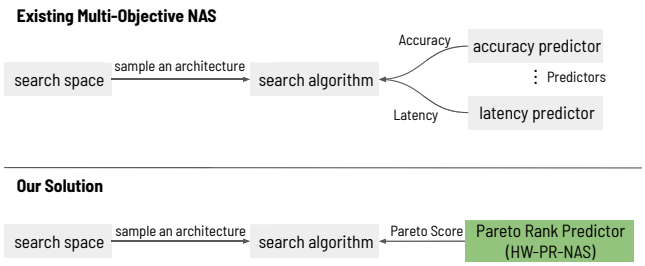

Fig. 2. Simplified illustration of the use of HW-PR-NAS in a NAS process.

The rest of this paper is organized as follows. Section II presents the related work and relevant concepts used in this work. We then describe our approach in Section III. We first explain our listwise ranking loss and the training procedure we used. Then, we overview the HW-PR-NAS system and detail the two main components, latency and accuracy predictors. Next, we describe a modified version that allows adding more metrics easily. Finally, in Section IV, we validate our approach by comparing our obtained Pareto front approximations with state-of-the-art surrogate models, namely GATES [11] and BRP-NAS [9] in a Multi-Objective Evolutionary Algorithm (MOEA). We also analyze how the hypervolume indicator is optimized. Finally, we prove, in section IV, that our system allows a faster (2.5 speedup) and more efficient (98% hypervolume of the true Pareto front) NAS process. Our experimentation is done on two NAS benchmarks: FBNet [7] and NAS-Bench-201 [10] and target a diverse set of 7 hardware platforms: Edge GPU, Edge TPU, Raspberry Pi, FPGA (Xilinx ZC706), FPGA (Xilinx ZCU102), Pixel3, and Eyeriss.

In addition, the code and supplementary results are available in the GitHub repository[1].

II. Background and Related Work

This section presents a literature review of related research efforts proposed to model and optimize the performances in the NAS process. First, we present related works that have proposed surrogate models for neural architectures' accuracy and latency predictions. Second, we overview the main concepts needed to understand the learning-to-rank theory and the Multi-Objective Optimization Problem in the NAS process.

A. Surrogate Models in NAS

Several methods have been proposed to use ML models to predict NN architectures' performance in NAS. For NN accuracy prediction, we distinguish two approaches depending on the type of the ML model inputs:

[1]https://github.com/IHIaadj/HW-PR-NAS

1) Architecture Encoding: The more direct method is to use the features extracted from the NN architecture to create an ML model that predicts the accuracy. GATES [11] uses a particular graph-based encoding using a Graph Convolutional Network (GCN) to add precise information about the connections between the NN operators in the architecture. FENAS [6] divides the architecture according to the position of the downsampling operations, and a set of possible operations represents each block. They use the random forest to achieve the regression and predict the accuracy.
2) Learning Curves: Learning curves represent the loss obtained after training the architecture for a few epochs. Loss values are used to extrapolate the accuracy in more epochs. In [12], the authors use the results of training the model for 30 epochs in addition to the architecture encoding and dataset characteristics to score the architectures. The used scoring approach uses a pairwise ranking loss.

For NN latency prediction, most surrogate models use lookup tables (LUT). The LUT maps each operation in the search space to its corresponding pre-calculated latency. Once an NN architecture is sampled, a sum of all its operators' latencies will estimate the end-to-end latency. However, this layer-wise method shows limitations in the prediction accuracy and NAS performance [9], [13]. Therefore, HW-NAS methods often turn to ML models to predict the latency. Proxyless-NAS [1] builds a surrogate model for mobile phones based on: the type of the operators, input and output feature map size, kernel size, stride for convolution, and expansion ratio.

Our approach is the first work that builds a single surrogate model for Pareto front ranking in the NAS process to the best of our knowledge.

B. Learning-to-rank Theory

Learning-to-rank theory [14] is a class of algorithmic techniques used to construct a ranking model from data automatically. Learning to rank has been used extensively to solve information retrieval problems. During the search of a given query, the retrieved documents are ranked based on the scores of the documents provided by the model. There are many algorithms proposed for learning-to-rank. These algorithms learn their ranking functions by minimizing certain loss functions. These algorithms can be categorized into three approaches. The pointwise approach views every single object as a learning instance. Examples include subset regression [15] and McRank [16]. The pairwise approach uses a pair of objects as the learning instance (e.g., Ranking SVM [17] and RankBoost [18]). The listwise approach, as its name implies, uses the entire list of objects as the learning instance, such as ListNet [19] and ListMLE [20]

In [21], the authors used a Pareto Rank learning model to rank the offspring of the multi-objective evolutionary algorithm (MOEA) and enhance its Pareto front approximation. This latter is the closest work to our contribution. However, their model is trained after searching and creating a large dataset. Our surrogate model is trained before the search and is used directly, which helps accelerate the search over a large search space.

C. Multi-Objective Optimization Problem

In a multi-objective optimization, the problem is to minimize a set of objective functions $f_1(\alpha), ..., f_n(\alpha)$. For instance, in a multi-objective NAS, α refers to a sampled NN architecture, and f_i is a function that gives one of the performance metrics of this architecture (e.g., accuracy, latency, memory and energy consumption). Because of the antagonism of the objectives to optimize simultaneously, there is no single optimal solution to this problem. Instead, we need to find a set of Pareto-optimal solutions; *the Pareto front*. What makes these solutions optimal is that no f_i can be reduced without increasing at least one f_j $(i \neq j)$.

The notion of dominance defines if a solution is in the Pareto front (dominant) or not (non-dominant). If s_1 and s_2 denote two feasible solutions, s_1 dominates s_2 ($s_1 \succ s_2$) if and only if $\forall i\ f_i(s_1) \leq f_i(s_2)$ AND $\exists j\ f_j(s_1) < f_j(s_2)$. Few works have been dedicated to the Pareto front approximation in HW-NAS in the literature. For instance, among multi-objective NAS approaches, LEMONADE [22] uses network morphism to enhance a Multi-objective Evolutionary Algorithm (MOEA) and finds the Pareto optimal architectures. However, they use the number of parameters of the NN architectures as a hardware parameter that only characterizes the memory size and does not correlate well with other metrics such as latency.

More recently, [23] proposes a set of NAS+HPO multi-objective algorithms using the accuracy and number of parameters. To the best of our knowledge, no other surrogate model tries to combine all objectives into a predicted score and thus efficiently search for the Pareto front.

III. HW-PR-NAS: Hardware-Aware Pareto Rank-preserving NAS

This section describes the HW-PR-NAS system. We first explain the new Pareto ranking loss and how it is used during the training. Then, we detail HW-PR-NAS components as shown in figure 3. Finally, we describe a new flexible version of the surrogate model that allows adding more hardware metrics without incurring a considerable re-training cost.

A. HW-PR-NAS: Pareto Ranking Loss

HW-PR-NAS ranks the NN architectures from the most dominant to the non-dominant ones. In a multi-objective NAS problem, the set of N architecture solutions $S = s_1, s_2, ..., s_N$ may be sorted by their Pareto front rank $F = F_1, ..., F_K$ where K is the number of Pareto fronts we can have by successively solving the problem for $S - \bigcup_{s_i \in F_k \wedge k < K}$, i.e., the top dominant architectures are removed from the search space each time. Intuitively, we can classify a solution according to how many points it dominates, i.e., architectures that dominate all the solutions are in F_1, architectures that dominate all the solutions except those in F_1 are in F_2, and so on. This means that if architecture is in F_1, there's no other architecture

that can further minimize the latency without increasing the accuracy.

Theoretically, the sorting is done by the following conditions:

$$\forall s_i, s_j \in F_k,\ s_i \not\succ s_j \wedge\ s_j \not\succ s_i \tag{1}$$

$$\forall s_i \in F_{k+1}\ \forall s_j \in F_k,\ s_i \not\succ s_j \tag{2}$$

$$\forall s_i \in F_{k+1}\ \exists s_j \in F_k,\ s_j \succ s_i \tag{3}$$

Equation 1 formulates that for all the architectures with the same Pareto rank where no one dominates another. Equation 2 formulates that any architecture with a pareto rank $k+1$ cannot dominate all architecture with a pareto rank k. Equation 3 formulates each architecture with a Pareto rank $k+1$ there exists at least one architecture with a Pareto rank k dominates it.

We first consider a HW-NAS with two objectives: the accuracy and latency of a model. We define dominance according to these two objectives. Our surrogate model learns to rank the architectures according to this Pareto rank definition. Our surrogate model is not a classification model as this would require building and maintaining an extensive dataset. We use a regression model instead. Since we need to create the different Pareto sets for different sets of solutions, our surrogate model is trained to give high scores to the dominant points via a listwise loss function.

We compute the different Pareto ranks before computing the loss between the scores and the actual ranks when we make a batch. We design the listwise ranking loss following the ListMLE approach [20] as follows:

$$L(B) = \sum_{i=1}^{|B|}\{-f(a^{(i),B)}) + log \sum_{j=i}^{|B|} exp(f(a^{(j),B)})\} \tag{4}$$

Equation 4 represents our listwise loss. B denotes a batch, while $|B|$ denotes its batch size. $a^{(i),B}$ denotes the architecture whose Pareto is ranked i-th in the subset B. f denotes the output function of the surrogate model. We can have two architectures with the same Pareto rank. Using this loss function, the architectures within the same Pareto front will have a similar score, which helps us extract the final Pareto approximation.

Algorithm 1 explains how a multi-objective evolutionary algorithm (MOAE) is tweaked to use HW-PR-NAS. In a classical MOAE, we generate an initial population, either randomly or using a sampling strategy. The population then proceeds with crossover and mutation operators. The newly generated solution, known as offspring, undergoes fitness evaluations using exact measurements or single objective surrogate models. A population is formed using the fitness values, merging the parents and the offspring using a selection algorithm. The Pareto rank is used as one of the criteria in the selection algorithm. This process continues until the computational budget or specified termination criteria are reached. Unlike prior research on multi-objective search with surrogates, the core novelty of our work lies in the construction of a surrogate model to rank the architectures according to their dominance. Subsequently, HW-PR-NAS is used to assess the architectures while searching. The predictions are used to select the offspring with higher scores, i.e., the architectures with higher scores are more likely to be in the actual Pareto front. The selection algorithm is an elitist operator. The final size of the Pareto front k is equal to the population size.

Algorithm 1: Multi-Objective Evolutionary Algorithm with HW-PR-NAS

```
Set t = 0;
An initial population P_t is randomly generated and
  then evaluated using HW-PR-NAS;
while Stopping Criteria is not reached do
    Perform crossover and mutation to create new
      population Q_t from P_t;
    Evaluate Q_t using HW-PR-NAS model;
    P_{t+1} = P_t ⋃ Q_t;
    t = t + 1;
end
return Top_k ∈ P_t ;
```

B. HW-PR-NAS: Model Design

The high-level overview of HW-PR-NAS is illustrated in Figure 3. The system is composed of two components. (1) **Architecture Encoding** which encodes the architecture into a vector of numbers. (2) Predictors constituted of two predictors — one for accuracy and another for latency. Each predictor gets a different architecture encoding as input to better extract the features needed for each predicted metric. The accuracy and latency predictors are detailed in section III-D and III-E.

The outputs of the two models are then passed to a dense layer that sums the two results and outputs one single score per DL architecture on a specific HW platform. This score is correlated to the Pareto rank described in section III-A. In addition to our Pareto ranking loss, we adjust each model with the square root of the variance of the residuals (RMSE) according to its prediction metric to achieve faster training.

C. Architecture Encoding

HW-PR-NAS is composed of two different predictor models. Each predictor needs a different encoding scheme and ML algorithm. We conducted different tests considering the state of the art models in the literature and our system's need to find the best combination for the latency and accuracy predictors. For each experiment, we use the hinge loss with a margin of 0.1, inspired by GATES [11] to train the models to rank the architectures. We compare the final performance of the models using Kendal tau correlation, which is a ranking correlation metric. [2]

[2]We completed a series of tests with the RMSE only, but our new multi-objective loss performs better with the ranking scores.

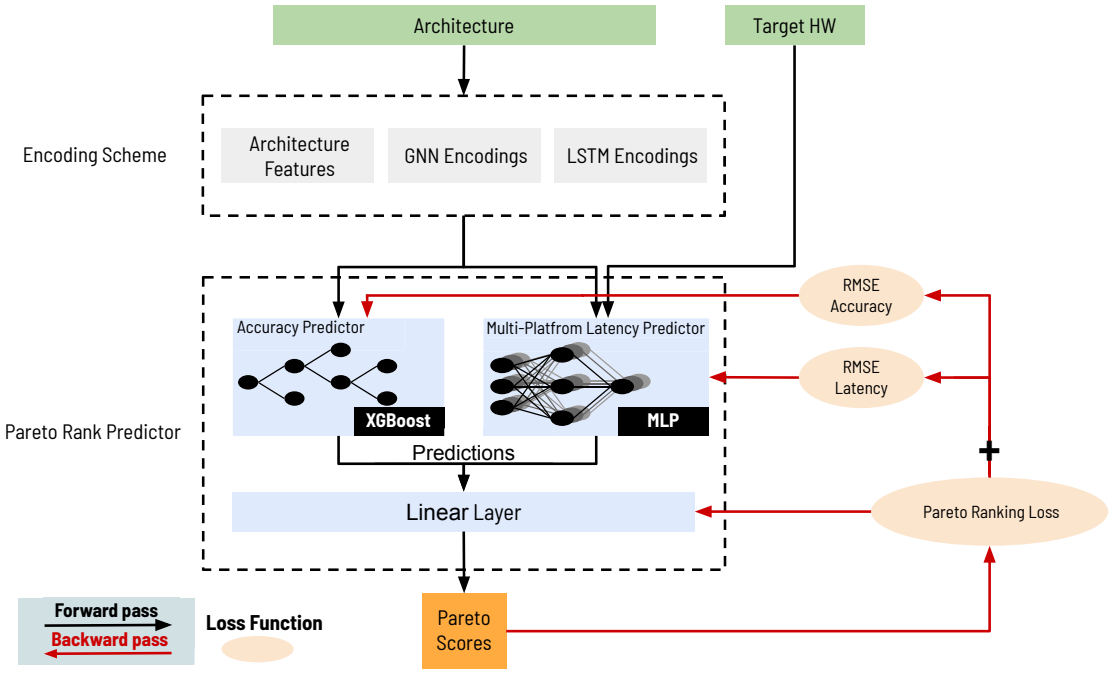

Fig. 3. General Overview of HW-PR-NAS

To decide the best encoding scheme, we fix the regressor to be an MLP model for both the latency and accuracy predictions. We then vary the encoding schemes through an ablation study to help assess which method extracts more valuable features.

We extract the features from the architectures in three different ways:

1) **Architecture Features:** First, we measure the number of FLOPs, the number of parameters, the number of convolutions, the input size, the architecture's depth, and the first and last channel size and the number of down-sampling. We refer to these manually extracted features as Architecture Features (AF).
2) **LSTM Encoding:** Second, we use an LSTM encoding. We first encode the string version of the architecture provided by NAS-Bench-201 [10] into a vector through layer embedding. This string "*—nor_conv_3x3 0—nor_conv_3x3 1—*" refers to a sequential link between two 3x3 convolutions. Subsequently, we apply a 2-layers LSTM with 225 hidden units. For FBNet architectures [7], we encode them into the same string format and apply the same model.
3) **GNN Encoding:** Lastly, we use a 2-layers Graph Convolution Neural Network (GCN) with 600 hidden units each to generate the encoding. The input architecture is encoded into its adjacency matrix and operation vector. Following [9], we also add a global node to aggregate all node-level information.

To decide the best ML algorithm to use, we start with the best encoding we found in the encoding scheme study for each performance metric and evaluate the regressors. We intentionally separate the encoding schemes from the regressor models to test multiple combinations. The training hyperparameters were selected and tuned using an exhaustive grid search. We evaluated three models for the ML algorithm used in each surrogate model: MLP, XGBoost and LGBoost. For XGBoost and LGBoost, the architecture encoding is first passed through a dense layer and then concatenated with the architecture features (AF).

D. Accuracy Predictor

Accuracy predictors are sensible to the types of operators and connections in a DL architecture.

Figure 4 shows the results we obtain after training the accuracy and latency predictors with different encoding schemes on NAS-Bench-201 [3].

Using only the architecture features (AF), we observe a small correlation (0.63) between the selected features and the accuracy, resulting in poor predictions performance. The best predictor is obtained using a combination of GCN encodings, which encodes the connections, node operation, and architectural features. Additionally, we notice that the Kendal tau correlation when using LSTM encoding and GCN is too small in FBNet compared to the difference in NAS-Bench-201 for

[3]For complementary results on FBNet check: https://github.com/IHIaadj/HW-PR-NAS

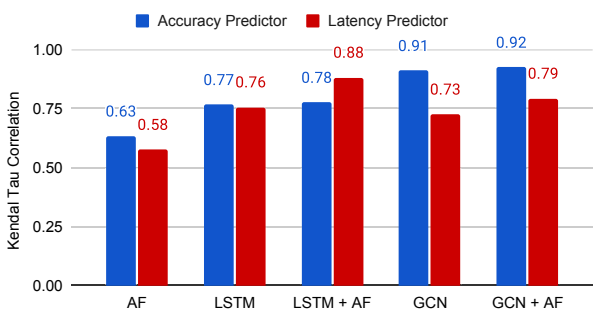

Fig. 4. Results of different encoding schemes for accuracy and latency predictions using NAS-Bench-201. AF refers to Architecture Features.

accuracy prediction. The reason for this difference is the use of operators such as zeroize or skip connect in NAS-Bench-201. These operators modify the connections between the operator in a cell. However, in FBNet, the connections are fixed. This has motivated us to choose GCN encoding for HW-PR-NAS, which better encodes the connections within a graph and thus generalizes to different benchmarks.

	Accuracy		Latency	
	RMSE	Kendal Tau Correlation	RMSE	Kendal Tau Correlation
MLP	4.88	0.924	3.238	**0.8817**
XGBoost	3.12	**0.931**	3.216	0.8742
LGBoost	3.58	0.864	3.058	0.8247

TABLE I
RESULTS OF DIFFERENT REGRESSORS ON NAS-BENCH-201

Table I shows the results for accuracy and latency regressor algorithms. XGBoost gives the best ranking correlation and RMSE for accuracy predictions. The table shows how the ranking correlation is not correlated to how close the predicted accuracy is to the real one (RMSE).

E. Multi-platform Latency Predictor

We conducted the same experiments on the latency predictor. For the encoding scheme, figure 4 shows that the LSTM encoding is better suited for latency prediction. For both benchmarks, NAS-Bench-201 and FBNet, the LSTM encoding outperforms the GCN encoding. Additionally, the architecture features significantly improve the overall performance. For the regressor component, table I shows that MLP is slightly better than XGBoost.

We studied the correlation between the different platforms' latencies to justify the difficulty of creating a once-for-all platform model. In general, we notice weak correlations. Even two different platforms of the same family, FPGA (ZC706) and FPGA (ZCU102), don't correlate (0.23). However, the three platforms, Raspberry4, Pixel3 and FPGA(ZC706), give high correlated values. This suggests that we can consider them as a family, and we can assume that the latency monotonocity [24] is respected between these platforms. We further investigate these correlations on CIFAR-100 and ImageNet datasets and notice that the three platforms' latencies are not correlated anymore when modifying the input size. This encourages us to create a multi-platform latency predictor by duplicating the

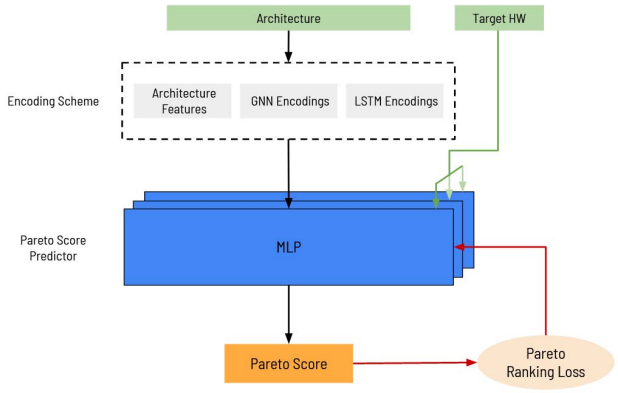

Fig. 5. Scalable version of HW-PR-NAS model.

MLP regressor. The HW platform identifier (Target HW in figure 3) is used as an index to point to the corresponding weights in the multi-platform latency predictor.

F. Generalization to more objectives

To allow users to add more metrics and objectives. We present a scalable version of our surrogate model. In this section, scalability is the ability to add more objectives with different types.

The characteristics of this version are as follows. (See figure 5 for more details). (1) The architecture encodings are a concatenation of the three vectors: architectural features, GNN encoding and LSTM encoding. The encoding has better coverage and represents every important architecture feature. The above study, in section III-E, provides evidence of the necessity to concatenate the three encodings. This encoding scheme can represent any architecture implemented as a DAG in PyTorch. This enables the model to be used with any search space. (2) One multi-layer perceptron (MLP) replaces the predictor. This MLP outputs the Pareto score of the architecture directly without predicting the objectives. Figure 9 illustrates the results of using the model with three objectives: accuracy, latency and energy consumption on CIFAR10. We fine-tune only the MLP part of the model for five epochs. The encoding component was frozen (not fine-tuned), and we used the exact encoding for each architecture.

IV. EXPERIMENTS

We start by presenting our experimental setup in Section IV-C and describe the comparison algorithms and their hyper-parameters in Section IV-C1. Finally, we discuss the results in Section IV-D.

A. Training Details

We train our surrogate model 5 times, each time using a randomly sampled set of 4000 models from both NAS-Bench-201 [10] and FBNet [7]. We used 1000 models for validation and the remaining architectures for testing. Note that all the

Epochs	80 (Early stopping at 30 epochs)
Initial Learning rate	0.0003
Learning rate schedule	Cosine Annealing
Batch_size	128
Optimizer	AdamW
L2 Weight Decay	0.0003
Dropout Ratio	0.02

TABLE II
TRAINING HYPERPARAMETERS OF HW-PR-NAS.

components of HW-PR-NAS are trained simultaneously. We further train the last dense layer one last time to achieve an optimal Pareto ranking. During the training, each component is trained with its RMSE, in addition to the Pareto loss. We can view HW-PR-NAS as a multi-branch model, where each branch has its task. Training the surrogate model took 3 GPU hours. Table II presents the different training hyperparameters used to train HW-PR-NAS.

B. Dataset

HW-PR-NAS training dataset consists of 1k architectures and their respective performances on CIFAR-10, CIFAR-100 and ImageNet120 sampled from two different benchmarks: NAS-Bench-201 [10], FBNet [7]. In the literature, two different benchmarks propose to extend NAS-Bench-201 with hardware metrics, namely HW-NAS-Bench [25] and BRP-NAS [9]. HW-NAS-Bench has also incorporated the FBNet [7] search space. We use this benchmark to get the dataset needed to train and test our surrogate model.

NAS-Bench-201 explores 15k architectures that provide a large enough space to compare our search algorithms and train surrogate models. On the other hand, FBNet has two critical features. First, it eliminates the cell-based architecture (i.e., the same cell is not repeated along the macro-architecture), which is more hardware-friendly. Second, it uses crucial operators such as depth-wise convolutions and grouped convolutions. Therefore, using both benchmarks gives us better coverage and allows us to get insights into which architectures are suitable for the different hardware platforms.

C. Experimental Setup

Our experiments are run on NVIDIA RTX 6000 GPU with 24G memory. We first target latency and accuracy as the objectives of our multi-objective optimization. Our search space is composed of NAS-Bench-201 and FBNet in which architectures are trained on CIFAR-10, CIFAR-100 and ImageNet16-120.

1) Comparison Algorithms: The search algorithms used in this paper are as follows.

Random Search. We randomly select an architecture from NAS-Bench-201 or FBNet.

MOAE. A standard multi-objective evolutionary algorithm with tournament parent selection. The population size, maximum generations and mutation rate have been set respectively to 150, 250 and 0.9. The stopping criteria is defined as maximum generation of 250 and time budget as 24hours.

2) Existing Surrogate Models: We compare HW-PR-NAS to two state of the art surrogate models:

GATES [11] We use GATES GCN-predictor to compare our multi-objective algorithm against existing surrogate models. We train their architectures on our search space. Note that, we fix the architecture proposed by the paper, but rerun a training hyperparameters sweep to find the optimal values for our dataset.

BRP-NAS. [9] This algorithm offers latency and accuracy surrogate models. We compare our search algorithms using these surrogates. We also compare their oracle NAS on NAS-Bench-201.

D. End-to-End Results

To gauge the efficiency and quality of HW-PR-NAS, we compare the different Pareto front approximations to the existing methods. As we can compute the real Pareto front on NAS-Bench-201, we first validate our method on this benchmark. Figure 6 presents the different Pareto front approximations using HW-PR-NAS and MOAE with BRP-NAS surrogate models and the optimal Pareto front when combining the results of 5 runs. We notice that our approach consistently gets closer to the Pareto optimal front on different platforms and different datasets than MOAE with BRP-NAS.

Table III summarizes the obtained hypervolume metrics of the final Pareto front for each method on both benchmarks NAS-Bench-201 and FBNet simultaneously. The hypervolume indicator encodes which Pareto front approximation is favoured by measuring objective function values' coverage. The hypervolume calculates the area dominated by the provided set of solutions with respect to a reference point. We use the implementation offered by [26], and we use the furthest point from the Pareto front as a reference point in each situation. The larger the hypervolume indicator corresponds, the better the Pareto front approximation is, and thus the better are the corresponding architectures. We notice that the results vary significantly across seeds when using two surrogate models. However, using HW-PR-NAS, we can have a descent standard error while covering a high area (hypervolume). This motivates further the use of a single model that looks for the dominant points in the search space.

We also compare, in figure 7, the speed of the search algorithms using different surrogate models. The time budget is 24hrs maximum. By calling our surrogate model, the search time is better than using a surrogate model for each metric. This is due to the shared call and decreased the number of comparisons to find the dominant points.

a) Analysis of the final Pareto front: We analyze the proportion of each benchmark on the final Pareto front for different hardware platforms in table IV. The results confirm the intuitive hypothesis concerning the proportions of the benchmarks. Indeed, in Pixel3 (mobile phone), we expect more architectures from FBNet, as this benchmark uses specialized convolutions, depthwise convolutions, in their macro-architecture. The depthwise convolution decreases the model's

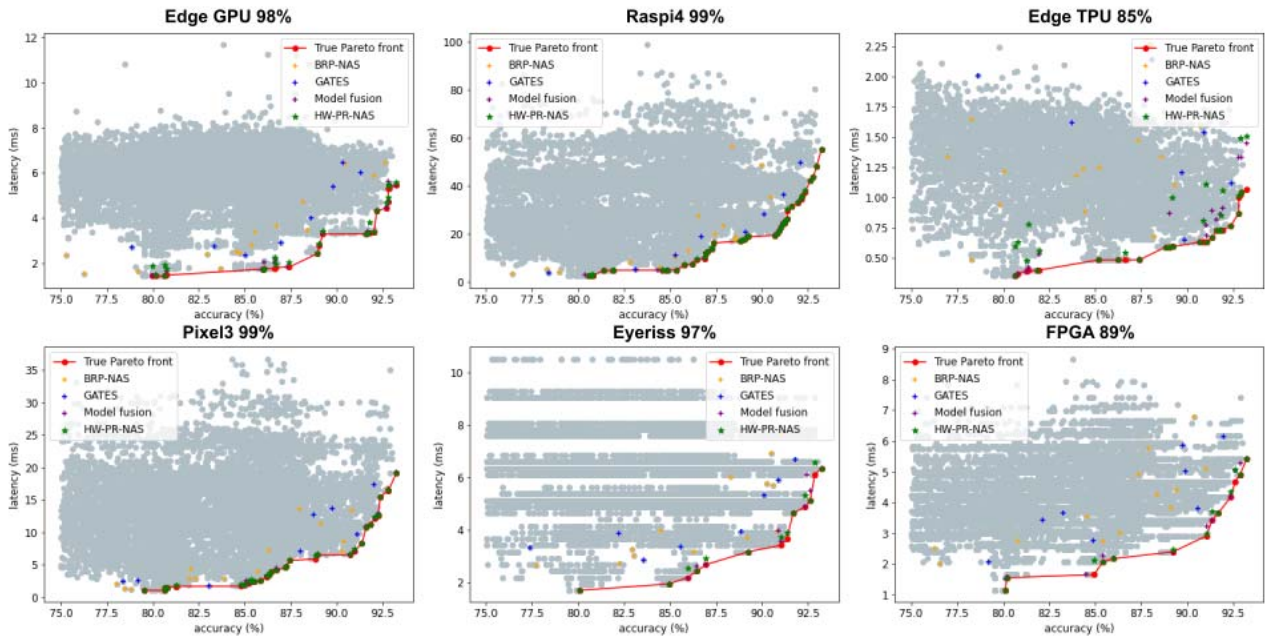

Fig. 6. Pareto front approximations on CIFAR-10 on edge hardware platforms. We show the true accuracies and latencies of the different architectures, and the normalized hypervolume on each target platform

	Hypervolume (mean ±std. error) ↑		
	CIFAR-10	CIFAR-100	ImageNet
Random Search (Measured Values)	542.30 ±2.37	480.34 ±2.24	465.21 ±2.85
Random Search (BRP-NAS)	520.12 ±2.99	529.65 ±3.68	-
Random Search (GATES)	563.94 ±3.59	459.67 ±3.20	490.35 ±3.56
Random Search (HW-PR-NAS)	**569.18 ±1.45**	**539.31 ±1.59**	**499.57 ±2.56**
MOAE (Measured Values)	523.67 ±2.49	512.67 ±2.13	486.32 ±2.54
MOAE (BRP-NAS)	584.32 ±3.37	546.38 ±3.65	-
MOAE (GATES)	534.59 ±3.58	511.39 ±3.21	479.36 ±3.68
MOAE (HW-PR-NAS)	**571.24 ±1.28**	**578.25 ±1.69**	**530.25 ±1.57**

TABLE III
FINAL HYPERVOLUME OBTAINED BY EACH METHOD ON THE THREE DATASETS. WE SHOW THE MEANS ± STANDARD ERRORS BASED ON 5 INDEPENDENT RUNS.

size and achieves faster and more accurate predictions. However, depthwise convolutions do not benefit from the GPU and FPGA optimizations and parallelizations compared to the standard convolutions used in NAS-Bench-201, which have a higher proportion in the Pareto front of these platforms.

Figure 8 illustrates the difference between the obtained architectures. We only showcase the architectures with the least latency in Pareto front approximations targeting Edge GPU and Pixel 3 platforms. The depthwise convolution available

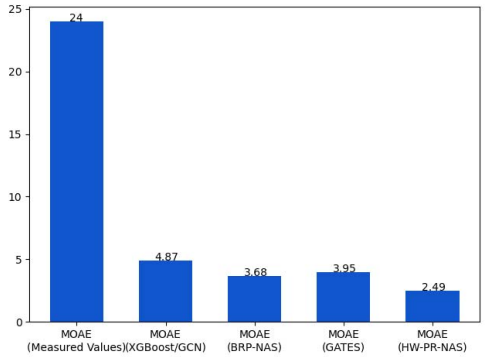

Fig. 7. Search time of MOAE using different surrogate models on 250 generations with a max time of 24 hrs

	EdgeGPU	EdgeTPU	FPGA	Pixel3
NAS-Bench-201	58.33	61.91	54.54	20
FBNet	41.67	38.09	45.46	80

TABLE IV
PROPORTION (IN PERCENTAGE) OF NAS-BENCH-201 AND FBNET IN THE FINAL PARETO FRONTS.

in FBNet is suitable for architectures on mobile devices. This operation allows fast execution without accuracy degradation. However, on edge GPU as the platform has more resources (4G memory), it prefers a bigger model from NAS-Bench-201 with higher accuracy.

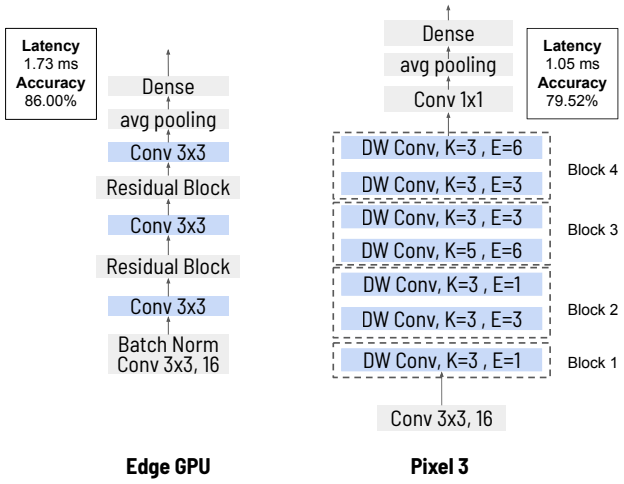

Fig. 8. Illustration of the final architectures obtained in the Pareto front of Edge GPU and Mobile pixel 3 respectively for CIFAR 10.

b) Energy Consumption: Our system can also work on any number of objectives. We need to add a model that predicts the desired objectives. The system aims to find fast and energy-efficient models by adding energy consumption. Architectures that can preserve the battery lifetime of the system are very important on edge devices. The Pareto front is of utmost significance in this situation. It allows the application to select the right architecture according to the system's hardware requirements. Figure 9 shows the results of the Pareto front on different architectures of NAS-Bench-201.

V. Conclusion

HW-NAS is a critical emerging area of research that allows the automatic synthesis of optimised neural network architectures that can run efficiently on constrained hardware platforms such as edge devices. However, HW-NAS exacerbates the computational problem that NAS already has. This paper proposes an improved surrogate model strategy to accelerate the search computation problem of HW-NAS while preserving the quality of the search results. Our experimentation results show that our Pareto rank surrogate model delivers a better Pareto front approximation and 2x speedup in search time using the NAS-Bench-201 and FBNet benchmarks on several hardware platforms.

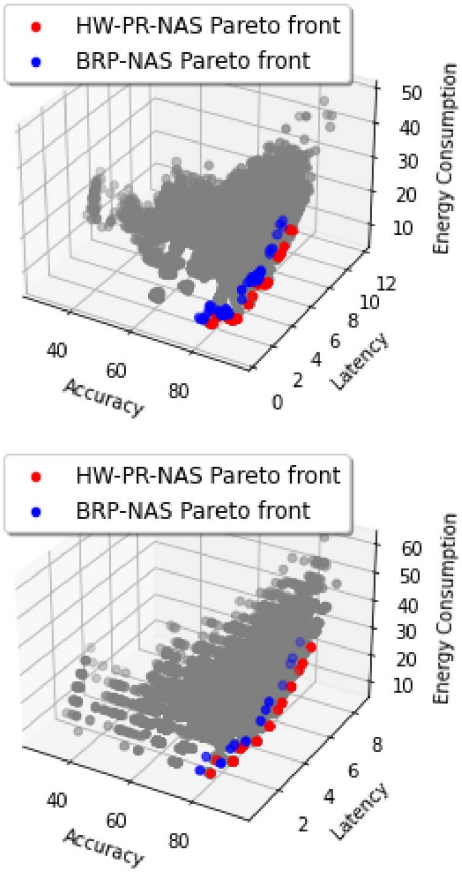

Fig. 9. Pareto front Approximations using three objectives: accuracy, latency and energy consumption on CIFAR-10 on Edge GPU.

References

[1] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019.

[2] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.

[3] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 2820–2828.

[4] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2020.

[5] J. Mellor, J. Turner, A. J. Storkey, and E. J. Crowley, "Neural architecture search without training," *CoRR*, vol. abs/2006.04647, 2020. [Online]. Available: https://arxiv.org/abs/2006.04647

[6] R. Shi, J. Luo, and Q. Liu, "Fast evolutionary neural architecture search based on bayesian surrogate model," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 1217–1224.

[7] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 10 734–10 742.

[8] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "HAT: hardware-aware transformers for efficient natural language processing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 7675–7688. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.686

[9] L. Dudziak, T. C. P. Chau, M. S. Abdelfattah, R. Lee, H. Kim, and N. D. Lane, "BRP-NAS: prediction-based NAS using gcns," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[10] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of repro-

ducible neural architecture search," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[11] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A generic graph-based neural architecture encoding scheme for predictor-based NAS," in *Computer Vision - ECCV - 16th European Conference*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12358. Springer, 2020, pp. 189–204.

[12] M. Wistuba and T. Pedapati, "Learning to rank learning curves," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 303–10 312.

[13] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *CoRR*, vol. abs/2101.09336, 2021. [Online]. Available: https://arxiv.org/abs/2101.09336

[14] C. He, C. Wang, Y.-X. Zhong, and R.-F. Li, "A survey on learning to rank," in *International Conference on Machine Learning and Cybernetics*, vol. 3, 2008, pp. 1734–1739.

[15] D. Cossock and T. Zhang, "Statistical analysis of bayes optimal subset ranking," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 5140–5154, 2008.

[16] P. Li, Q. Wu, and C. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2008.

[17] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*, P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola, Eds. MIT Press, 2000, pp. 115–132.

[18] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, no. null, p. 933–969, dec 2003.

[19] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 129–136. [Online]. Available: https://doi.org/10.1145/1273496.1273513

[20] Y. Lan, Y. Zhu, J. Guo, S. Niu, and X. Cheng, "Position-aware listmle: A sequential learning process for ranking," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI*, N. L. Zhang and J. Tian, Eds. AUAI Press, 2014, pp. 449–458.

[21] C.-W. Seah, Y.-S. Ong, I. W. Tsang, and S. Jiang, "Pareto rank learning in multi-objective evolutionary algorithms," in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[22] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019.

[23] S. Izquierdo, J. Guerrero-Viu, S. Hauns, G. Miotto, S. Schrodi, A. Biedenkapp, T. Elsken, D. Deng, M. Lindauer, and F. Hutter, "Bag of baselines for multi-objective joint neural architecture search and hyperparameter optimization," in *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. [Online]. Available: https://openreview.net/forum?id=yEGlj93aLFY

[24] B. Lu, J. Yang, W. Jiang, Y. Shi, and S. Ren, "One proxy device is enough for hardware-aware neural architecture search," *CoRR*, vol. abs/2111.01203, 2021. [Online]. Available: https://arxiv.org/abs/2111.01203

[25] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, and Y. Lin, "{HW}-{nas}-bench: Hardware-aware neural architecture search benchmark," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=_0kaDkv3dVf

[26] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.