MPENAS: Multi-fidelity Predictor-guided Evolutionary Neural Architecture Search with Zero-cost Proxies

Jinglue Xu
The University of Tokyo
Tokyo, Japan
jingluexu@gmail.com

Suryanarayanan NAV
The University of Tokyo
Tokyo, Japan
nav-surya@g.ecc.u-tokyo.ac.jp

Hitoshi Iba
The University of Tokyo
Tokyo, Japan
iba@iba.t.u-tokyo.ac.jp

ABSTRACT

Neural architecture search (NAS) aims to automatically design suitable architectures of artificial neural networks (ANNs) under various situations. Recently, NAS based on zero-cost proxies can predict the performance of ANNs with the cost of a single forward/backward propagation pass at most. While zero-cost proxies can speed up NAS by orders of magnitude, the gap between the predicted and actual performance of ANNs prevents zero-cost proxies from identifying ANNs with top performance.

One solution is to regard zero-cost proxies as a low-fidelity evaluation method and switch from zero-cost proxies to high-fidelity evaluation methods when the zero-cost proxies struggle at selecting architectures. Based on this idea, we propose Multi-fidelity Predictor-guided Evolutionary Neural Architecture Search (MPENAS). MPENAS is based on a novel surrogate-assisted evolutionary computation framework. With a predictor, MPENAS combines architecture encodings, zero-cost proxies, learning curve extrapolations, and fully trained ANNs' performance into one consistent fitness across different fidelity.

To our knowledge, MPENAS is the first work that integrates zero-cost proxies into a multi-fidelity optimization framework. MPENAS outperforms ten other methods for the NAS-Bench-201 search space in all cases. In addition, we demonstrate the generalizability of MPENAS for the TransNAS-Bench-101 search space.

CCS CONCEPTS

• **Computing methodologies → Neural networks; Search with partial observations**.

KEYWORDS

Neural Architecture Search, Zero-cost Proxy, Multi-fidelity Optimization, Surrogate-assisted Evolutionary Computation

ACM Reference Format:
Jinglue Xu, Suryanarayanan NAV, and Hitoshi Iba. 2023. MPENAS: Multi-fidelity Predictor-guided Evolutionary Neural Architecture Search with Zero-cost Proxies. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3583131.3590513

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590513

1 INTRODUCTION

Artificial neural networks (ANNs) attract significant attention in recent years, with strong performance across multiple domains, such as computer vision, natural language processing, and audio signal processing. Neural architecture search (NAS) aims to automate the process of designing and/or training ANNs. NAS relies on evaluations of a large number of ANNs to identify ANNs with high performance.

Since the publication of NASWOT [37], multiple NAS methods based on zero-cost (ZC) proxies have been proposed to analyze the architectures and predict the performance of ANNs (nearly) without training the ANNs. A ZC proxy is a calculation that estimates an ANN's actual performance. Typically, the cost of a ZC proxy is no more than a single forward/backward propagation pass on a minibatch of data, which is almost zero compared to conventional NAS methods [50]. For example, the number of parameters in an ANN can be a naive but effective ZC proxy [5, 38, 48]. In addition, several ZC proxies, such as neural tangent kernel [20], can explain the mechanisms behind the performance of ANNs, which can offer insights into the traditional black box view of neural architectures [7]. NAS based on ZC proxies significantly accelerates the search process by orders of magnitude while outperforming state-of-the-art methods like DARTS [34], ENAS [39], and GDAS [7, 9].

However, methods based on ZC proxies can be negatively affected by the evaluation gap between the predicted and actual performance of ANNs. Just like weight-sharing NAS methods, a gap exists between the predicted (relative) performance and the actual (relative) performance of the ANNs. While under most situations the correlation is positive, a survey across 13 ZC proxies reveals that the Spearman rank correlation between the values from ZC proxies and the validation accuracies rarely exceeds 0.8 [30]. The negative effect of this gap becomes obvious during the differentiation among a small subset of ANNs with top performance, which is typically critical during the later stages of the search process.

These issues contribute to a lack of ability to identify ANNs with top performance, especially compared to traditional methods that involve evaluations of fully trained ANNs, like REINFORCE [57] and BOHB [10, 15, 47]. Not surprisingly, ZC proxies share similar pros and cons with weight-sharing methods: high speed with compromised performance. Despite the weaknesses, ZC proxies are a highly promising direction in NAS. Compared to weight-sharing methods, ZC proxies are usually much faster. In terms of the evaluation gap, the comparison is limited in the current literature, although one study suggests the gap in ZC proxies is similar or smaller, compared to that of weight-sharing methods [1].

Is there a method to reach the high performance of traditional NAS methods while leveraging the low cost of ZC proxies? One

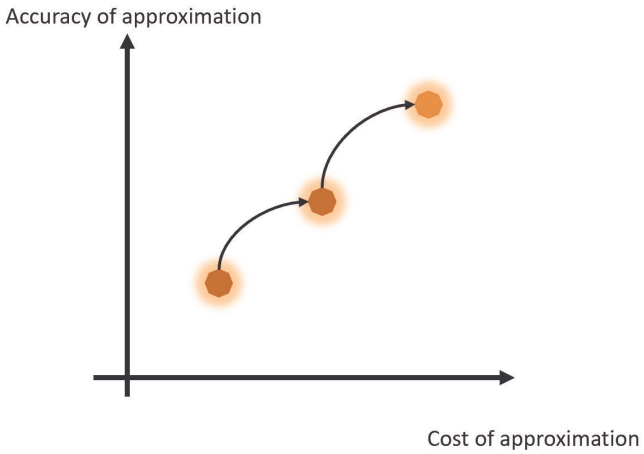


Figure 1: Multi-fidelity optimization. The black arrow represents the switch from one fidelity to another fidelity

solution is multi-fidelity optimization, as shown in Figure 1. Mutli-fidelity optimization trades off between the cost (e.g., computational time, memory consumption, or the number of data points) and fidelity (e.g., the correlation between predicted and actual fitness) of evaluation methods to maximize optimization efficiency. During the early stages of the search process, ZC proxies can be an extremely efficient low-fidelity evaluation method in terms of identifying a subset of ANNs with relatively high performance. Once the subset is identified, the search algorithm switches to evaluation methods with higher fidelity to accurately capture the small differences among the performance of the ANNs.

In evolutionary computation (EC), expensive optimization problems (EOPs) refer to optimization problems that require computationally expensive simulation or calculation to evaluate candidate solutions [56]. For EOPs, one approach is multi-fidelity fitness approximation. In this approach, the fitness evaluation process is replaced by multiple evaluation methods with different costs and fidelity [32]. Generally speaking, among the different evaluation methods, the larger the cost of the method, the higher the fidelity. The key to the approach is efficient approximations and a suitable strategy that adjusts fidelity. In this paper, we present Multi-fidelity Predictor-guided Evolutionary Neural Architecture Search (MPENAS). MPENAS is based on a novel surrogate-assisted EC framework.

The main contributions of this paper include:

- **Integration of ZC proxies into multi-fidelity optimization.** To our knowledge, MPENAS is the first work that integrates ZC proxies into a multi-fidelity optimization framework.
- **A novel multi-fidelity fitness approximation method for EOPs.** While we haven't shown the effectiveness of the method in a general case, the method suits the case of the integration of ZC proxies into multi-fidelity NAS. We provide a theoretical rationale behind the design of the method. In addition, we demonstrate the effectiveness of the method for the case of MPENAS experimentally.
- **High performance across various settings.** For the NAS-Bench-201 [10] search space, MPENAS outperforms ten other popular NAS methods across 3 categories. MPENAS also shows high generalizability in TransNAS-Bench-101 [12].

2 RELATED WORK

2.1 NAS

The earliest NAS method dates back to the 1990s [28]. With the recent popularity of deep neural networks and AutoML [29], NAS became a hot topic in recent years. The novelty of MPENAS is mainly about multi-fidelity optimization and ZC proxies.

Multi-fidelity optimization: Notable multi-fidelity optimization methods for NAS include Successive Halving [21] and Hyperband [33]. Successive Halving first allocates a chosen amount of resources for evaluating each individual in a chosen population. In each successive iteration, the number of individuals decreases and the resource for each individual increases, both in a predetermined geometrical distribution. Hyperband combines multiple Successive Halving processes with different initiations. Both methods do not require the exact fidelity of each evaluation method and iterate over all preset evaluation methods. This feature contributes to the wide applicability of the two methods. However, if (estimates of) the exact fidelity of each evaluation method are available, then a multi-fidelity optimization can utilize such information to improve the optimization efficiency. MPENAS adaptively controls the fidelity based on available testing labels for the surrogates. Although multi-fidelity optimizations can achieve benefits for NAS when simply combined with random search, most NAS methods based on multi-fidelity optimization integrate multi-fidelity optimization with a general optimization framework. For example, BOHB combines Hyperband with Bayesian optimization, and DEHB [2] combines Hyperband with differential evolution [45]. The surrogate in DEHB uses one type of information (performance of ANNs trained at different epochs) to select individuals from one subpopulation to another. By comparison, the surrogates in MPENAS approximate fitness evaluation within a single population via multiple types of information (architecture encodings, ZC proxies, learning curves at different epochs, and performance of fully trained ANNs)

ZC proxies: ZC proxies are a relatively new approach in NAS. Because of the low cost, ZC proxies are readily available in most NAS situations and most NAS methods based on ZC proxies include a combination between ZC proxies and other components. TENAS [7] integrates zero-cost proxies (neural tangent kernel and the number of linear regions in the input space [53]) into weight-sharing NAS methods. [1] directly predicts ANN performance using a ZC proxy (*synflow* [1]). In addition, [1] also examines the combination between ZC proxies and other components such as general optimization frameworks (aging evolution [52] and reinforcement learning [57]) and a predictor (graph convolutional network [13]), suggesting clear benefits in the combinations. [37] directly predicts ANN performance based on a ZC proxy (*naswot* [37]) and also shows benefits in the combination between the ZC proxy and a general optimization framework (regularised evolutionary search [39]). Recently, [30] explores adding information from ZC proxies as input features of predictors (XGboost [6] and BANANAS [49]), and [50] uses information from ZC proxies and learning curves

as additional features for predictors (NGboost [11] and SemiNAS [36]). Both works suggest a strong benefit in the added features for predictors. Based on multiple studies that suggest the benefit of combining ZC proxies with additional components and the existing issues in ZC proxies (as mentioned in Section 1), MPENAS is a step forward that integrates ZC proxy in a multi-fidelity optimization framework.

2.2 EC for EOPs

In EC, EOPs heavily overlap with surrogate-assisted optimization problems [24], large-scale optimization problems [22], and data-driven optimization problems [27]. NAS is usually an EOP as the NAS process without speedup can take more than 3000 GPU days [8]. There are several different approaches for solving EOPs in EC. EC based on fitness approximation predicts the actual fitness of each individual using cheaper substitutes. The first EC based on fitness approximation dates back to the mid-1980s [17]. Surrogate-assisted EC can be used for fitness approximation. In surrogate-assisted EC, typically some cheap machine learning methods or simulations are used as surrogates to replace the original fitness evaluation. In MPENAS, the instances of a predictor (i.e., a supervised machine learning method) are the surrogates.

In terms of the overall structure of the optimization, surrogates for fitness evaluations can be individual-based, generation-based, and population-based [23]. In individual-based methods, in each generation, a subset of individuals is evaluated with the original fitness evaluation method. MPENAS is an individual-based method. Individual-based methods could be more suited than the other two for implementations on a single machine or multiple machines with identical specifications [24]. Examples of individual-based methods include [25] and [26].

Some surrogate-assisted ECs rely on a single surrogate, and some have multiple surrogates. [16] uses a single surrogate with architecture encoding as input for NAS. For methods based on multiple surrogates, each surrogate can represent a single fidelity and constitute a multi-fidelity optimization. Multiple studies suggest the effectiveness of increasing the fidelity as the search process proceeds in surrogate-assisted EC [14, 18], which suits the case of solving the issue related to the evaluation gap of ZC proxies. In terms of the relations among the multiple surrogates, some techniques, such as [26], use an ensemble of similar surrogates. Other techniques, such as [43], use a combination of different categories of surrogates. MPENAS is based on multiple surrogates across different categories.

From the perspective of EC, a similar work to our method is SAFE [51]. Both MPENAS and SAFE are based on the idea of adaptively switching to higher fidelity in the later stages of the search process. The key differences between MPENAS and SAFE are: (1) For SAFE the switch criterion is based on the best individual that could be found at a given fidelity and the alignment among different fidelity. For MPENAS the criterion is based on an analysis of errors in fitness approximation. (2) In SAFE, the individuals in each generation are evaluated with a single adaptively chosen fidelity. While MPENAS can include multiple adaptively chosen fidelity. (3) SAFE is tested in the case of crowd shipping scheduling, while MPENAS is tested in the case of NAS.

3 METHODS

As shown in Figure 2, roughly speaking, MPENAS contains (1) An initialization phase that generates training data for the surrogates and warmstarts the evolution process by pre-selection. (2) A multi-fidelity optimization phase that approximates fitness evaluation based on the surrogates. (3) An evolution phase that executes standard EC operations according to the approximated fitness. Phases (2) and (3) are executed alternatively until termination. From a NAS perspective, MPENAS contains multiple elements and each element in MPENAS can be a standalone NAS method. The motivation behind MPENAS is to solve the existing issues with ZC proxies and improve NAS based on ZC proxies. We describe MPENAS by explaining the benefits of adding one element at a time, starting from ZC proxies.

3.1 Multi-fidelity from multiple surrogates

Because information from ZC proxies is obtained before finishing the 1_{st} epoch of the training, a natural idea of creating surrogates with higher fidelity is to increase the number of epochs. Most existing multi-fidelity NAS methods (without ZC proxies) also set fidelity according to the number of epochs trained [2, 10]. By increasing the number of epochs, a piece of readily available information is from learning curve extrapolations. The learning curve of an ANN is the change in the ANN's performance in some aspects over time during training or validation. Similar to ZC proxies, the learning curve of partially trained ANNs correlates with the validation accuracy of fully trained ANNs [3, 41]. Therefore, we add information from learning curve extrapolations for surrogates with higher fidelity.

For higher fidelity, instead of using only learning curve extrapolations, we combine learning curve extrapolations with ZC proxies, because of the low cost of ZC proxies and the benefit of extra information. While a simple voting combination between ZC proxies and learning curve extrapolations can be effective, predictors (i.e., supervised machine learning methods) are much more reliable in terms of combining information from the two categories, as demonstrated by [50] and [30]. In addition, predictors are also proved to be effective in improving the accuracy of ZC proxies with only ZC proxies as input features [30]. Therefore, we use predictors as surrogates. For NAS methods based on predictors, architecture encodings are often used as input features [35, 44]. Because architecture encodings are readily available, we also add architecture encodings as input features.

In MPENAS, multiple instances of a predictor with different numbers of input features are trained from past evaluation results. For all instances, the labels are the actual performance of fully trained ANNs (the two dark green blocks and black dashed arrows in Figure 2). Each instance corresponds to a particular fidelity, and the higher the fidelity, the higher the number of input features. We combine architecture encodings and multiple ZC proxies as the input of the predictor at the lowest fidelity. For fidelity between the highest and lowest fidelity, we combine architecture encodings, ZC proxies, and learning curve extrapolations at different training epochs as the predictor input. We use the actual performance of fully trained ANNs (without predictor) as the highest fidelity. The multiple instances of the predictor connect the evaluation methods

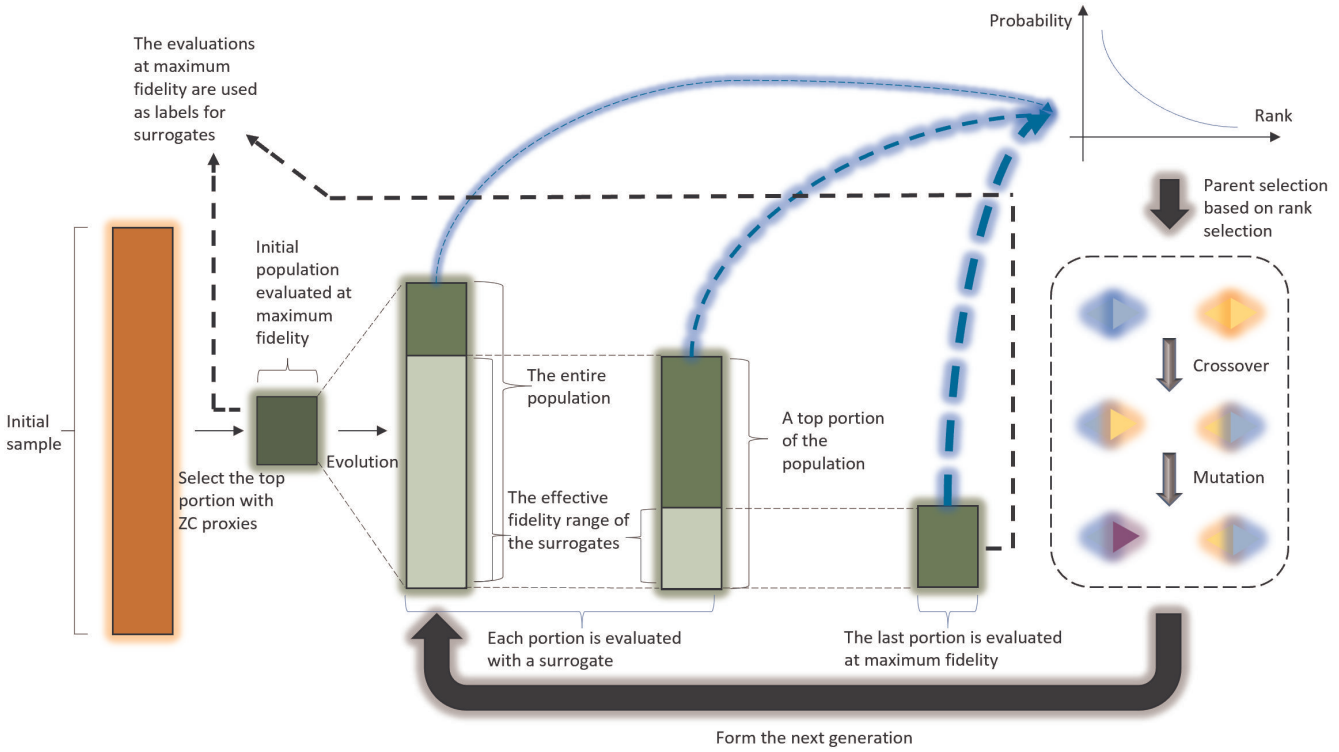


Figure 2: Overview of MPENAS. The thin black dashed lines represent equivalence in the number of individuals. The blue dashed arrows represent selection as parents. The thicker the blue arrow is, the higher the probability of being selected

at different fidelity into a consistent multi-fidelity fitness approximation. Switching between surrogates is achieved by adjusting the number of input features in the predictor.

3.2 Adaptive control of fidelity

With multiple fidelity, the next step is to design a suitable strategy to adaptively switch fidelity. As surrogates, the supervised learning methods themselves can be evaluated with testing labels available during the search process. Therefore, a good adaptive strategy can be based on the approximation error of the surrogates. A strict analysis of the approximation error in surrogate-assisted EC is very difficult if not impossible because the distribution of the errors is usually neither Gaussian nor uniform [24]. However, we can still analyze the errors to some extent based on simplifications.

The approximation error can be absolute or relative. For the predictors, the absolute error is the error in the prediction of the validation accuracy of each of the ANNs. We assume that the absolute approximation error for an individual is independent of the actual rank of the individual. Under this assumption, for different n, the mean absolute error (MAE) of the prediction for the individual with n_{th} actual performance from a surrogate across a large number of trials tends to be the same. In other words, for a surrogate, a single value of MAE approximates the MAEs for individuals with different actual ranks.

As shown in Figure 3, imaging a population of 10 individuals (the dots) evaluated by two surrogates. The MAE of surrogate B is much

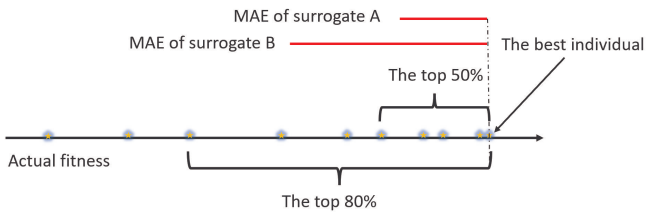


Figure 3: Analysis of the approximation error

smaller than the range between the fitness of the 8_{th} individual and the 3_{rd} individual. For each of the top 3 individuals, especially the top 1 individual, the likelihood of being evaluated by surrogate B as not within the top 80 percent is bounded by a small value (see Appendix A). Therefore, we can evaluate the entire population with surrogate B first, then select the top 80 percent and evaluate the 80 percent with surrogate A. Then we select the top 50 percent and evaluate the portion with a surrogate of higher fidelity. We can repeat the process to narrow down the range. For a surrogate, we define the minimum size of the top portion of the population selected by the surrogate such that the portion likely includes the actual top individuals as the **effective fidelity range** of the surrogate. A formal calculation of the effective fidelity range from MAE is given in Appendix A. Under an appropriately configured

mapping from MAE to effective fidelity range, most of the top individuals stay in the final range with a low total cost. The mapping is independent of the specific distribution of the approximation error, which increases the robustness of our method.

Based on the above rationale, we design an algorithm: Adaptive Multi-fidelity Fitness Approximation (Algorithm 1). This algorithm relies on a mapping from fidelity to effective fidelity range, which can be calculated from the MAE corresponding to each fidelity. Algorithm 1 requires a list of available top portions of the population (L_N) and fidelity (e.g., number of input features in the predictor) ((L_F)) to simplify the process. In Algorithm 1, the size of the effective fidelity range of the surrogate in one iteration ($N_{fidelity}$ in line 10, the light green regions of the blocks in Figure 2) is equal to the size of the portion in the next iteration ($P_{fidelity}$ in line 14, the two green blocks on the right in Figure 2).

The design of Algorithm 1 also has a biological parallel. In nature, an individual often needs to outperform other individuals in multiple rounds of non-lethal selections in order to reproduce. For example, for the population of monkeys in Jigokudani Yaen Koen, Japan, a season with an unusual temperature can render some individuals unhealthy. Out of the healthy individuals, some individuals are able to join a group. For the monkeys in groups, only some of the monkeys can successfully mate. Each round of the selection selects a portion of individuals for the next round (effective fidelity range).

Algorithm 1: Adaptive Multi-fidelity Fitness Approximation

Input : A list of portions of the population L_N, a list of choices of fidelity L_F, a population P, a mapping from fidelity to effective fidelity range R, a mapping from fidelity to trained predictors S

1 $N_{fidelity} \leftarrow \infty$
2 $P_{fidelity} \leftarrow P$
3 $f \leftarrow 0$
4 $evaluation\_results \leftarrow \{\}$
5 **while** $f < max(L_F)$ **do**
6 **if** $N_{fidelity} = min(L_N)$ **then**
7 $f \leftarrow max(L_F)$
8 **else**
9 $f \leftarrow$ Find the smallest f in L_F such that $R[f]$ is smaller than $|P_{fidelity}|$
10 $N_{fidelity} \leftarrow R[f]$
11 **end if**
12 $evaluation\_results \leftarrow$ Predict the performance of each individual in $P_{fidelity}$ using $S[f]$ and increment $evaluation\_results$
13 **if** $|P_{fidelity}| > min(L_N)$ ***and*** $f < max(L_F)$ **then**
14 $P_{fidelity} \leftarrow$ Based on $evaluation\_results$, select the top $N_{fidelity}$ number of individuals in $P_{fidelity}$ to replace $P_{fidelity}$
15 **end if**
16 **end while**
17 **return** $evaluation\_results$

3.3 Navigation and initiation

Although the cost of ZC proxies is low, searching through the entire search space at once for the first iteration in Algorithm 1 (i.e., setting P as the entire search space) is still inefficient or even infeasible in many situations. For example, the DARTS search space [34] contains more than 10^{18} possible ANNs. In MPENAS, we use EC (the dashed block on the right in Figure 2) as a guide to sample the search space.

Formally, we describe MPENAS in Algorithm 2. MPENAS warmstarts the evolution with ZC proxies and evaluations at maximum fidelity. At first, a large number of ANNs are randomly sampled from the search space (the crimson block in Figure 2). Then each of the ANNs is evaluated with ZC proxies to select N number of the best-performing ANNs (the leftmost dark green block in Figure 2). N is equal to the size of the population involved in the evolution processes in MPENAS. This N number of ANNs is re-evaluated with the original fitness function (i.e., validation accuracy of the fully trained networks). Then the set of ANNs evolves into the next generation, using the same evolution operators as used in later iterations (the dashed block on the right in Figure 2).

In addition to warmstarting the evolution process, this process is critical for initializing the predictors (line 6 in Algorithm 2). The predictors and values of effective fidelity range are updated each time new evaluation results are available (unless the evolution should terminate) (line 10 to line 14 in Algorithm 2).

4 EXPERIMENTS

4.1 Search space and datasets

The search space of NAS refers to the possible ANNs (defined by some basic building blocks) that could be searched by the NAS. Popular traditional NAS search spaces include the NAS-RL search space [19], the cell-based (NASNet) search space [58], the DARTS search space, and the MobileNet (MNASNet) search space [46]. These traditional search spaces require training the ANNs during the search process of NAS. The performance of a NAS method tends to be heavily affected by the specific training procedures of the ANNs and data processing, which leads to irreproducible results and lack of fairness [10, 52, 55]. In addition, these traditional tasks, especially when testing traditional NAS methods, tend to require a large number of computational resources, leading to high carbon emissions. To solve the two issues, novel databases that include precomputed results (such as the validation accuracy) for each possible ANN have been proposed. We test MPENAS using the following two precomputed databases:

- `NAS-Bench-201`: With a cell-based search space, the fixed macro skeleton resembles the DARTS search space. Each cell allows 4 nodes, 6 edges, and 5 (edge-associated) operations, totaling 6466 unique ANNs. It includes the training costs as queryable information. It is based on the CIFAR-10 [31], CIFAR-100 [31], and ImageNet-16-120 [42] datasets. And, each ANN is evaluated as an average of up to 3 trials. [10]
- `TransNAS-Bench-101`: TransNAS-Bench-101 is designed to test the generalizability and/or transferability of NAS methods. The database contains a diverse range of tasks including

Algorithm 2: MPENAS

Input : Initial number of individuals to be evaluated with ZC proxies N_0, size of a population N, a list of portions of the population L_N, a list of choices of fidelity L_F

1 $P_0 \leftarrow$ Randomly sample N_0 number of individuals from the search space to form a set of individuals
2 Evaluate each individual in P_0 with ZC proxies evaluations without any predictors
3 $P \leftarrow$ Rank and select the best N number of individuals in P_0 to form a set of individuals
4 *evaluation_results* $\leftarrow$ Re-evaluate the individuals in P with maximum fidelity, starting from the individual with the highest performance to the individual with the lowest performance, and record the results
5 $R \leftarrow$ Estimate the effective fidelity range of predictors for each fidelity based on *evaluation_results*
6 $S \leftarrow$ Train the predictors and save the trained predictors for each fidelity in L_F using the labels in P
7 $P \leftarrow$ Apply evolution operators to P and form a new generation based on *evaluation_results*
8 **while** *termination criteria is not reached* **do**
9 *evaluation_results* $\leftarrow$ Increment *evaluation_results* with *Adaptive Multi-fidelity Fitness Approximation* (L_N, L_F, P, R, S)
10 **if** *termination criteria is not reached* **then**
11 $S \leftarrow$ Update S using the new labels in P
12 $R \leftarrow$ Update R using *evaluation_results*
13 $P \leftarrow$ Apply evolution operators to P and form a new generation based on *evaluation_results*
14 **end if**
15 **end while**
16 *best_individual* $\leftarrow$ Based on *evaluation_results*, rank all individuals that are evaluated with maximum fidelity and select the best individual
17 **return** *best_individual*

object classification, scene classification, semantic segmentation, autoencoding, room layout, surface normal, and jigsaw. The search space considers both the macro skeleton and cell-level configurations. [12]

Although the cost of ZC proxies is low, a single NAS trial based on ZC proxies can still take more than 1 GPU hour [7]. NAS-Bench-Suite-Zero [30] includes the precomputed results of 13 ZC proxies into NAS-Bench-201 and 12 ZC proxies into Trans-NAS-Bench. We use the precomputed results from NAS-Bench-Suite-Zero.

4.2 Implemenation details

For all experiments, we use the following evolution operators from genetic algorithms (1) Exponential rank selection as parent selection with a base probability of 0.9 (2) Uniform crossover with a probability of 0.9 (3) Swap mutation and random resetting mutation of a random number of genes, each with a probability of 0.2. For the predictor, we use NGboost [11], which is suggested by [50] as an excellent predictor for ZC proxies and learning curve extrapolations. For the ZC proxies, we use the 13 ZC proxies [30] for NAS-Bench-201 and 12 proxies for Trans-NAS-Bench implemented in NAS-Bench-Suite-Zero. We refer to [30] for the details regarding each proxy. For ZC proxies evaluations without predictors (line 2 of Algorithm 2), we use the voting strategy as described in [30]. For learning curve extrapolations, we use the SoTL method [41] as suggested by [50] as well as the validation loss, training accuracy, and validation accuracy over each epoch trained so far.

For experiments in Section 4.4, Section 4.5, and Section 4.6, for each surrogate, we map the effective fidelity range from the estimated MAE as described in Section 4.3 using the method described in Appendix A with an λ_{max} of 0.2 and an objective to preserve the top 1 individual.

For NAS-Bench-201, we set L_F as a geometrically distributed list: $[0, 4, 8, 16, 32, 64, 128, 200]$. In the list, 0 represents the surrogate with only ZC proxies and architecture encodings as input features (corresponding to "the 0_{th} epoch"), 200 represents the evaluation of fully trained ANN (each ANN in NAS-Bench-201 is trained with 200 epochs), and each integer f between 0 and 200 in L_F represents the surrogate with features available at the f_{th} epoch. We set N_0, N and L_N as 3000, 100 and $[5, 12, 25, 50, 100]$.

4.3 Homogeneity of appproximation errors

First, we examine the assumption we made in Section 3.2: the absolute approximation error of an individual is independent of the actual rank of the individual. If this assumption holds, then for a generation, the MAE across the individuals with ranks in different intervals should be similar.

With a randomly sampled P in line 3 of Algorithm 2, we divide the population into 10 bins. Each bin represents an interval of the ranks. For each bin, we train 7 surrogates over the fidelity in L_F using 100 training datapoints from another randomly sampled P in line 3 of Algorithm 2. We repeat the experiment 10 times to calculate the MAE across the 10 individuals in each bin for each fidelity across 3 datasets. The results are shown in Table 2, Table 3, and Table 4 in Appendix B.

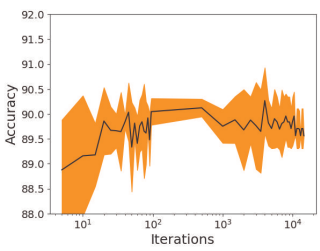
From the results, we can safely assume the homogeneity of the absolute error in fitness approximation over the ranks of individuals. Moreover, for any n, we can estimate the MAE of the n_{th} ranked individual in the population in one generation over multiple trials with the MAE of a number of top individuals in the recent generations in one trial. Because the quality of the surrogates in close generations is similar and both the n_{th} ranked individual in a trial and an individual with top performance are equal to a randomly sampled individual in terms of approximation error.

Therefore, for experiments in Section 4.4, Section 4.5, and Section 4.6, for each fidelity, we estimate the single corresponding MAE by analyzing the results of 50 individuals recently evaluated at maximum fidelity (If an individual is evaluated at maximum fidelity, then all surrogates can make predictions for the individual because the input features are available).

4.4 Performance of ZC proxies

Next, we test the performance of ZC proxies without any other elements in MPENAS. For each dataset in NAS-Bench-201, we randomly sample a number of individuals in the search space and select the individual with the highest predicted performance based on ZC proxies. Then we test the actual performance of the individual.

In Figure 4, Figure 5, and Figure 6, the error bands represent the 95 percent confident interval, and the iterations represent the number of individuals sampled. As shown in the figures, ZC proxies are capable of rapidly identifying an individual with relatively high performance. But after about 10^2 iterations, the efficiency drastically decreases. After about 10^3 iterations, the actual performance of the identified individual starts to decrease, which suggests the discrepancy between the actual top individuals and predicted top individuals from ZC proxies.



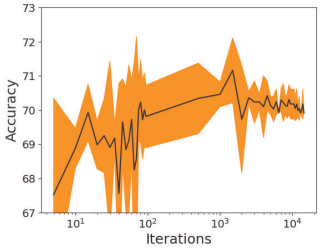


Figure 4: ZC proxies - CIFAR-10 **Figure 5: ZC proxies - CIFAR-100**

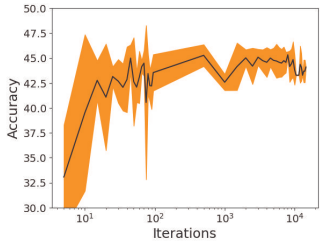


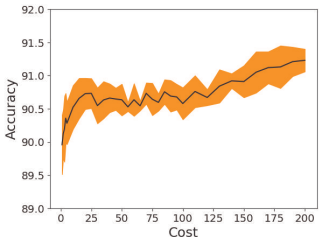
Figure 6: ZC proxies - ImageNet

4.5 Performance of MPENAS

MPENAS is designed to solve the inability of ZC proxies to identify the individuals with top performance in the later stages of NAS. We tested MPENAS with the settings in Section 4.2 on 3 datasets in NAS-Bench-201. The results are shown in Figure 7, Figure 8, and Figure 9. In the three figures, the cost is calculated as the multiple of the average cost of evaluating an individual at maximum fidelity. For example, the cost of an evaluation based on ANNs trained at 64_{th} epoch is roughly 0.32. The accuracy in the figures represents the actual performance of the best ANN identified so far with the cost.

To examine the effectiveness of Algorithm 1, with the same settings and cost, we also tested a version of MPENAS by setting Hyperband as the multi-fidelity optimization method in line 9 of Algorithm 2: MPENAS-HB. In the Hyperband of MPENAS-HB, we set the maximum resource as the average cost of a full evaluation and use the same settings as described in [15].

From the results, we observe that the search process of MPENAS can be roughly divided into three stages. In the first stage (between 0 and 25 cost), the accuracy rapidly increases due to the effectiveness of ZC proxies in the early stages as demonstrated in section 4.4 (line 1 to line 4 in Algorithm 2). In the second stage (between 25 and 100 costs), the gain of accuracy over cost stagnates because of the limitations of individuals identified by ZC proxies (line 2 in Algorithm 2). However, this stage is preparing for the next stage as the quality of the surrogates increases over the increase of available training data. In the third stage (between 100 and 200 cost), the increase of accuracy becomes higher from the surrogate-assisted EC process (line 8 to line 15 in Algorithm 2).



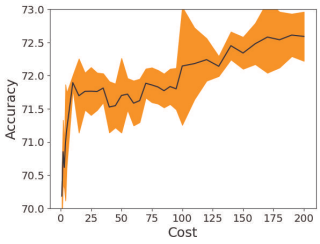


Figure 7: MPENAS - CIFAR-10 **Figure 8: MPENAS - CIFAR-100**

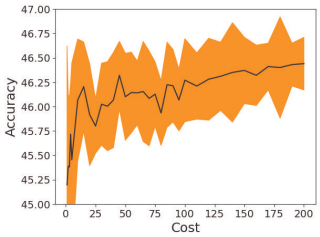


Figure 9: MPENAS - ImageNet

We also compare our method with some state-of-the-art methods using the NAS-Bench-201 search space. The results are based on 10 trials, and we report the mean and standard deviation of the validation accuracy based on the settings described in Section 4.2 with a cost equivalent to 200 full evaluations, as shown in Table 1. The numbers in the brackets represent standard deviation. The entries in the table are categorized into three blocks. The first block includes weight-sharing methods, in which a method finishes in 2 to 10 GPU hours. The second block includes two popular methods based on ZC proxies. NASWOT finishes in seconds and TENAS takes about 0.5 GPU hours. The 3rd block contains two traditional NAS methods which are tested with the same cost as MPENAS. While methods (purely) based on ZC proxies and weight-sharing can find ANNs with good performance with little cost, they are generally unable to identify ANNs with very high performance.

In addition, to test the generalizability of our method, we also compare MPENAS with BOHB and REA on TransNAS-Bench-101 using the settings described in Section 4.2. In the experiments, we use the default settings for BOHB and REA as described in [15] and [40]. For each of the three methods, we use a cost equivalent to 200 full evaluations. We repeat each combination 10 times and

Table 1: Results on NAS-Bench-201

	CIFAR-10	CIFAR-100	ImageNet-16-120
RSPS [4]	84.16(1.69)	45.78(6.33)	31.09(5.65)
DARTS-V1 [34]	39.77(0.00)	38.57(0.00)	18.87(0.00)
DARTS-V2 [34]	39.77(0.00)	38.57(0.00)	18.87(0.00)
GDAS	90.01(0.46)	24.05(8.12)	40.66(0.00)
ENAS [39]	39.77(0.00)	10.23(0.12)	16.43(0.00)
PC-DARTS [54]	89.96(0.15)	67.12(0.39)	40.83(0.08)
TENAS	90.96(0.43)	71.27(0.52)	41.95(0.48)
NASWOT [37]	89.12(1.77)	67.01(3.34)	36.74(5.76)
BOHB	90.91(0.47)	71.03(1.42)	44.65(1.34)
REA [40]	91.20(0.22)	71.93(1.07)	45.13(0.67)
MPENAS-HB	91.02(0.31)	71.86 (0.74)	45.52(0.49)
MPENAS	**91.23(0.27)**	**72.61(0.42)**	**46.44(0.14)**
Optimal	91.61	73.49	46.77

report the mean and standard deviation in Table 5 and Table 6 in Appendix C. The results show that MPENAS outperforms REA and BOHB in most situations.

4.6 Quality of each surrogate

We also examined the change of MAE (as estimated using the method described in Section 4.3) and Kendall Tau's correlation over different fidelity during the search process at the end of the second stage of the search process as described in Section 4.5. In Figure 10 and Figure 11, fidelity represents the surrogate corresponds to an f in L_f as described in Section 4.2. The Kendall Tau's correlation is lower than some studies reported [30, 50]. The reason is that the metrics are evaluated with individuals of high performance (the top 100 out of 3000 individuals as predicted by ZC proxies), and these individuals tend to have very close actual performance, which significantly increases the difficulty. Similar to [30], we also find the benefits of architecture encodings in the surrogates are limited, as removing the encodings decreases Kendall Tau's correlation by only 3.4 ± 0.4% in the worst case.

Interestingly, adding just a small amount of information from learning curve extrapolations can be very effective, particularly for ImageNet-16-120, as the sharp changes between the MAE and Kendal Tau's correlation at 0 and 4 fidelity show. This result suggests benefits from the intermediate fidelity in multi-fidelity optimization, compared to a simple combination between evaluation methods of the highest cost (e.g., fully trained ANNs) and the lowest cost (e.g., ZC proxies and architecture encodings).

5 DISCUSSION

We demonstrate that MPENAS is an effective integration of ZC proxies into multi-fidelity optimization. However, there are still limitations to MPENAS. Although the estimation of effective fidelity range based on the worst case of the distribution of approximation error can be applicable for a wide range of situations, this type of estimation can reduce the efficiency of the optimization when the distribution is far from the worst case. For a specific task, if the specific distribution of the errors is available, then the optimization process can be accelerated. For the case of MPENAS, we assume the distribution is unavailable, which is true in most situations of NAS. Additionally, in some situations, the absolute error of fitness approximation can be highly dependent on the ranks of the individuals, which makes MPENAS unsuitable. Devising methods without this limitation can be a challenging but interesting future direction.

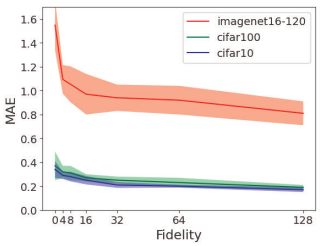


Figure 10: Quality by MAE

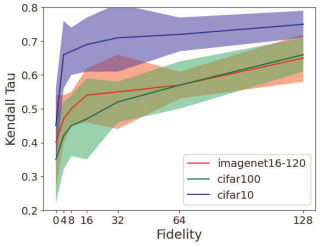


Figure 11: Quality by Kendall Tau

In MPENAS, we combine five common elements for NAS: a general optimization framework (EC), multi-fidelity optimization, predictor, ZC proxies, and learning curve extrapolations. Each one of the elements is capable of performing NAS alone, although most NAS methods involve a combination of multiple elements. Similar to [1, 30, 37, 50], MPENAS suggests that combining ZC proxies with other elements is beneficial. An interesting future direction could be integrating weight-sharing methods into MPENAS. However, such a combination might be difficult, as many existing ZC proxies' performance degrades over training [38, 48].

In theory, Algorithm 1 works for problems beyond MPENAS, as long as the absolute approximation error is independent of the ranks of the individuals. However, testing the general applicability of Algorithm 1 is beyond the scope of this study.

6 CONCLUSION

In this paper, MPENAS integrates ZC proxies into a multi-fidelity optimization framework. To our knowledge, MPENAS is the first work that integrates ZC proxies into multi-fidelity optimization. In the experiments, MPENAS outperforms multiple state-of-the-art NAS methods for the NAS-Bench-201 search space. In addition, We show the high generalizability of MPENAS using the TransNAS-Bench-101 database.

For the integration, we design a novel method for surrogate-assisted EC. In the case of MPENAS, we demonstrate the effectiveness of the method by comparing the method with Hyperband.

For the multi-fidelity optimization, we regard predictions based on architecture encodings and ZC proxies as the lowest fidelity. We add information from learning curve extrapolations for higher fidelity. We show that the effectiveness of architecture encodings in this combination is limited, but the inclusion of a small amount of information from learning curve extrapolations tends to significantly increase the quality of the surrogates.

ACKNOWLEDGMENTS

We thank João Batista for sharing his knowledge about genetic algorithms.

REFERENCES

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. 2021. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134* (2021).

[2] Noor Awad, Neeratyoy Mallik, and Frank Hutter. 2021. Dehb: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization. *arXiv preprint arXiv:2105.09821* (2021).

[3] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823* (2017).

[4] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[5] Hanlin Chen, Ming Lin, Xiuyu Sun, and Hao Li. 2021. Nas-bench-zero: A large scale dataset for understanding zero-shot neural architecture search. (2021).

[6] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. 2021. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535* (2021).

[8] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. 2020. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355* (2020).

[9] Xuanyi Dong and Yi Yang. 2019. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1761–1770.

[10] Xuanyi Dong and Yi Yang. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326* (2020).

[11] Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. 2020. Ngboost: Natural gradient boosting for probabilistic prediction. In *International Conference on Machine Learning*. PMLR, 2690–2700.

[12] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. 2021. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5251–5260.

[13] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. 2020. Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems* 33 (2020), 10480–10490.

[14] Michael TM Emmerich, Kyriakos C Giannakoglou, and Boris Naujoks. 2006. Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 421–439.

[15] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*. PMLR, 1437–1446.

[16] Bryson Greenwood and Tyler McDonnell. 2022. Surrogate-assisted neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1048–1056.

[17] John J Grefenstette and J Michael Fitzpatrick. 2014. Genetic search with approximate function evaluations. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Psychology Press, 112–120.

[18] M Hüscken, Y Jin, and B Sendhoff. 2005. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal* 9, 1 (2005), 21–28.

[19] Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. 2019. Reinforcement learning for neural architecture search: A review. *Image and Vision Computing* 89 (2019), 57–66.

[20] Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* 31 (2018).

[21] Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*. PMLR, 240–248.

[22] Jun-Rong Jian, Zhi-Hui Zhan, and Jun Zhang. 2020. Large-scale evolutionary optimization: a survey and experimental comparative study. *International Journal of Machine Learning and Cybernetics* 11, 3 (2020), 729–745.

[23] Yaochu Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing* 9, 1 (2005), 3–12.

[24] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1, 2 (2011), 61–70.

[25] Yaochu Jin, Markus Olhofer, Bernhard Sendhoff, et al. 2000. On Evolutionary Optimization with Approximate Fitness Functions.. In *Gecco*. 786–793.

[26] Yaochu Jin and Bernhard Sendhoff. 2004. Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*. Springer, 688–699.

[27] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. 2018. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation* 23, 3 (2018), 442–458.

[28] Hiroaki Kitano. 1990. Designing neural networks using genetic algorithms with graph generation system. *Complex systems* 4 (1990), 461–476.

[29] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. 2019. Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA. *Automated machine learning: methods, systems, challenges* (2019), 81–95.

[30] Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. 2022. NAS-Bench-Suite-Zero: Accelerating Research on Zero Cost Proxies. *arXiv preprint arXiv:2210.03230* (2022).

[31] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[32] Jian-Yu Li, Zhi-Hui Zhan, and Jun Zhang. 2022. Evolutionary computation for expensive optimization: A survey. *Machine Intelligence Research* 19, 1 (2022), 3–23.

[33] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18, 1 (2017), 6765–6816.

[34] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[35] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Accuracy prediction with non-neural model for neural architecture search. *arXiv preprint arXiv:2007.04785* (2020).

[36] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Semi-supervised neural architecture search. *Advances in Neural Information Processing Systems* 33 (2020), 10547–10557.

[37] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. 2021. Neural architecture search without training. In *International Conference on Machine Learning*. PMLR, 7588–7598.

[38] Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. 2021. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems* 34 (2021), 12265–12277.

[39] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*. PMLR, 4095–4104.

[40] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[41] Binxin Ru, Clare Lyle, Lisa Schut, Mark van der Wilk, and Yarin Gal. 2020. Revisiting the train loss: an efficient performance estimator for neural architecture search. *stat* 1050 (2020), 8.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115 (2015), 211–252.

[43] Mourad Sefrioui and Jacques Périaux. 2000. A hierarchical genetic algorithm using multiple models for optimization. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*. Springer, 879–888.

[44] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. 2020. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777* (2020).

[45] Rainer Storn and Kenneth Price. 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341.

[46] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2820–2828.

[47] Renbo Tu, Mikhail Khodak, Nicholas Carl Roberts, and Ameet Talwalkar. 2021. Nas-bench-360: Benchmarking diverse tasks for neural architecture search. (2021).

[48] Colin White. 2022. *A Deeper Look at Zero-Cost Proxies for Lightweight NAS*. Retrieved January 15, 2022 from https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/

[49] Colin White, Willie Neiswanger, and Yash Savani. 2021. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10293–10301.

[50] Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. 2021. How powerful are performance predictors in neural architecture search? *Advances in Neural Information Processing Systems* 34 (2021), 28454–28469.

[51] Sheng-Hao Wu, Zhi-Hui Zhan, and Jun Zhang. 2021. SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems. *IEEE Transactions on Evolutionary Computation* 25, 3 (2021), 478–491.

[52] Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Lanfei Wang, Zhengsu Chen, An Xiao, Jianlong Chang, Xiaopeng Zhang, et al. 2021. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing*

Surveys (CSUR) 54, 9 (2021), 1–37.
[53] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. 2020. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 10514–10523.
[54] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2019. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737* (2019).
[55] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. 2019. NAS evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522* (2019).
[56] Zhi-Hui Zhan, Lin Shi, Kay Chen Tan, and Jun Zhang. 2022. A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review* (2022), 1–52.
[57] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).
[58] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.

A FORMAL CALCULATION OF EFFECTIVE FIDELITY RANGE

In Algorithm 1, in any generation, the effective fidelity range of a surrogate needs to be given before employing the surrogate. Therefore, we estimate the relative effective ranges based on results from the previous generation. Given a sorted (from the highest to the lowest) list of the predicted performance of the P individuals in the previous generation:

$$\{\hat{y}_n\}, n = 1, 2, 3, \ldots, P \tag{1}$$

For each $\hat{y}_n$:

$$\hat{y}_n + \epsilon_n = y_n \tag{2}$$

In Equation (2), ϵ_n is the error from the fitness approximation. And y_n is the actual performance. Similarly, let the current generation be a sorted list:

$$\{\hat{y}_n^{'}\}, n = 1, 2, 3, \ldots, P \tag{3}$$

For each $\hat{y}_n^{'}$:

$$\hat{y}_n^{'} + \epsilon_n^{'} = y_n^{'} \tag{4}$$

Let T be the number of trials. Based on the assumption in section 3.2., for any n:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{T} |\epsilon_n^{'t}|}{T} \approx \lim_{T\to\infty} \frac{\sum_{t=1}^{T} |\epsilon_n^{t}|}{T} \approx MAE \tag{5}$$

, where MAE can be the measured MAE of any ϵ_n across multiple (but limited) trials.

If the individual with an actual performance of $\hat{y}_A^{'}$ is predicted to have a performance less than or equal to $\hat{y}_B^{'}$ ($\hat{y}_B^{'} < \hat{y}_A^{'}$), then:

$$\epsilon_A^{'} \geq \hat{y}_A^{'} - \hat{y}_B^{'} \tag{6}$$

Let

$$\delta = \hat{y}_A - \hat{y}_B - \hat{y}_A^{'} - \hat{y}_B^{'} \tag{7}$$

Then:

$$\epsilon_A^{'t} \geq \hat{y}_A - \hat{y}_B - \delta \tag{8}$$

Suppose among the trials, in a portion (λ) of the trials, Equation (8) holds:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{\lambda T} (|\epsilon_A^{'t}|) + \sum_{t=\lambda T}^{T} (|\epsilon_A^{'t}|)}{T} = MAE \tag{9}$$

,where $\epsilon_A^{'t} \geq \hat{y}_A - \hat{y}_B - \delta, t = 1, 2, 3, \ldots, \lambda$.
When

$$\begin{cases} \epsilon_A^{'t} = \hat{y}_A - \hat{y}_B - \delta, t = 1, 2, 3, \ldots, \lambda T \\ \epsilon_A^{'t} = 0, t = \lambda T + 1, \lambda T + 2, \lambda T + 3, \ldots, T \end{cases} \tag{10}$$

, the distribution of the error is in the worst case, λ is maximized, and Equation (9) becomes:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{\lambda_{max} T} (|\hat{y}_A - \hat{y}_B - \delta^t|)}{T} = MAE \tag{11}$$

In most situations:

$$\delta^t \ll \hat{y}_A - \hat{y}_B \tag{12}$$

Therefore:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{\lambda_{max} T} (|\hat{y}_A - \hat{y}_B - \delta^t|)}{T} \approx \lim_{T\to\infty} \frac{\sum_{t=1}^{\lambda_{max} T} (|\hat{y}_A - \hat{y}_B|)}{T} \tag{13}$$

$$\lim_{T\to\infty} \frac{\lambda_{max} T (|\hat{y}_A - \hat{y}_B|)}{T} = MAE \tag{14}$$

$$\lambda_{max} = \frac{MAE}{\hat{y}_A - \hat{y}_B} \tag{15}$$

From the above calculation, the probability that the individual with an actual performance of $\hat{y}_A^{'}$ is predicted to have a performance less than or equal to $\hat{y}_B^{'}$ is estimated to be bounded by $\frac{MAE}{\hat{y}_A - \hat{y}_B}$.

Let's assume that the actual performance of the n_{th} (in terms of actual performance) individual in the current generation $\approx \hat{y}_n^{'}$. Then, to preserve the n_{th} (in terms of actual performance) individual, given the upper bound of the probability (of losing the individual) λ_{max} and MAE, the effective fidelity range can be estimated as the number of individuals with predicted performance between $\hat{y}_n$ and $\hat{y}_n - \frac{MAE}{\lambda_{max}}$ in the previous generation. The estimation avoids underestimation of the effective fidelity range for individuals with high actual performance, even if the assumption fails. If the actual performance is higher than $\hat{y}_n^{'}$, then the upper bound of the likelihood of losing the individual is actually lower than λ_{max}.

B MAE DISTRIBUTION OVER THE RANKS

For all tables in this section, the numbers in the brackets represent standard deviation.

Table 2: MAE for Cifar10

	0	4	8	16	32	64	128
1_{st} to 10_{th}	0.32(0.02)	0.30(0.04)	0.28(0.02)	0.25(0.03)	0.20(0.03)	0.20(0.03)	0.17 (0.02)
11_{th} to 20_{th}	0.34(0.03)	0.31(0.03)	0.29(0.03)	0.26(0.02)	0.22(0.03)	0.21 (0.01)	0.17 (0.02)
21_{th} to 30_{th}	0.34(0.04)	0.29(0.02)	0.29(0.02)	0.24(0.02)	0.22(0.02)	0.21(0.02)	0.19 (0.03)
31_{th} to 40_{th}	0.33(0.02)	0.29(0.03)	0.27(0.02)	0.24(0.03)	0.21(0.04)	0.18 (0.03)	0.17 (0.01)
41_{th} to 50_{th}	0.35(0.04)	0.28(0.02)	0.29(0.03)	0.26(0.02)	0.23(0.04)	0.20(0.03)	0.15 (0.03)
51_{th} to 60_{th}	0.34(0.03)	0.29(0.03)	0.28(0.04)	0.24(0.04)	0.21(0.03)	0.19(0.02)	0.19 (0.02)
61_{th} to 70_{th}	0.36(0.03)	0.28(0.02)	0.28(0.03)	0.25(0.03)	0.20(0.02)	0.19(0.03)	0.16 (0.02)
71_{th} to 80_{th}	0.34(0.02)	0.27(0.03)	0.29(0.03)	0.23(0.03)	0.21(0.03)	0.20(0.02)	0.18 (0.03)
81_{th} to 90_{th}	0.34(0.03)	0.30(0.02)	0.30(0.03)	0.26(0.02)	0.22(0.02)	0.21(0.03)	0.17 (0.02)
91_{th} to 100_{th}	0.35(0.02)	0.29(0.03)	0.28(0.02)	0.24(0.02)	0.21(0.03)	0.20 (0.03)	0.18 (0.02)

Table 3: MAE for Cifar100

	0	4	8	16	32	64	128
1_{st} to 10_{th}	0.41(0.04)	0.32(0.02)	0.31(0.03)	0.29(0.03)	0.25(0.04)	0.24(0.02)	0.20 (0.03)
11_{th} to 20_{th}	0.37(0.02)	0.33(0.02)	0.31(0.04)	0.27(0.02)	0.24(0.02)	0.23 (0.04)	0.18 (0.04)
21_{th} to 30_{th}	0.36(0.02)	0.33(0.03)	0.29(0.03)	0.27(0.03)	0.25(0.04)	0.24(0.03)	0.19 (0.01)
31_{th} to 40_{th}	0.40(0.03)	0.31(0.02)	0.30(0.02)	0.25(0.04)	0.24(0.02)	0.22 (0.03)	0.20 (0.02)
41_{th} to 50_{th}	0.39(0.04)	0.29(0.02)	0.29(0.02)	0.27(0.02)	0.26(0.04)	0.23(0.04)	0.19 (0.02)
51_{th} to 60_{th}	0.38(0.03)	0.31(0.02)	0.31(0.03)	0.28(0.04)	0.26(0.03)	0.21(0.02)	0.19 (0.03)
61_{th} to 70_{th}	0.38(0.02)	0.31(0.03)	0.33(0.04)	0.27(0.02)	0.23(0.03)	0.25(0.02)	0.18 (0.02)
71_{th} to 80_{th}	0.37(0.04)	0.30(0.04)	0.30(0.02)	0.26(0.03)	0.25(0.03)	0.23(0.04)	0.20 (0.03)
81_{th} to 90_{th}	0.36(0.03)	0.34(0.02)	0.28(0.02)	0.27(0.02)	0.26(0.04)	0.24 (0.03)	0.19 (0.03)
91_{th} to 100_{th}	0.41(0.03)	0.31(0.02)	0.31(0.03)	0.25(0.03)	0.24(0.03)	0.22 (0.04)	0.18 (0.02)

Table 4: MAE for ImageNet

	0	4	8	16	32	64	128
1_{st} to 10_{th}	1.59(0.11)	1.11(0.08)	1.07(0.08)	0.91(0.13)	0.94(0.10)	0.89(0.07)	0.81 (0.10)
11_{th} to 20_{th}	1.51(0.09)	1.05(0.07)	1.03(0.09)	0.97(0.10)	0.89(0.09)	0.94 (0.09)	0.88 (0.10)
21_{th} to 30_{th}	1.54(0.06)	1.08(0.13)	0.98(0.13)	0.98(0.07)	0.90(0.12)	0.90(0.06)	0.76 (0.09)
31_{th} to 40_{th}	1.54(0.07)	1.07(0.10)	1.00(0.07)	0.98(0.09)	0.93(0.07)	0.91 (0.10)	0.79 (0.06)
41_{th} to 50_{th}	1.49(0.10)	1.09(0.09)	1.04(0.06)	0.97(0.08)	0.99(0.06)	0.92(0.09)	0.80 (0.08)
51_{th} to 60_{th}	1.50(0.07)	1.15(0.08)	1.03(0.09)	1.00(0.11)	0.97(0.08)	0.93(0.10)	0.81 (0.11)
61_{th} to 70_{th}	1.54(0.06)	1.13(0.11)	1.09(0.08)	1.01(0.09)	0.96(0.09)	0.92(0.07)	0.84 (0.09)
71_{th} to 80_{th}	1.55(0.12)	1.02(0.12)	1.07(0.08)	0.97(0.08)	0.94(0.08)	0.94(0.08)	0.82 (0.07)
81_{th} to 90_{th}	1.57(0.08)	1.07(0.11)	1.10(0.10)	0.98(0.10)	0.94(0.09)	0.89 (0.08)	0.83 (0.08)
91_{th} to 100_{th}	1.59(0.10)	1.09(0.09)	1.05(0.11)	0.99(0.09)	0.95(0.07)	0.90 (0.07)	0.80 (0.09)

C RESULTS ON TRANSNAS-BENCH-101

For all tables in this section, the numbers in the brackets represent standard deviation.

Table 5: Results on TransNAS-Bench-101-Micro

	Cls. Object	**Cls. Scene**	**Autoencoding**	**Surf Normal**	**Sem. Segment**	**Room Layout**	**Jigsaw**
Metric	*Acc.*	*Acc.*	*SSIM*	*SSIM*	*mloU*	*L2 loss*	*Acc.*
BOHB	44.06(0.34)	54.64(0.15)	56.14(0.65)	57.96(0.54)	24.93(0.39)	60.92(0.74)	94.37(0.42)
REA	44.97(0.66)	**54.87(0.03)**	56.02(0.70)	58.03(0.75)	24.75(0.42)	61.09(0.63)	94.75(0.25)
MPENAS (ours)	**45.24(0.53)**	54.77(0.09)	**56.82(0.42)**	**58.14(0.68)**	**25.38(0.37)**	**60.69(0.98)**	**94.93(0.20)**
Optimal	46.32	54.94	57.72	59.62	26.27	59.38	95.37

Table 6: Results on TransNAS-Bench-101-Macro

	Cls. Object	**Cls. Scene**	**Autoencoding**	**Surf Normal**	**Sem. Segment**	**Room Layout**	**Jigsaw**
Metric	*Acc.*	*Acc.*	*SSIM*	*SSIM*	*mloU*	*L2 loss*	*Acc.*
BOHB	45.60(0.41)	56.28(0.51)	72.86(3.49)	60.93(0.95)	27.48(0.63)	59.75(0.92)	96.49(0.27)
REA	46.95(0.34)	56.21(0.35)	72.14(3.07)	61.07(1.09)	28.18(0.48)	60.57(1.10)	**96.80(0.19)**
MPENAS (ours)	**47.23(0.29)**	**56.49(0.60)**	**73.26(2.37)**	**61.54(1.10)**	**28.70(0.33)**	**58.74(1.07)**	96.67(0.19)
Optimal	47.96	57.48	76.88	64.35	29.66	56.28	97.02