# NAS-HPO-Bench-II: A Benchmark Dataset on Joint Optimization of Convolutional Neural Network Architecture and Training Hyperparameters

**Yoichi Hirose** HIROSE-YOUICHI-KC@YNU.JP
**Nozomu Yoshinari** YOSHINARI-NOZOMU-RY@YNU.JP
**Shinichi Shirakawa** SHIRAKAWA-SHINICHI-BG@YNU.AC.JP
*Yokohama National University, Kanagawa, Japan*

**Editors:** Vineeth N Balasubramanian and Ivor Tsang

## Abstract

The benchmark datasets for neural architecture search (NAS) have been developed to alleviate the computationally expensive evaluation process and ensure a fair comparison. Recent NAS benchmarks only focus on architecture optimization, although the training hyperparameters affect the obtained model performances. Building the benchmark dataset for joint optimization of architecture and training hyperparameters is essential to further NAS research. The existing NAS-HPO-Bench is a benchmark for joint optimization, but it does not consider the network connectivity design as done in modern NAS algorithms. This paper introduces the first benchmark dataset for joint optimization of network connections and training hyperparameters, which we call NAS-HPO-Bench-II. We collect the performance data of 4K cell-based convolutional neural network architectures trained on the CIFAR-10 dataset with different learning rate and batch size settings, resulting in the data of 192K configurations. The dataset includes the exact data for 12 epoch training. We further build the surrogate model predicting the accuracies after 200 epoch training to provide the performance data of longer training epoch. By analyzing NAS-HPO-Bench-II, we confirm the dependency between architecture and training hyperparameters and the necessity of joint optimization. Finally, we demonstrate the benchmarking of the baseline optimization algorithms using NAS-HPO-Bench-II.

**Keywords:** Neural Architecture Search, Hyperparameter Optimization, Automated Machine Learning, Convolutional Neural Network

## 1. Introduction

While deep learning has succeeded in various practical tasks such as image recognition (Krizhevsky et al., 2012; He et al., 2016; Huang et al., 2017) and machine translation (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017), deep neural architectures and learning techniques have many tunable hyperparameters. In such a situation, it is difficult and time-consuming to tune machine learning systems manually. Thus, automated machine learning (AutoML), optimizing components of a machine learning system, is recognized as an important research topic in the machine learning community (Hutter et al., 2019).

A field of optimizing neural network architectures, called neural architecture search (NAS), is an active research topic in AutoML (Elsken et al., 2019). It has succeeded in reducing human efforts to design architectures and has discovered state-of-the-art architec-

tures (Tan and Le, 2019; Tan et al., 2019; So et al., 2019). General NAS methods repeat the following steps: (1) sampling candidate architectures from a search space (candidate architecture pool) based on a specific strategy, (2) training the architectures using training data, and (3) evaluating the trained models using validation data. Because this procedure is the same as naive hyperparameter optimization, we can employ a lot of black-box optimizers, such as evolutionary algorithms (Real et al., 2017; Suganuma et al., 2017; Real et al., 2019), reinforcement learning (Zoph and Le, 2017; Baker et al., 2017), and Bayesian optimization (Kandasamy et al., 2018; White et al., 2021), for architecture search.

Most NAS algorithms have the fixed training hyperparameters, but they affect the performance of the architecture obtained by NAS (Yang et al., 2020). Thus, developing joint optimization methods of architecture and training hyperparameters is an important direction in the NAS community to enhance the performance of the obtained architectures. Most simple way of joint optimization is to apply NAS methods with small modification because joint optimization can be also viewed as black-box optimization. However, the most joint optimization methods require much computational time due to the repetition of model training, while an exceptional, efficient joint optimization method (Dong et al., 2021b) exists. Also, different methods use different configurations such as search space or data augmentation. It makes harder to compare joint optimization algorithms under fair conditions.

To tackle the issues and accelerate the research of joint optimization, we produce the first benchmark dataset for joint optimization of convolutional neural network architecure and training hyperparameters. Our benchmark dataset, termed NAS-HPO-Bench-II, can evaluate joint optimization algorithms under the same conditions at low cost. The benchmark dataset adopts the product space of modern cell-based architecture search space and space of learning rates and batch sizes as hyperparameter space. The cell-based architecture space contains 4K candidate architectures. Additionally, we examine eight learning rates and six batch sizes as the training hyperparameters. We train the models for 12 epochs three times each using the CIFAR-10 dataset (Krizhevsky, 2009) and collect the performance data of 192K configurations. Also, we train 4.8K configurations for 200 epochs and build a surrogate model to predict the performance after 200 epoch training from an architecture and training-hyperparameter configuration. As a result, we provide the exact performance dataset of 192K architectures and training hyperparameters for 12 epoch training and the surrogate model-based performance dataset after 200 epoch training. We also analyze the dataset to understand the relationship between architectures and training hyperparameters. We show that the best-performed training hyperparameters differ depending on the architecture. The result supports the importance of joint optimization of architecture and training hyperparameters. Finally, we demonstrate the benchmarking of well-known NAS and hyperparameter optimization algorithms using our proposed dataset.

The contribution of this paper is to provide the first benchmark dataset for joint optimization of network connections and training hyperparameters. Figure 1 illustrates the overview of NAS-HPO-Bench-II. Our proposed dataset, NAS-HPO-Bench-II, will contribute to accelerating the research on joint optimization of architecture and training hyperparameters.[1]

1. The dataset of NAS-HPO-Bench-II is publicly available at https://github.com/yoichii/nashpobench2api.

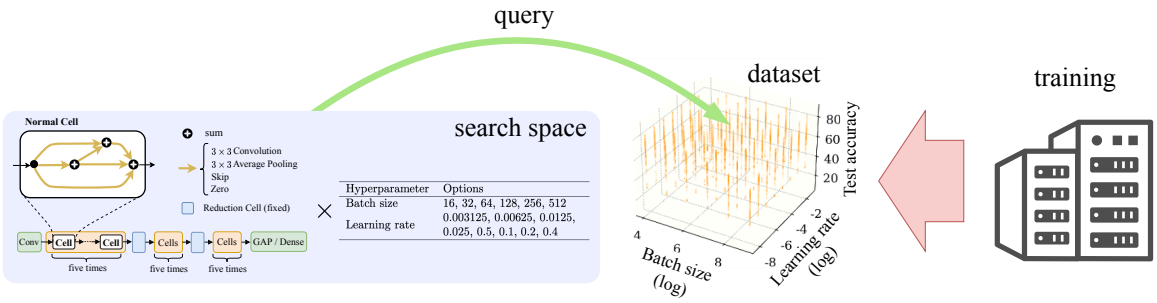


Figure 1: Overview of NAS-HPO-Bench-II.

## 2. Related Work

Many benchmark datasets for NAS have been proposed to realize the fair comparison and the reproducibility of NAS algorithms and have reduced the evaluation cost. NAS-Bench-101 (Ying et al., 2019) is the first NAS benchmark dataset consisting of 510M convolutional neural network (CNN) models, in which 423K computationally unique architectures are included. Each model is represented by stacking cells, and each cell is a computational graph whose nodes represent operations such as a convolution operation. NAS algorithms optimize the architecture of the cell. While NAS-Bench-101 has succeeded in the fair comparison of NAS methods, it cannot be used directly to evaluate one-shot NAS methods (Pham et al., 2018; Liu et al., 2019; Akimoto et al., 2019), which greatly reduce the search cost, because the number of operations in the cell varies across the models. The original search space should be modified to evaluate one-shot NAS methods, as done in NAS-Bench-1Shot1 (Zela et al., 2020).

In contrast, NAS-Bench-201 (Dong and Yang, 2020) and its extended version NATS-Bench (Dong et al., 2021a) innately support one-shot NAS methods by representing operations as edges of computational graphs, which keeps the number of operations in the cell constant. NAS-Bench-201 trained all 15K CNN models on three image datasets, CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and the smaller version of ImageNet (Russakovsky et al., 2015), and shows the consistent ranking of model performances across them. The surrogate dataset NAS-Bench-301 (Siems et al., 2020) provided the model performances in the commonly-used DARTS (Liu et al., 2019) search space by predicting performances of $10^{18}$ models from 60K actually trained performance data. Besides, other NAS benchmark datasets have been proposed in addition to CNN and image classification tasks, such as NAS-Bench-ASR (Mehrotra et al., 2021) for automatic speech recognition, NAS-Bench-NLP (Klyuchnikov et al., 2020) for natural language processing, HW-NAS-Bench (Li et al., 2021) for hardware-aware NAS, and TransNAS-Bench-101 (Duan et al., 2021) for transfer learning. The NAS benchmarks are leveraged for NAS algorithm evaluation (White et al., 2021; Wang et al., 2019).

NAS-HPO-Bench (Klein and Hutter, 2019) is the only benchmark dataset that includes architecture and training hyperparameters to be optimized. This benchmark dataset permits the comparison of joint optimization methods of architecture and training hyperparameters, but the architecture search space is too simple. Namely, the base model is a

Table 1: Comprehensive comparison of NAS benchmark datasets (Size: search space size including equivalent architectures, Arch. type: architecture type to be optimized, Training HPs: presence of training hyperparameters to be optimized, One-shot NAS: support for One-shot NAS, Data: dataset type used for training).

| | Name | Size | Arch. type | Training HPs | One-shot NAS | Data |
|---|---|---|---|---|---|---|
| NAS | NAS-Bench-101 | 510M | Cell | – | – | Image |
| | NAS-Bench-1Shot1 | 363K | Cell | – | ✓ | Image |
| | NAS-Bench-201 | 15K | Cell | – | ✓ | Image |
| | NATS-Bench | 32K | #channels | – | ✓ | Image |
| | NAS-Bench-301[a] | $10^{18}$ | Cell | – | ✓ | Image |
| | NAS-Bench-NLP | 14K | RNN | – | – | Text |
| | NAS-Bench-ASR | 13K | Cell | – | ✓ | Audio |
| | HW-NAS-Bench[b] | 15K / $10^{21}$ | Cell | – | ✓ | Image |
| | TransNAS-Bench-101 | 7K | Cell | – | ✓ | Image |
| NAS + HPO | NAS-HPO-Bench | 62K | MLP | ✓ | – | Tabular |
| | NAS-HPO-Bench-II (Ours)[c] | 192K | Cell | ✓ | ✓ | Image |

*a.* 60K models are trained, and the performance of other models are predicted by a suggorate model.
*b.* Only the data of latency and energy consumption during inference are provided. Two search spaces are adopted and the data of the larger search space is estimated.
*c.* Exact performance data of 192K models for 12 epoch training are available. Estimated performance data for 200 epoch training are provided by using a surrogate model.

multi-layer perceptron (MLP) with two hidden layers, and the architecture hyperparameters are the numbers of units and the types of activations in each layer. Also, the models were trained using the tabular datasets for regression tasks. We note that the architecture search space and target dataset in NAS-HPO-Bench are greatly different from the modern NAS benchmark such as NAS-Bench-101 and 201. To the best of our knowledge, a NAS benchmark including both layer connectivity and training hyperparameters is not established.

This paper develops a novel benchmark dataset for joint optimization of network connections and training hyperparameters that overcomes the drawbacks of NAS-HPO-Bench. We adopt the larger cell-based CNN architecture search space and image classification task, which is based on modern NAS research. Table 1 shows the comparison of existing NAS benchmarks and our proposed benchmark.

## 3. NAS-HPO-Bench-II

This section describes the detail of our benchmark dataset, NAS-HPO-Bench-II, for joint optimization of the cell-based CNN architecture, learning rate, and batch size. We use the CIFAR-10 dataset for image classification to train the models because it is a widely-used dataset for NAS evaluation and is also used in existing NAS benchmark datasets. The architecture search space and training configurations are based on NAS-Bench-201 but with some slight differences. We explain the architecture search space design and training

hyperparameter design in 3.1 and 3.2, respectively. We train all models with each training hyperparameters for 12 epochs. Our dataset provides the train/valid/test accuracies, losses, and wall-clock time of all the configurations for every epoch. We further train the randomly selected architectures for 200 epochs to build the surrogate benchmark dataset for 200 epoch training, as done in NAS-Bench-301. We describe the surrogate model construction in 3.3.

We develop an application programming interface (API) to query performances easily, following existing benchmark datasets. Users can run their NAS methods by querying the training and validation accuracies of 12 epoch for a specific architecture and training hyperparameters and evaluate the discovered architecture and training hyperparameters by querying the surrogate test accuracy of 200 epoch training. Of course, users can use the 12 epoch's test accuracy for NAS method evaluation instead of surrogate data if they do not prefer the surrogate prediction. Using the low-cost API facilitates further developments of joint optimization methods of architecture and training hyperparameters.

### 3.1. Architecture Search Space Design

The overall CNN architecture is constructed by stacking two types of cells: a normal cell and a reduction cell. The former is architecture-specific, whereas the latter is common in all candidate architectures. The normal cell is represented as a directed acyclic graph (DAG) with four nodes: $G = (\{v_0, v_1, v_2, v_3\}, \{e \mid e = (v_i, v_j)\ (0 \leq i < j \leq 3)\})$. Each edge indicates one of the following operations, and each node indicates the sum of incoming edges.

- $3 \times 3$ convolution (ReLU activation, $3 \times 3$ convolution with stride 1, then batch normalization (BN) (Ioffe and Szegedy, 2015)).
- $3 \times 3$ average pooling with stride 1.
- Skip, which outputs the input tensor.
- Zero, which outputs the zero tensor with the same dimension as the input.

The zero operation ensures representing an arbitrary depth and width of architectures. Although NAS-Bench-201 includes a $1 \times 1$ convolution, we remove it to reduce the search space size while keeping various operations. The reduction cell, which halves the image size and doubles the channel size, outputs the sum of two paths. One path consists of an average pooling with stride 2 and a $3 \times 3$ convolution. The other path consists of a $3 \times 3$ convolution with stride 2 and a $3 \times 3$ convolution with stride 1. The model begins with a $3 \times 3$ convolution with 16 output channels, followed by batch normalization. Next, a block of five stacked normal cells is repeated three times connected with the reduction cell. The model ends with the sequence of batch normalization, ReLU activation, global average pooling, and a fully connected layer. This architecture search space contains $4^6 = 4096$ possible cell representations.

We reduce the architectures to be trained from 4K to 1K by identifying the equivalent cell architectures in the search space. First, we convert edge-labeled graphs to node-labeled graphs. Then, we remove the zero operation nodes and their input and output edges and replace the skip operation nodes with edges. We also remove the nodes that are not

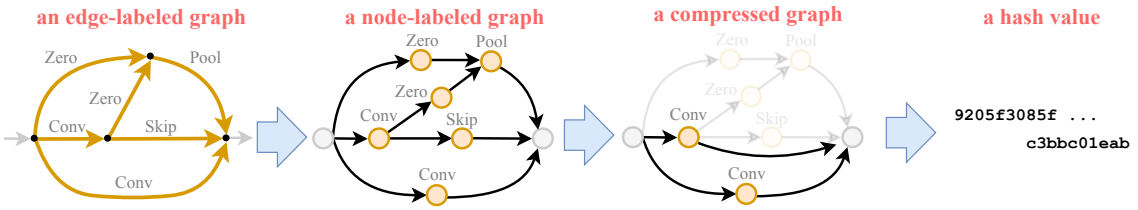


Figure 2: The procedure for getting a graph hash value to identify the equivalent cell architectures.

connected with input or output nodes. Finally, we apply the graph hashing algorithm for node-labeled DAG (Ying, 2019) to the compressed graphs and get the hash values. We identify the equivalent cell architectures by comparing these hash values. Figure 2 illustrates the procedure for getting a graph hash value.

### 3.2. Training Hyperparameter Design

We chose the initial learning rate and batch size as the training hyperparameters to be optimized in our benchmark dataset. All the architectures were trained for the combination of eight initial learning rates and six batch sizes, as shown in Table 2. Several settings have the same ratio of batch size to learning rate, which enables detailed analysis with the scaling rule described in Section 4.2.

Other training settings follow NAS-Bench-201. We use the stochastic gradient descent with Nesterov's momentum of 0.9 for 12 and 200 epochs, the cosine annealing scheduler (Loshchilov and Hutter, 2017), and the cross-entropy loss with the weight decay of $5 \times 10^{-4}$. We use a single NVIDIA V100 GPU to train each model with each initial learning rate and batch size setting. The CIFAR-10 dataset was split into a 25K training dataset, a 25K validation dataset, and a 10K test dataset, a typical setting in NAS. The input pixels are normalized over RGB channels. We apply the random horizontal flip with the probability of 0.5 and the random cropping of a $32 \times 32$ patch with 4 pixels padding on each border as data augmentation.

We train three times with different random seeds in all the configurations. As a result, we get the performance values for 4K architectures and 48 training hyperparameters, a total

Table 2: Training hyperparameter options. The combinations of eight initial learning rates and six batch sizes are used. The learning rate and batch size are set to 0.003125 and 16 as the minimum value, and increase by a factor of 2 to 0.4 and 512, respectively.

| Training Hyperparameter | Options |
|---|---|
| Initial learning rate | 0.003125, 0.00625, 0.0125, 0.025, 0.05, 0.1, 0.2, 0.4 |
| Batch size | 16, 32, 64, 128, 256, 512 |

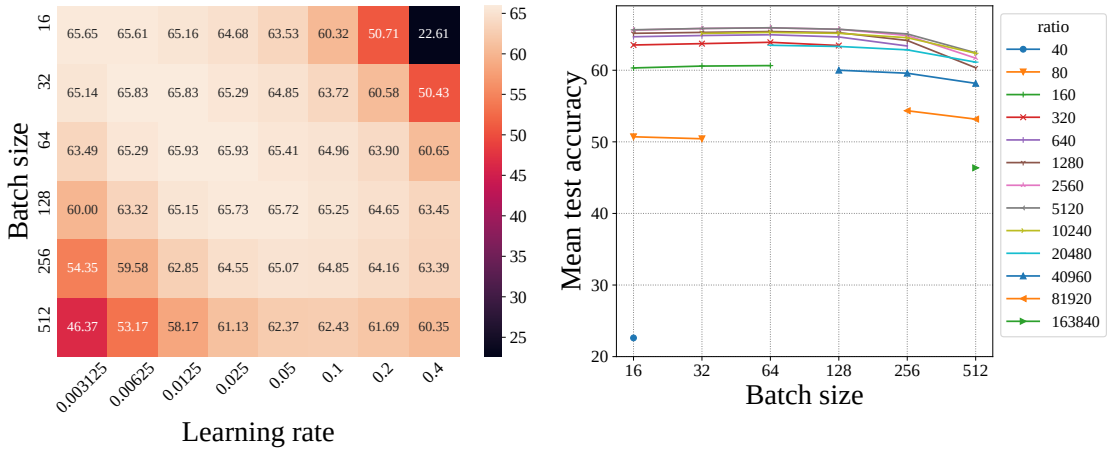


Figure 3: The heatmap on the left shows the mean test accuracy of all the architectures for each training hyperparameter combination. The graph on the right is plotted the performance changes of each ratio's group of the scaling rule, showing that the scaling rule also works in our dataset on average.

of 192K combinations, for 12 epoch training. We note that the actual number of model training is 48K.

### 3.3. Surrogate Model for 200 Epoch Training

We build a surrogate model that predicts the expected test accuracy after 200 epoch training from the architecture and training hyperparameters. To create the dataset for the surrogate model, we train 100 randomly selected architectures three times for 200 epochs using all combinations of initial learning rate and batch size. Therefore, we obtain 4,800 pairs of the hyperparameter and its expected test accuracy as the dataset for the surrogate model.

The surrogate model consists of the bootstrap aggregating (bagging) (Breiman, 1996) of the 10 base models. Each base model transforms the normal cell architecture and the one-hot encoded training hyperparameters into two feature vectors using the graph isomorphism network (GIN) (Xu et al., 2019) and MLP, respectively. Then, the two feature vectors are concatenated and fed to another MLP to predict the expected test accuracy. We perform the architecture-grouped 5-fold cross-validation to evaluate the surrogate model. In each fold, we keep 20% of the training data for early stopping as validation data. Then, we chose the surrogate model at the epoch that recorded the highest $R^2$ score in the validation data and report the test performance. The experimental result shows that our surrogate model achieves a mean $R^2$ score of 0.876 in the test data. Finally, NAS-HPO-Bench-II provides the 200 epoch performance predicted by the surrogate model ensembling five models obtained in the cross-validation process.

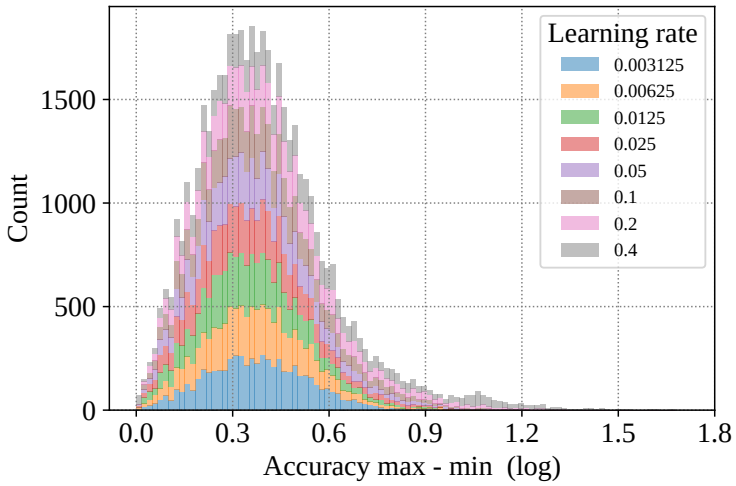


Figure 4: Stacked histogram of the difference between the maximum and minimum test accuracies for 1K unique architectures and 48 training hyperparameters, which is color-coded by learning rates. The horizontal axis shows $\log_{10}(1 + d)$, where $d$ indicates the difference between the maximum and minimum test accuracies of three model training.

## 4. Analysis of NAS-HPO-Bench-II

This section analyzes our benchmark dataset. We inspect the 1K unique architectures identified by graph hash values to avoid duplication and use 12 epoch training results for the analysis.

### 4.1. Data Summary

Figure 3 (left) shows the mean test accuracy of the architectures for each training hyperparameter. We can see that the accuracies vary depending on the combination of hyperparameters. We also observe that the mean test accuracy degrades when using a large batch size and a small learning rate or vice versa. This indicates that a search algorithm should choose better training hyperparameters in our benchmark dataset.

Figure 4 shows the histogram of the difference between the maximum and minimum test accuracies among three model training with different seeds. The most frequent value in the histogram is the 1% accuracy difference, and the values above 1% account for 63.41%. The result indicates the randomness for training cannot be ignored. Therefore, we use the mean accuracy of three model training in the later analysis.

### 4.2. Scaling Rule

We consider the linear scaling rule (Krizhevsky, 2014; Goyal et al., 2017; Smith et al., 2018) to visualize and understand the relationship between test accuracy and training hyperpa-

rameters. The linear scaling rule states that the accuracies become similar when the ratio of the batch size to the learning rate is constant as $|\mathcal{B}|/\eta = r$, where $|\mathcal{B}|$ and $\eta$ denote the mini-batch size and learning rate, respectively, and $r$ is a constant value. We refer to $r$ as the ratio.

We group the settings with the same ratio of $r$, i.e., such settings are the diagonal elements in the heatmap in Figure 3 (left). Figure 3 (right) shows the performance changes of each ratio's group. We observe that the accuracies are stable in the small batch size region, meaning that the scaling rule is also valid in our dataset on average.

### 4.3. Dependency of Architecture and Training Hyperparameters

We investigate the dependency of the architecture and training hyperparameters for test accuracy. Figure 5 (left) depicts the mean test accuracy changes of all 1K unique architectures for the ratio of $r$. We observe that the best-performed ratio differs depending on the architectures, i.e., the best combination of the learning rate and batch size are different. This result supports the necessity of joint optimization of architecture and training hyperparameters to discover the best-performed configuration.

Next, we deeply analyze the relationship between architecture and training hyperparameters. We can heuristically categorize the architectures into three groups that have a similar trend of accuracy change as follows:

**Group 1** Architectures with one direct edge

**Group 2** Architectures not in group 1 and having one or less pooling path and one or more convolutions

**Group 3** Architectures not in groups 1 or 2

Here, a direct edge refers to the edge from the input node to the output node in the compressed graph introduced in Section 3.1, and a pooling path refers to the path from input to output in which all the nodes are average pooling in the compressed graph.

The colors of the lines in Figure 5 (left) indicate the architecture groups. Figure 5 (right) shows how each group's mean test accuracy changes for the ratio of $r$. Also, the accuracy differences of the consecutive ratios are plotted in Figure 5 (right). In group 1, the amounts of change are positive initially and become negative around the ratio value of 320. In group 2, the amounts of change are positive at the beginning and become near zero around 640, then become positive again and turn to negative around 10240. In group 3, the amounts of change are positive at first and turn to negative around 1280. We observe that the training hyperparameters with the ratio around 320 are a good choice for the architectures in group 1, while the ratios around 10240 and 1280 are appropriate for groups 2 and 3, respectively. This result implies that the architecture topology information may be useful to determine the best training hyperparameters.

## 5. Benchmarking of Joint Optimization Algorithms

We evaluate several search algorithms using NAS-HPO-Bench-II to demonstrate the use case of our benchmark dataset. We run well-known NAS and hyperparameter optimization

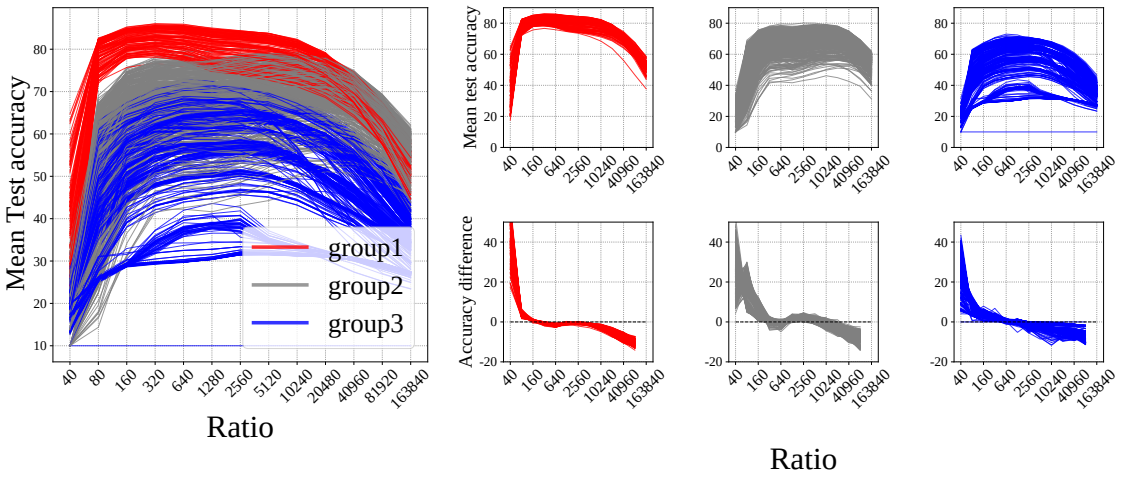


Figure 5: The left side graph shows the relationship between the ratio of the scaling rule and test accuracy. Each line indicates the transition of a single architecture. The test accuracies are averaged over the same ratio value of $r$ as $\bar{a}_{(r,m)} = \sum_{h \in H_r} f(m,h)/|H_r|$, where $m$ represents some architecture, $H_r$ is a set of training hyperparameters with the same ratio $r$, and $f$ is a function that outputs test accuracy. The upper-right three graphs show the relationship between the ratio and test accuracy of each architecture group. The lower-right three graphs show the transition of accuracy differences of the consecutive ratios, where the vertical axis denotes $\bar{a}_{(2r,m)} - \bar{a}_{(r,i)}$.

algorithms for 500 trials with a time budget of 20,000 seconds $\approx$ 6 hours: random search (RS) (Bergstra and Bengio, 2012), regularized evolution (RE) (Real et al., 2019), REINFORCE (Williams, 1992), and BOHB (Falkner et al., 2018) as joint optimization. Also, we run a combination of BOHB and RE and a combination of RS and RE as sequential optimization (running BOHB/RS first as HPO, then running RE in the remaining time as NAS). Specifically, we initially run HPO for a time budget of 5,000 seconds with a randomly selected architecture and then run the NAS algorithm for the remaining time budget. The implementation of compared algorithms is based on the publicly available code.[2] We treat the architecture and training hyperparameters as categorical variables. In the optimization phase, the validation accuracy of 12 epoch training with a specific random seed is used as the performance measure. The selected architecture and training hyperparameters by the optimization algorithm are evaluated using the test accuracy of 12 epoch training and the predicted mean test accuracy of 200 epoch training provided by the surrogate model. NAS-HPO-Bench-II allows us to reduce the benchmarking cost of 750 GPU days to less than one CPU hour.

2. https://github.com/D-X-Y/AutoDL-Projects/tree/97717d826e1ef9c8ccf4fa29b454946772b2137e/exps/NATS-algos

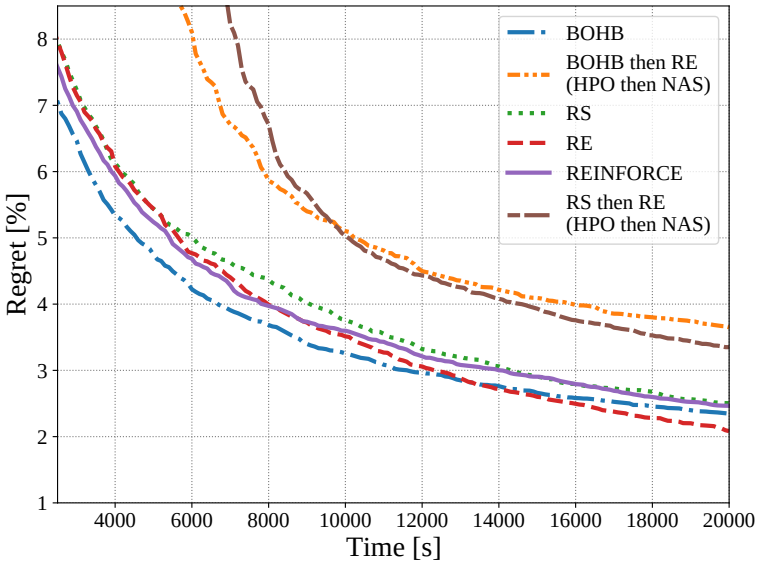


Figure 6: The transition of the mean regret of six algorithms.

Table 3: The test accuracies obtained by six algorithms. The test accuracies of exact 12 epoch training and surrogate-predicted 200 epoch training are reported. The mean and standard deviation of test accuracies over 500 runs are displayed.

| Algorithm | RS | RE | REINFORCE | BOHB | BOHB then RE | RS then RE |
|---|---|---|---|---|---|---|
| Result (200 epoch) | $87.41 \pm 2.14$ | $87.85 \pm 1.45$ | $87.65 \pm 1.32$ | $87.86 \pm 0.89$ | $87.39 \pm 6.54$ | $87.82 \pm 1.30$ |
| Result (12 epoch) | $83.90 \pm 1.34$ | $84.32 \pm 1.71$ | $83.95 \pm 1.13$ | $84.47 \pm 0.92$ | $82.63 \pm 3.84$ | $83.16 \pm 2.26$ |

Figure 6 shows the transition of the mean regret of the validation accuracy. The regret is defined by $R_T = f(s^*) - \max_{t \leq T} f(s_t)$, where $s_t$ is the model setting of the $t$-th iteration in a single run, $s^*$ is the best setting in the search space, and $f$ outputs the validation accuracy of a given model setting. The result shows joint optimization methods greatly outperform sequential optimization methods in our settings, which suggests the necessity of joint optimization. Among the joint optimization methods, regularized evolution reaches the best regret at the end of the optimization, while BOHB shows an early convergence compared to the other algorithms.

Table 3 shows the test accuracies of selected model settings by six algorithms. We report both test accuracies of 12 and 200 epoch training. We have compared the simple baseline algorithms using our dataset. Our dataset may be useful to develop a more sophisticated optimization algorithm for joint optimization of architecture and training hyperparameters.

## 6. Conclusion and Future Work

We have created NAS-HPO-Bench-II, the first benchmark dataset for joint optimization of network connections and hyperparameter optimization in the cell-based CNN. The dataset analysis supports the necessity of joint optimization due to the dependency of architecture and training hyperparameters for test accuracy. We have also demonstrated the comparison of the baseline optimization algorithms using our proposed dataset. We believe that NAS-HPO-Bench-II contributes to the further development of NAS and hyperparameter optimization algorithms.

We limited the candidate architecture and training hyperparameters in the search space due to the cost of dataset construction. For example, although the total search space size of our dataset is greater than many existing benchmarks, the architecture search space is smaller than that of NAS-Bench-201, and we focus on two types of training hyperparameters. Expanding the search space size is a desired research direction to simulate a realistic scenario, e.g., adding the weight decay coefficient and dropout ratio as training hyperparameters to be optimized. While this paper focuses on the architecture and training hyperparameters in CNN, it would be worthwhile to create complementary benchmark datasets, such as those treating the data augmentation setting and other deep neural network models.

## Acknowledgments

We are grateful to Dr. Aaron Klein and Prof. Frank Hutter as the authors of NAS-HPO-Bench for allowing us to call our paper NAS-HPO-Bench-II. This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We used the computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) for our experiments.

## References

Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *International Conference on Machine Learning (ICML)*, 2019.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)*, 13(10):281–305, 2012.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. NATS-Bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021a.

Xuanyi Dong, Mingxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc V. Le. AutoHAS: Efficient hyperparameter and architecture search. In *International Conference on Learning Representations (ICLR) Workshop*, 2021b.

Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research (JMLR)*, 20(55):1–21, 2019.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning (ICML)*, 2018.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*, 2019.

Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. NAS-Bench-NLP: Neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*, 2012.

Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Lin. HW-NAS-Bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations (ICLR)*, 2021.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Abhinav Mehrotra, Alberto Gil C. P. Ramos, Sourav Bhattacharya, Łukasz Dudziak, Ravichander Vipperla, Thomas Chau, Mohamed S. Abdelfattah, Samin Ishtiaq, and Nicholas Donald Lane. NAS-Bench-ASR: Reproducible neural architecture search for speech recognition. In *International Conference on Learning Representations (ICLR)*, 2021.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*, 2018.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning (ICML)*, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. NAS-Bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations (ICLR)*, 2018.

David So, Quoc V. Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning (ICML)*, 2019.

Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2017.

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. AlphaX: eXploring neural architectures with deep neural networks and Monte Carlo tree search. *arXiv preprint arXiv:1903.11059*, 2019.

Colin White, Willie Neiswanger, and Yash Savani. BANANAS: Bayesian optimization with neural architectures for neural architecture search. In *AAAI Conference on Artificial Intelligence*, 2021.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Antoine Yang, Pedro M. Esperança, and Fabio Maria Carlucci. NAS evaluation is frustratingly hard. In *International Conference on Learning Representations (ICLR)*, 2020.

Chris Ying. Enumerating unique computational graphs via an iterative graph invariant. *arXiv preprint arXiv:1902.06192*, 2019.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning (ICML)*, 2019.

Arber Zela, Julien Siems, and Frank Hutter. NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.