Arch-Graph: Acyclic Architecture Relation Predictor for Task-Transferable Neural Architecture Search

Minbin Huang[1] Zhijian Huang[1] Changlin Li[3] Xin Chen[4] Hang Xu[2]
Zhenguo Li[2] Xiaodan Liang[1*]
[1]Shenzhen Campus of Sun Yat-sen University [2]Huawei Noah's Ark Lab
[3]ReLER, AAII, UTS [4]The University of Hong Kong
{huangmb5,huangzhj56}@mail2.sysu.edu.cn, changlinli.ai@gmail.com, cyn0531@connect.hku.hk,
chromexbjxh@gmail.com, li.zhenguo@huawei.com, xdliang328@gmail.com

Abstract

*Neural Architecture Search (NAS) aims to find efficient models for multiple tasks. Beyond seeking solutions for a single task, there are surging interests in transferring network design knowledge across multiple tasks. In this line of research, effectively modeling task correlations is vital yet highly neglected. Therefore, we propose **Arch-Graph**, a transferable NAS method that predicts task-specific optimal architectures with respect to given task embeddings. It leverages correlations across multiple tasks by using their embeddings as a part of the predictor's input for fast adaptation. We also formulate NAS as an architecture relation graph prediction problem, with the relational graph constructed by treating candidate architectures as nodes and their pairwise relations as edges. To enforce some basic properties such as acyclicity in the relational graph, we add additional constraints to the optimization process, converting NAS into the problem of finding a Maximal Weighted Acyclic Subgraph (MWAS). Our algorithm then strives to eliminate cycles and only establish edges in the graph if the rank results can be trusted. Through MWAS, Arch-Graph can effectively rank candidate models for each task with only a small budget to finetune the predictor. With extensive experiments on TransNAS-Bench-101, we show Arch-Graph's transferability and high sample efficiency across numerous tasks, beating many NAS methods designed for both single-task and multi-task search. It is able to find top 0.16% and 0.29% architectures on average on two search spaces under the budget of only 50 models.*[1]

1. Introduction

Neural Architecture Search (NAS) methods [2, 40] have the potential to democratize deep learning and reduce costly human labor in designing neural networks. By automatically searching for optimal architectures, many NAS methods have discovered models exceeding human-designed ones on various tasks. However, many NAS solutions are computationally expensive as they require training over numerous candidate architectures. Under cases where networks for multiple tasks are needed, searching for an architecture for each task requires repeatedly running NAS methods from scratch to find the top performing network, throwing away potentially valuable knowledge accumulated over the course of searching. There are many recent attempts [15, 34] investigating transferable NAS problems over different tasks by mining task correlations. For instance, [15] proposes to use meta-learning to generate architectures for a given new task. However, it makes a strong assumption that information on top-performing architectures for each pretrain task is always available, which can limit its use case. [34] proposed to use task embeddings to inform an RNN controller of the task information and framed NAS as a reinforcement learning (RL) problem, which inherits the sample inefficiency problem from RL. Weight-sharing

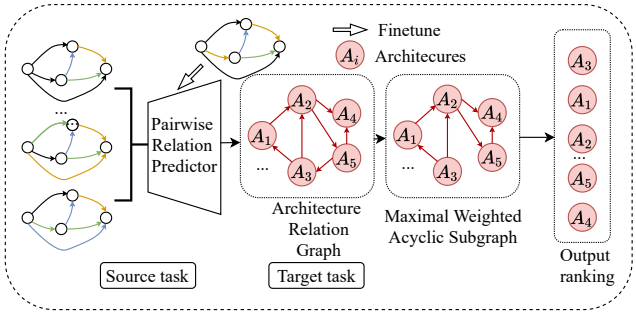

Figure 1. Overview of our Arch-Graph that trains a pairwise relation predictor on a source task and transfers to target task by finetuning. It constructs an architecture relation graph based on the pairwise relation predictor. After selecting the MWAS of the architecture relation graph, Arch-Graph can give a proper ranking of different candidate architectures.

*Corresponding author.
[1]Code: https://github.com/Centaurus982034/Arch-Graph

techniques [16–18, 25] are recently more popular among researchers due to their efficiency in cost reduction, typically by training a supernet and then inheriting weights from it. However, due to their restrictions in supernet design, weight-sharing methods are usually constrained in the choice of network search space.

Predictor-based NAS methods [23, 28, 33, 35, 37] alleviate these concerns by sampling architecture-performance pairs and fitting a proxy accuracy predictor to reduce computation costs. However, training a large number of architectures for fitting a good predictor can also be computationally challenging. Besides, this approach is ultimately converting NAS into a regression problem, which can be hard to solve since the model space is usually highly non-convex, making accurately identifying top performers extremely difficult. In this paper, we instead argue that approaching NAS as a *ranking* problem can bring along many extra benefits compared to other methods, largely due to its added constraints that provide extra learning signals.

This key observation motivated us to develop a predictor that captures pairwise relations among architectures and formulate NAS as a graph ordering problem. Our method, Arch-Graph, treats architectures as nodes and order information as directed edges, such that an edge pointing from $arch_a$ to $arch_b$ represents the superiority of $arch_a$ in its performance when compared to $arch_b$. We propose to use a pairwise relation predictor to construct this graph. This predictor is optimized with objective of finding the correct *pairwise* order of nodes in the graph, which greatly improves data efficiency and prediction accuracy comparing to previous pointwise predictor that directly predict architecture performance.

To allow transfering among different tasks without retraining the predictor, another key ingredient *task embedding* that represents a task during the predictor training process stablizes the knowledge transfer between tasks. Previous works on task embedding mostly focus on classification tasks [1], whereas our proposed task embedding method is more general and can be applied to many other vision domains such as autoencoding and semantic segmentation.

After constructing the relation graph through the pairwise predictor, the architecture selection can then be formulated as a topological ordering problem on this graph. Under this setting, it is vital to enforce that the graph follows basic properties of a partial order, such as acyclicity, which prohibits circular ordering (A $>$ B $>$ C while C $>$ A). Therefore, a central component of our work is the definition of a Maximal Weighted Acyclic Subgraph (MWAS) problem with Trust Score to make sure the constructed graph follows the irreflexive, transitive, and anti-symmetric properties of a partial order. We propose an approximation solution to it by iteratively applying the *max-MAS* algorithm.

Our experiments on TransNAS-Bench-101 proves the effectiveness of Arch-Graph, identifying architectures with average rank 5.24 (top 0.16%) and 12.2 (top 0.29%) on macro and micro search space respectively, with only randomly sampling 50 architectures, saving at least 37.5% of samples in other methods to achieve comparable results.

To conclude, the contributions of our work can be summarized as follows:

- We propose Arch-Graph, a task transferable NAS method by formulating NAS from a novel perspective: A graph ordering problem, and solve this problem by training a *pairwise* relation predictor, which is more data efficient, saving at least 37.5% training samples.
- We generalize *task embeddings* to any kind of tasks, and further enables task-transferable NAS by predicting architecture relation on any given task embeddings.
- To remove incorrect edges in the relation graph constructed by the predictor, we define the Maximal Weighted Acyclic Subgraph problem and propose an approximation algorithm to solve it.
- Extensive experiments demonstrate that Arch-Graph can beat many existing transferable NAS methods by a large margin, finding top 0.16% and 0.29% architectures on two search spaces.

2. Related Work

Predictor-based NAS. NAS has achieved many breakthroughs in the past few years. Its early works utilized reinforcement learning [30, 38, 39, 41] and evolutionary algorithms [20, 26, 27, 29, 36] and found many top-performing architectures at a high computational cost. Later works then strive to reduce the search cost while improving performance. Among numerous directions, predictor-based NAS methods are most relevant to our work. They try to predict the performance of a given neural architecture both accurately and efficiently. These methods usually involve two steps: 1) Sampling pairs of architectures and their accuracies, and 2) learning the accuracy predictor. The objective of fitting the predictor can be regarded as a regression [33] or ranking [23, 37] problem, and there is a wide range of choices for predictors [9, 21, 22, 32]. Shi *et al*. [28] adopted a bayesian sigmoid regression as the surrogate model for Bayesian Optimization (BO) to select candidates. As applying BO on the whole search space is difficult, weakNAS [35] replaced one strong predictor with a set of weaker predictors to get oversimplified BO. Different from these previous works, we propose pairwise relation predictor and formulate NAS problems as a graph ordering problem where the graph is given by the predictor.

Transferable NAS. Transfer learning for NAS mainly focuses on transferring between tasks using the same search space and between search space on a specified task. There are some recently proposed cross-task NAS benchmarks

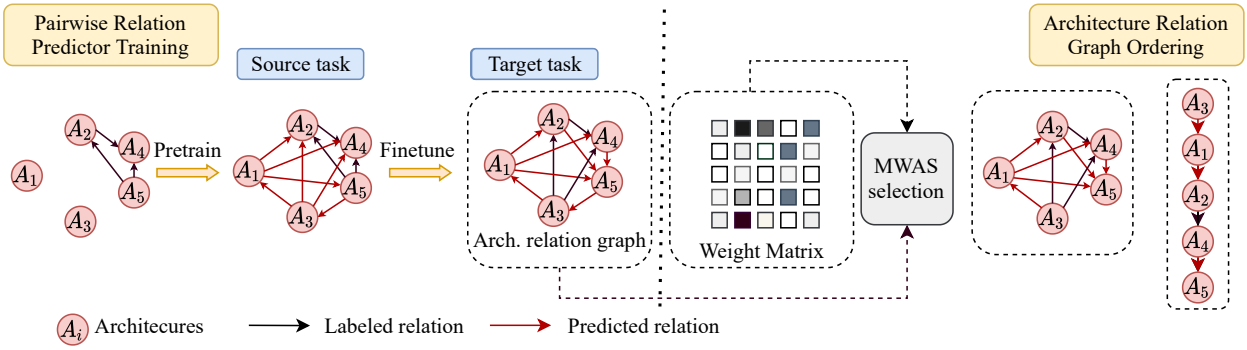

Figure 2. Framework of our proposed Arch-Graph. In the Pairwise Relation Predictor Training stage, given a source task and architectures of interest, we sample a small budget of architectures to fit the predictor then finetune it on a target task. Next, in the Architecture Relation Graph Ordering stage, we construct a relation graph according to the prediction, treating each architecture as a node and directed edge as ranking information. To get a proper ordering from the relation graph, we assign weights related to the confidence to the edges and select the Maximum Weighted Acyclic Subgraph and get a Directed Acyclic Graph (DAG) from the relation graph. Finally, we evaluate the top architectures given by the topological sorting of this DAG.

[7, 31] for improving the transferability and generalizability of NAS algorithms. Though relatively neglected when compared to single-task NAS, there are still some outstanding algorithms. CAS [24] applies continuous learning on multi-task architecture search based on a weight sharing strategy, trying to find a single cell structure that can generalize well to unseen tasks. Catch [4] combined meta-learning with RL to swiftly adapt to new tasks. Different from [22] for single task, Lee *et al*. [15] proposed to generate graphs from datasets in a meta-learning style to make the methods generalize well across multiple datasets. However, it requires top-performing architectures during training to learn characteristics of good models, which can incur high computational costs. Contrary to this, our method achieves remarkable results only by random sampling.

3. Arch-Graph

Transferable NAS methods aim to reuse the architecture selection knowledge from source tasks and find top-performing architectures on a target task. Consistent with this setting, the Arch-Graph algorithm consists of two parts: pairwise relation predictor training and architecture relation graph ordering, as illustrated in Fig. 2. We train the pairwise relation predictor (Sec. 3.1) on the source task using sampled architecture pairs and task embeddings (Sec. 3.2), then finetune it on the target task. After constructing architecture relation graphs using the finetuned predictor, we rank the architectures by finding a Maximum Weighted Acyclic Subgraph (MWAS) (Sec. 3.3).

3.1. Pairwise Relation Predictor

For a predictor-based NAS algorithm, the predicted ranking of models might matter more than absolute numbers of model performance prediction, since we only care about the top-ranking ones. Many predictor-based NAS methods focus on directly predicting the accuracy of models [33] or the ranking of all models of interest through a ranking loss [23, 37]. However, since model spaces are usually highly non-linear, these predictors typically cannot be trained to have high accuracy. Moreover, these methods are not data-efficient since they need lots of samples to fit the predictor on a complicated model space.

We propose to study NAS from a new perspective, which is to formulate it as an architecture relation graph ordering problem. Our key observation is that while ranking all models can be problematic, it is much easier to make comparisons just between two models. Besides, as previous works [8, 10] illustrated, when challenged with limited available data, learning pairwise relations can yield a higher classifier performance than many common regression methods. This is because we can "augment" the data by constructing $n^2 - n$ pairs of relations when we only have n labeled samples. This is extremely helpful in settings where obtaining labels is computationally expensive, such as NAS. This inspired us to use a well-trained pairwise relation predictor to get a ranking of models in a search space. It is thus crucial to properly define *relation* in our settings, where the most relevant concept is *partial order*.

Definition 1 *(partial order) A (strong) partial order on a set P is a relation $\prec$ that is both **irreflexive**, **transitive** and **anti-symmetric**, that is, for $\forall a, b, c \in P$:*
1. irreflexive: not $a \prec a$.
2. transitive: if $a \prec b$ and $b \prec c$ then $a \prec c$.
3. anti-symmetric: if $a \prec b$ then not $b \prec a$.

Definition 2 *(total order) a total order is a partial order on a set P so that for $\forall a, b \in P$, either $a \prec b$ or $b \prec a$.*

If a well-trained predictor defines a partial order, the problem of ranking models is then reduced to extending a partial order (Definition 1) to a total order (Definition 2), which has been extensively studied in the existing literature. There-

fore, our predictors are trained to define a partial order on the model space.

Given a source task τ_s, we first randomly pick m models from τ_s and fully evaluate them to get their performance on the test dataset. In this way, we obtain $m^2 - m$ samples by forming pairwise relations. Details of the pairwise relation predictor is illustrated in Fig. 3. The $(arch_a, arch_b)$ are randomly sampled architectures that are first concatenated as the input of a Graph Convolutional Network (GCN) [14]predictor. The GCN predictor then generates two embeddings to represent these two architectures. Next, these embeddings are concatenated with a task embedding, which is generated by applying a fully connected layer to the feature extractor described in Sec. 3.2. Together, they are fed into a softmax function to construct a simple probability distribution $p = (p_a, p_b) \in \mathbb{R}^2$ with $p_a > p_b$ indicating $arch_a$ is better than $arch_b$. The produced probability distribution is then compared with the ground truth label$\{[0,1]^T, [1,0]^T\}$. The objective is to minimize the Binary Cross Entropy (BCE) Loss. Specifically, we include both $(arch_a, arch_b)$ pairs and $(arch_b, arch_a)$ pairs to encourage anti-symmetry. If neither $a \to b$ nor $b \to a$ exists, we simply mark them as incomparable, which is allowed in a partial order.

After training the pairwise relation predictor on a source task, we conduct transfer learning by finetuning the predictor on a set of t target tasks $\{\tau_1, \tau_2, ..., \tau_t\}$ with a small budget of b architectures chosen from each target task. More specifically, b_f architectures for finetuning the predictor and b_v architectures for pairwise relation validation. We pick the predictor with the highest validation accuracy as the final result. Then, the architecture relation graph ordering is performed on τ_i on top of the finetuned predictor.

3.2. Task Embedding

When transferring architecture knowledge across tasks, it is important to inform NAS methods of the target task's intrinsic characteristics and adjust the architecture selection strategy accordingly. We therefore follow [1], which only generates task embeddings for classification tasks, extend it to generate embeddings for other tasks.

A task's nature can be quantified by the neural network's weights when trained on this task. When a pre-trained model is finetuned on a task τ_i, it is actually adding some perturbation $w' = w + \delta w$ to a network's weights and we can measure the average KL divergence between the original output distribution $p_w(y|x)$ and the perturbed one $p_{w'}(y|x)$. It can be measured by

$$\mathbb{E}_{x\sim\hat{p}} KL(p_{w'}(y|x)||p_w(y|x)) = \delta w F \delta w \tag{1}$$

where F is the Fisher Information Matrix (FIM):

$$F = \mathbb{E}_{x,y\sim\hat{p}(x)p_w(y|x)} \left[\nabla_w \log p_w(y|x) \nabla_w \log p_w(y|x)^T\right] \tag{2}$$

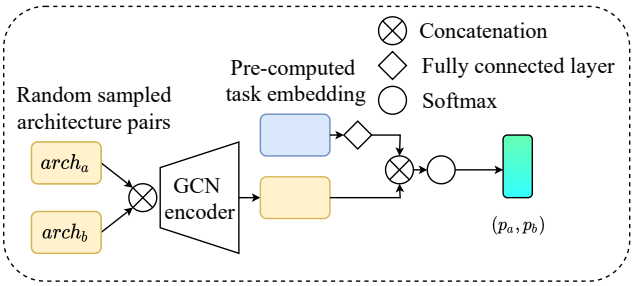

Figure 3. Detailed structure of our proposed pairwise relation predictor. The predictor takes an architecture pair $(arch_a, arch_b)$ and a task embedding as input and produce a probability vector p_a, p_b, where$p_a > p_b$ indicates that $arch_a$ is better than $arch_b$.

Algorithm 1: Calculate the approximation of MWAS

Input:
A: the adjacency matrix of a (cyclic) graph $\mathcal{G}$;
S: the edge weight matrix;
ϵ: threshold, calculated by $\epsilon = 1 - Acc(\tau_t)$;
Set $s_0 = a = 0, r = b = ||A||_1, seg = b - a$;
while $seg > 1$ **do**
  $A_T \leftarrow$ *max-MAS*(A, r);
  if A_T *doesn't exist* **then**
    Find larger r: $r \leftarrow r + 1$;
    Move the left endpoint of the interval to r: $a \leftarrow r$;
  else
    Calculate score: $s \leftarrow \sum_{i,j}(A_T \odot S)_{ij}$;
    if $R(A_T) < \epsilon$ *and* $s > s_0$ **then**
      Record maximal score: $s_0 \leftarrow s$;
      Maintain a subgraph with maximal score: $A_T^{(best)} \leftarrow A_T$;
    end
    Halve the length of the interval: $seg \leftarrow \lfloor \frac{seg}{2} \rfloor$;
    $r \leftarrow r - seg$;
    Move the right endpoint of the interval to r: $b \leftarrow r$;
  end
end
Output: $A_T^{(best)}$ as the approximation of the MWAS

The FIM then indicates the set of feature maps which are more informative for solving the current task. We use an ImageNet pre-trained ResNet-50 as the encoder, then train an encoder-decoder network for each task with a randomly initialized decoder. In this way, parameters of the ResNet-50 encoder are adjusted according to each task's characteristics. The encoder is essentially a task feature extractor, and we simply compute an FIM for this feature extractor. The FIM is then used as the task embedding for each task, which is a fixed dimensional vector.

3.3. Architecture relation graph ordering

Relation Graph Construction After obtaining the finetuned pairwise relation predictor on the target task τ_k, we can construct a directed graph $\mathcal{G}^{\tau_k}$ with an adjacency matrix A^{τ_k}. The presence of a directed edge from node a to node b in $\mathcal{G}^{\tau_k}$ represents the prediction that architecture $arch_a$ is

better than architecture $arch_b$. That is, $A_{ij}^{\tau_k} = 1$ indicates that $arch_a$ has higher performance than $arch_b$ in task τ_k. Since the predictor can be error-prone on the pairwise relation, there can be lots of noisy edges in the graph. This can result in cycles that violate the transitivity of a partial order, which can affect the ranking of the models. (as in Fig. 2, $A_3 \to A_2 \to A_5 \to A_3$ forms a cycle). Ideally, we want to obtain a Directed Acyclic Graph (DAG), where its edges collectively define a partial order and its topological sort defines a ranking of nodes.

Maximal Weighted Acyclic Subgraph (MWAS). Based on our observations above, we aim to find a subgraph that satisfies the following properties: a) The edges exist with high confidence; b) There are no cycles in the subgraph; c) The subgraph is as close to the original graph as possible. This leads to our definition of Maximal Weighted Acyclic Subgraph (MWAS):

Definition 3 *(Maximal Weighted Acyclic Subgraph, MWAS) Given a directed (cyclic) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $|\mathcal{V}| = n$, with adjacency matrix A and non-negative edge weight matrix S, the MWAS is the problem of finding an acyclic subgraph $T = (\mathcal{V}, \mathcal{E}_T)$ of $\mathcal{G}$ with adjacency matrix A_T, that maximizes the score $p(T) = \sum_{e \in \mathcal{E}_T} S_e = \sum_{i,j}^{n} A_T \odot S$ and minimizes $||A_T - A||$.*

Intuitively, maximizing $p(T)$ enforces that if an edge exists, the network has high confidence in its correctness. Minimizing $||A_T - A||$ pushes the subgraph to keep as much edges from $\mathcal{G}$ as possible. As Guo *et al*. mentioned [11], the raw confidence values from a classifier can be poorly calibrated and are not reliable to determine the confidence of the classifier itself. Therefore, we adopt trust score defined in [12] to assign weights to the edges, determining to what extend we should trust an edge in $\mathcal{G}^{\tau_k}$.

Definition 4 *(Trust Score) Given a testing sample $x \in \mathcal{X}$ and a trained classifier $h : \mathcal{X} \to \mathcal{Y}$, let y_{pred} be the predicted class of x and y_n be the nearest class different from y_{pred}, then the trust score is defined by $\frac{d(x-y_n)}{d(x-y_{pred})}$.*[3]

With this definition, we calculate the trust score for each edge in Graph $\mathcal{G}$ and get the edge weight matrix S.

The Maximal Acyclic Subgraph (MAS) problem was included by R.Karp in his list of 21 NP-complete problems. [13]. Cvetkovic *et al*. proposed an algorithmic solution [5] to the max-MAS problem (Definition 5) closely related to the MAS problem.

Definition 5 *(The max-MAS problem) Finding the minimal integer r such that a given graph $\mathcal{G}$ with adjacency matrix A can be made acyclic by cutting at most r incoming edges from each vertex.*

Cvetkovic *et al*. formulated it as the following optimization problem (see S.M.[2]):

$$\begin{aligned} &\min \ \rho(X) \\ &s.t. \quad X \in \mathcal{B}(A, r) \end{aligned} \tag{3}$$

where ρ is the spectral radius and $\mathcal{B}(A, r)$ is the L_1 ball centered at A. Consequently, they find by integer bisection the smallest r such that the objective function value equals to zero (a directed graph is acyclic if and only if its spectral radius is zero, see S.M.[2]). We denote the solution with respect to r as *max-MAS*(A, r). We notice that in our relation graph where a large number of edges are clean, *max-MAS*(A, r) can return reasonable solution A_T in the measure of $||A_T - A||$ even when r is not small enough. Suppose the accuracy on the validation set of target task τ_t is $Acc(\tau_t)$, we empirically save the *max-MAS*(A, r) with edge dropping ratio $R(A_T)$ smaller than $1 - Acc(\tau_t)$ during integer bisection of r from $||A||_1$ to 0. With different approximations available, we calculate the trust score of these approximations and pick the one with the highest trust score as our approximation of MWAS. Details of the selection process are described in Algorithm 1.

After obtaining a MWAS, we can apply transitive reduction to it and get a Hasse Digram. A topological sorting is easy to find out on a Hasse Digram and hence the predicted ranking of the models for this task is determined by the topological order.

3.4. Training and Inference

In a transfer learning setting, we first randomly sample a small budget of m models from the source task and fully evaluate them. After the pairwise relation predictor is trained on the source task, we finetune it on each target task for another small budget of b models. After finding the MWAS for each task's Arch-Graph, we evaluate the top p models given by MWAS for each task and pick the best model as the final result.

4. Experiments

4.1. Datasets and Implementation Details

TransNAS-Bench-101. TransNAS-Bench-101 (TB101) [7] is a benchmark dataset providing architecture performance across seven vision domains including classification, regression, pixel-level prediction and self-supervised tasks. It provides opportunities to evaluate transferable NAS methods among different tasks.[3] There are two types of search space in this benchmark, i.e., the widely-studied cell-based search space containing 4096 architectures and macro skeleton search space based on residual blocks containing 3256 architectures.[2] Following Lucaz *et al*. [8], we

[2]S.M. for Supplementary Materials.

[3]More details can be found in Supplementary Materials

	Tasks	Cls.O.	Cls.S.	Auto.	Normal	Sem. Seg.	Room.	Jigsaw	Avg. Rank
	Metric	Acc.$^{\uparrow}$	Acc.$^{\uparrow}$	SSIM$^{\uparrow}$	SSIM$^{\uparrow}$	mIoU$^{\uparrow}$	L2 loss$^{\downarrow}$	Acc.$^{\uparrow}$	
Single NAS	RS [3]	46.85	56.50	70.06	60.70	28.37	59.35	96.78	59.26
	REA [26]	47.09	56.57	69.98	60.88	28.87	58.73	96.88	41.03
	BONAS [28]	46.85	56.47	74.45	61.62	28.82	59.39	96.76	33.37
	weakNAS [35]	47.40	56.88	72.54	62.37	29.18	57.86	96.86	10.49
	Arch-Graph-single	47.35	56.77	71.32	62.78	29.09	58.05	96.70	12.68
Transfer NAS	DT	45.48	54.96	59.35	58.60	26.21	62.07	95.37	534.31
	CATCH [4]	47.29	56.49	70.36	60.85	28.71	59.37	-	37.72
	REA-t [26]	46.98	56.60	73.41	61.02	28.90	58.18	-	28.98
	BONAS-t [28]	47.06	56.86	71.41	61.44	28.76	58.35	-	27.87
	nsganetv2 [19]	46.86	56.29	73.77	61.41	28.73	59.07	-	34.39
	weakNAS-t [35]	47.13	56.83	73.59	61.86	29.07	58.55	-	15.43
	Arch-Graph-zero	47.42	56.78	75.51	63.39	29.17	58.15	-	7.83
	Arch-Graph	**47.44**	**56.98**	**75.90**	**64.35**	**29.19**	**57.75**	-	**5.24**
	Global Best	47.96	57.48	76.88	64.35	29.66	56.28	97.02	1

$^{\uparrow}$ indicates higher is better, $^{\downarrow}$ indicates lower is better, **bold** indicates the best result.

Table 1. Performance comparisons between different NAS methods on our Arch-Graph on Macro level search space. Jigsaw results are omitted for TransferNAS methods because it is used as the pretrain task.

change the operation-on-edge setting in TransNAS-Bench-101 to an operation-on-node setting and encode each architecture as a graph with a fixed adjacency matrix and node feature matrix representing different operations.

NAS-Bench-201. NAS-Bench-201 (NB201) [6] is a benchmark containing 15,625 architectures. It provides full information of these architectures on three classification tasks including CIFAR-10, CIFAR-100 and ImageNet-16-120. Note that our Arch-Graph can also be applied to single-task setting. To further verify the effectiveness of our Arch-Graph, we conduct experiments of a single-task variant named *Arch-Graph-single* by simply pretraining and finetuning the predictor on the same task.

Pairwise Relation Predictor. To match the experiment in [7], we pretrain the pairwise relation predictor on the least time-consuming task, jigsaw (details of pretraining on other tasks can be found in S.M.[2]), restricting to a fixed budget of $m = 50$ models. Then we finetune on each remaining task for another $b = 30$ models using $b_f = 20$ for training and $b_v = 10$ for validation. Consequently, we construct the Arch-Graph using the predicted directed edges for each task and use them to get an ordering of architectures.

Architecture Relation Graph Ordering. After we obtain architecture relation graph on the target tasks, we first use a naive method to order the architectures on the relation graph, named **Arch-Graph-zero**[4]. We implement the insertion sort algorithm to the model space by using the finetuned pairwise relation predictor as the comparison operator. Since there are incomparable elements and noisy edges (cycles) confusing the comparison operator, we simply skip the comparison until we can find a place to insert the not-yet-sorted architecture. This gives us a coarse ranking of the model space.

Because of high complexity of obtaining MWAS, we do not compute MWAS for the whole Arch-Graph. Instead, we pick top 500 models given by the coarse prediction of Arch-Graph-zero and construct the relation graph of 500 nodes using their predicted edges. Later ordering is conducted on this graph. After finding the MWAS (Algorithm 1), we evaluate the top $p = 20$ models given by the topological sort of these nodes. If any model selected for the final evaluation is already sampled, we simply skip it and evaluate the next model until we have evaluated p models. Results on macro level search space and micro level search space can be found in Tab. 1 and Tab. 2.

4.2. Comparison with state-of-the-art NAS

Single-task NAS. On TB101, we use Random Search (RS) [3] and Regularized Evolutionary Algorithm (REA) [26] for 50 epochs as baselines. We then conduct experiments using two state-of-the-art predictor-based NAS methods, BONAS [28] and weakNAS [35] on each task. The total budget for each method is set to 50 randomly selected models. The average model rank is averaged across six target tasks. As in Tabs. 1 and 2, weakNAS is the best in single-task setting and Arch-Graph-single achieves comparable results to weakNAS. On NB201, we conduct experiments on CIFAR-100 (Tab. 3) and set the budgets to 150 models. Better than REA and RS, Arch-Graph has an average performance of 73.38% that outperforms BONAS. Although slighter lower than weakNAS, Arch-Graph has a much larger kendall-rank coefficient (0.67) than weakNAS (0.49), indicating a better

[4]More comparisons with comparator-based sorting algorithms can be seen in Sec. 5 in Supplementary Materials

	Tasks	Cls.O.	Cls.S.	Auto.	Normal	Sem. Seg.	Room.	Jigsaw	Avg. Rank
	Metric	Acc.↑	Acc.↑	SSIM↑	SSIM↑	mIoU↑	L2 loss↓	Acc.↑	
Single NAS	RS [3]	45.16	54.41	55.94	56.85	25.21	61.48	94.47	85.61
	REA [26]	45.39	54.62	**56.96**	57.22	25.52	61.75	94.62	38.50
	BONAS [28]	45.50	54.46	56.73	57.46	25.32	61.10	94.81	34.31
	weakNAS [35]	45.66	54.72	56.77	57.21	**25.90**	60.31	94.63	20.03
	Arch-Graph-single	45.48	54.70	56.52	57.53	25.71	61.05	94.66	22.15
Transfer NAS	DT	42.03	49.80	51.20	55.03	22.45	66.98	88.95	935.12
	CATCH [4]	45.27	54.38	56.13	56.99	25.38	60.70	-	63.49
	REA-t [26]	45.51	54.61	56.52	57.20	25.46	61.04	-	40.14
	BONAS-t [28]	45.38	54.57	56.18	57.24	25.24	60.93	-	55.30
	nsganetv2 [19]	45.61	54.75	56.47	57.24	25.36	61.73	-	34.89
	weakNAS-t [35]	45.29	54.78	56.90	57.19	25.41	60.70	-	35.73
	Arch-Graph-zero	45.64	54.80	56.61	57.90	25.73	60.21		14.7
	Arch-Graph	**45.81**	**54.90**	56.58	**58.27**	25.69	**60.08**	-	**12.2**
	Global Best	46.32	54.94	57.72	59.62	26.27	59.38	95.37	1

↑ indicates higher is better, ↓ indicates lower is better, **bold** indicates the best result.

Table 2. Performance comparisons between different NAS methods and our Arch-Graph on Micro level search space. Jigsaw results are omitted for TransferNAS methods because it is used as the pretrain task.

ordering of the whole model space.

Task-Transferrable NAS. The transferred version of weakNAS and BONAS are also pretrained on jigsaw with a budget of 50 models. After initializing the predictors, we sample another 50 models to finetune the GCN embedding extractor and Bayesian Sigmoid Regression in BONAS and sets of the weak predictors in weakNAS on the target task. In addition to the searched models' accuracy, we also report the model rank in the search space, averaged across 6 targeted tasks (Tab. 3). Our Arch-Graph shows great superiority over both single task methods and transferable NAS methods when transferring knowledge from a pre-trained predictors, surpassing weakNAS [35] by average model rank 10.19 on macro level search space and 23.53 on micro level search space. It takes at least 60% extra samples for other methods to achieve comparable results, in Tab. 3.

To better illustrate the effectiveness of our Arch-Graph, in Fig. 4, we show the visualization result of the predicted top 50 models in the macro level search space on two tasks. More visualizations of search results on other tasks can be found in S.M.[2]. We first use t-SNE to project the model into the 2-dimensional space and colors to indicate model performance. The shallower the color, the stronger the model. In this projection, top models for each task tend to form local clusters. WeakNAS and Arch-Graph-zero can both attend to local optima, whereas Arch-Graph's predictions are significantly closer to globally optimal architectures.

4.3. Ablation Study

Task embedding. Some works on transferable NAS [34] also propose to use task embeddings to guide the search when facing different tasks. However, they use a randomly initialized embedding to represent each task and it is learned jointly with the NAS model's parameters. We verify the effectiveness of our task embedding defined in Sec. 3. We compare our task embedding with randomly initialized vectors for each task's embedding. We show the averaged architecture rank over 6 target tasks, with experiments repeated over 5 random seeds in Tab. 4. The performance using randomly initialized task embedding is highly unstable, resulting in a much larger variance (0.63 vs 692.03) and a significantly lower average performance (24.13) compared to Task2Vec (5.24), indicating a randomly initialized task embedding can't guarantee a stable knowledge transfer.

	Methods	τ↑	ρ↑	#budgets↓
TB101	BONAS [28]	0.26	0.38	100+
	BONAS-t [28]	0.24	0.34	100+
	nasganetv2 [19]	0.19	0.28	100+
	weakNAS [35]	0.36	0.51	80
	weakNAS-t [35]	0.16	0.24	100
	Arch-Graph-zero	0.58	0.76	60
	Arch-Graph	**0.61**	**0.79**	**50**
	Methods	Acc.↑	τ↑	ρ↑
NB201	RS [3]	71.80	-	-
	REA [26]	72.70	-	-
	BONAS [28]	72.84	0.43	0.60
	weakNAS [35]	**73.42**	0.49	0.56
	Arch-Graph	73.38	**0.67**	**0.79**

Table 3. Comparison of different methods on TransNAS-Bench-101 and NAS-Bench-201 benchmarks. τ, ρ are Kendall rank coefficient, Pearson correlation coefficient respectively. #budgets indicates the number of architectures for a method to find top 0.3% architectures in the macro level search space.

MWAS. Obtaining the approximation of the Maximal

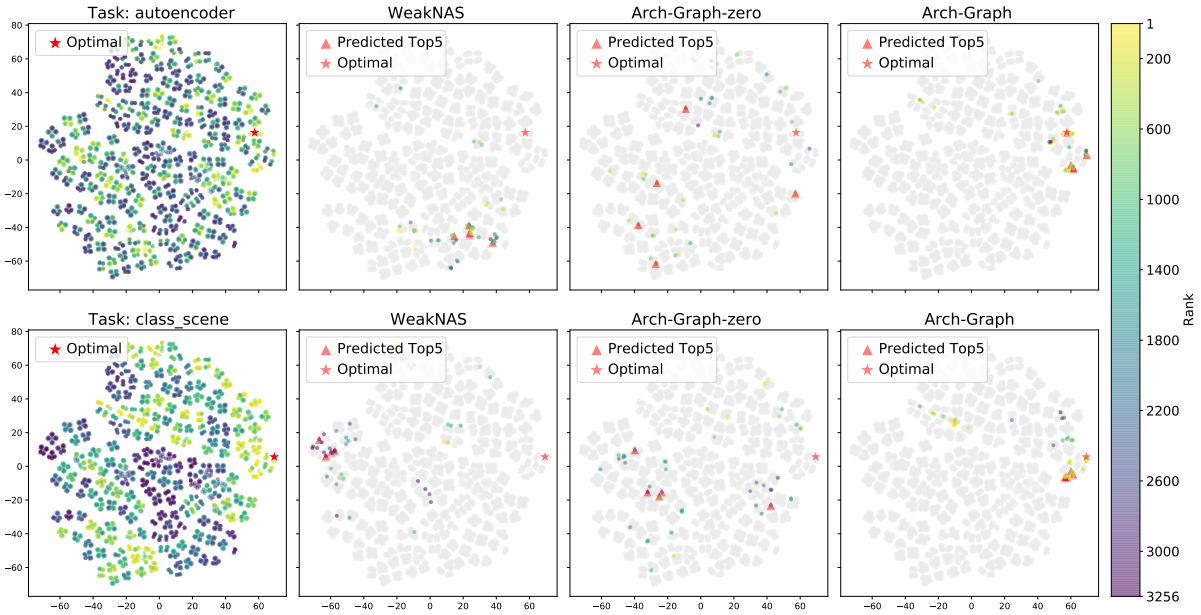

Figure 4. Visualization of the network search space on object classification and autoencoding tasks. For each algorithm, we color its predicted top-50 models and grey out everything else. We use triangles to mark each algorithm's top-5 prediction, and use stars to label the search space's global optima.

Average rank	Mean	Variance
Ours	5.24	0.63
Random	24.13	692.03

Table 4. Searched network's rank comparison by two embedding methods on Arch-Graph (lower is better).

Weighted Acyclic Subgraph Problem is a central component of our model to improve graph construction. To show its advantages over Arch-Graph-zero, we first pick 20 fine-tuned predictors on each task with the highest validation accuracy among the b_v validation architectures. We then compare the predicted accuracy between Arch-Graph-zero and Arch-Graph. Arch-Graph can identify better models than Arch-Graph-zero, which on average improves the rank by 3.14 and 5.28 on the macro and micro search space, respectively. More detailed differences of these top models can be found in Tabs. 1 and 2.

Arch-Graph-single. To verify the effect of knowledge transfer from source tasks to new target tasks, we compare the performance of Arch-Graph and Arch-Graph-single and fix the total budget to 50 models. Compared to transferring knowledge from a pretrained predictor, Arch-Graph-single is worse than Arch-Graph as shown in Tabs. 1 and 2. It shows the effectiveness of knowledge transfer from predictor trained on a previous task.

5. Conclusions and Discussions

In this work, we propose Arch-Graph, a task-transferable NAS method that formulate NAS as a graph ordering problem on an architecture relation graph. Directed edges of this graph are obtained through training a pairwise relation predictor with knowledge transfer. With extensive experiment, we demonstrate Arch-Graph's transferability and sample efficiency over many other NAS methods.

Potential negative societal impact. We have not identified any potential negative social impact. All the datasets we use are public and conform with ethical standards.

Limitation and Future Work. With Arch-Graph-zero, it is possible to exclude the ground truth global optima before the MWAS calculation. Future work could explore along this direction and construct subgraphs more efficiently for ranking. For example, the pairwise relation predictor training and the MWAS calculation can be done in an iterative style, so that we can progressively shrink the search space and improve the performance.

6. Acknowledgements

This work was supported in part by National key R&D Program of China under Grant No.2020AAA0109700, National Natural Science Foundation of China (NSFC) No.61976233, Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Guangdong Outstanding Youth Fund (Grant No. 2021B1515020061), Shenzhen Fundamental Research Program (Project No.RCYX20200714114642083, No.JCYJ20190807154211365).

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6429–6438. IEEE, 2019. 2, 4

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 1

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012. 6, 7

[4] Xin Chen, Yawen Duan, Zewei Chen, Hang Xu, Zihao Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. CATCH: context-based meta reinforcement learning for transferrable architecture search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, volume 12364 of *Lecture Notes in Computer Science*, pages 185–202. Springer, 2020. 3, 6, 7

[5] Aleksandar Cvetkovic and Vladimir Yu. Protasov. Maximal acyclic subgraphs and closest stable matrices. *SIAM J. Matrix Anal. Appl.*, 41(3):1167–1182, 2020. 5

[6] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 6

[7] 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, virtual, June 19-25, 2021*, pages 5251–5260. Computer Vision Foundation / IEEE, 2021. 3, 5, 6

[8] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10480–10490. Curran Associates, Inc., 2020. 3, 5

[9] Lukasz Dudziak, Thomas C. P. Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. BRP-NAS: prediction-based NAS using gcns. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[10] Lei Feng, Senlin Shu, Nan Lu, Bo Han, Miao Xu, Gang Niu, Bo An, and Masashi Sugiyama. Pointwise binary classification with pairwise confidence comparisons. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3252–3262. PMLR, 18–24 Jul 2021. 3

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017. 5

[12] Heinrich Jiang, Been Kim, Melody Y. Guan, and Maya R. Gupta. To trust or not to trust A classifier. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5546–5557, 2018. 5

[13] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. 5

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 4

[15] Hayeon Lee, Eunyoung Hyung, and Sung Ju Hwang. Rapid neural architecture search by learning to generate graphs from datasets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 3

[16] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1986–1995. Computer Vision Foundation / IEEE, 2020. 2

[17] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12261–12271. IEEE, 2021. 2

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2

[19] Zhichao Lu, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture

search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2020. 6, 7

[20] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh D. Dhebar, Kalyanmoy Deb, Erik D. Goodman, and Wolfgang Banzhaf. NSGA-NET: A multi-objective genetic algorithm for neural architecture search. *CoRR*, abs/1810.03522, 2018. 2

[21] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture search with GBDT. *CoRR*, abs/2007.04785, 2020. 2

[22] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7827–7838, 2018. 2, 3

[23] Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based NAS. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIII*, volume 12358 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2020. 2, 3

[24] Ramakanth Pasunuru and Mohit Bansal. Continual and multi-task architecture search. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1911–1922. Association for Computational Linguistics, 2019. 3

[25] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4092–4101. PMLR, 2018. 2

[26] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4780–4789. AAAI Press, 2019. 2, 6, 7

[27] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR, 2017. 2

[28] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T. Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with BONAS. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2, 6, 7

[29] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4778–4787. PMLR, 2018. 2

[30] 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, Long Beach, CA, USA, June 16-20, 2019*, pages 2820–2828. Computer Vision Foundation / IEEE, 2019. 2

[31] Renbo Tu, Mikhail Khodak, Nicholas Roberts, and Ameet Talwalkar. Nas-bench-360: Benchmarking diverse tasks for neural architecture search. *CoRR*, abs/2110.05668, 2021. 3

[32] Chen Wei, Chuang Niu, Yiping Tang, and Jimin Liang. NPENAS: neural predictor guided evolution for neural architecture search. *CoRR*, abs/2003.12857, 2020. 2

[33] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Helen Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374 of *Lecture Notes in Computer Science*, pages 660–676. Springer, 2020. 2, 3

[34] Catherine Wong, Neil Houlsby, Yifeng Lu, and Andrea Gesmundo. Transfer learning with neural automl. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8366–8375, 2018. 1, 7

[35] Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Stronger nas with weaker predictors. *arXiv preprint arXiv:2102.10490*, 2021. 2, 6, 7

[36] Lingxi Xie and Alan L. Yuille. Genetic CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1388–1397. IEEE Computer Society, 2017. 2

[37] Yixing Xu, Yunhe Wang, Kai Han, Yehui Tang, Shangling Jui, Chunjing Xu, and Chang Xu. Renas: Relativistic eval-

uation of neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4411–4420. Computer Vision Foundation / IEEE, 2021. 2, 3

[38] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2423–2432. Computer Vision Foundation / IEEE Computer Society, 2018. 2

[39] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 2

[40] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1

[41] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. Computer Vision Foundation / IEEE Computer Society, 2018. 2