Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap

LINGXI XIE, XIN CHEN, KAIFENG BI, and LONGHUI WEI, Huawei Inc.
YUHUI XU, Shanghai Jiao Tong University
LANFEI WANG, Beijing University of Posts and Telecommunications
ZHENGSU CHEN, AN XIAO, JIANLONG CHANG, XIAOPENG ZHANG, and QI TIAN, Huawei Inc.

Neural architecture search (NAS) has attracted increasing attention. In recent years, individual search methods have been replaced by weight-sharing search methods for higher search efficiency, but the latter methods often suffer lower instability. This article provides a literature review on these methods and owes this issue to the optimization gap. From this perspective, we summarize existing approaches into several categories according to their efforts in bridging the gap, and we analyze both advantages and disadvantages of these methodologies. Finally, we share our opinions on the future directions of NAS and AutoML. Due to the expertise of the authors, this article mainly focuses on the application of NAS to computer vision problems.

CCS Concepts: • Computing methodologies → Machine learning approaches;

Additional Key Words and Phrases: AutoML, neural architecture search, weight-sharing, super-network, optimization gap, computer vision

ACM Reference format:
Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Lanfei Wang, Zhengsu Chen, An Xiao, Jianlong Chang, Xiaopeng Zhang, and Qi Tian. 2021. Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap. ACM Comput. Surv. 54, 9, Article 183 (October 2021), 37 pages.
https://doi.org/10.1145/3473330

1 INTRODUCTION

The rapid development of deep learning [169] has claimed its domination in the area of artificial intelligence. In particular, in the computer vision community, deep neural networks have been successfully applied to a wide range of challenging problems including image classification [124, 137, 166, 279, 299], object detection [103, 199, 258, 260], semantic segmentation [38, 123, 202], boundary detection [350], pose estimation [309], and so on, and surpassed conventional approaches based on hand-designed features (e.g., SIFT [206]) by significant margins.

Authors’ addresses: L. Xie, X. Chen, K. Bi, L. Wei, Z. Chen, A. Xiao, J. Chang, X. Zhang, and Q. Tian, Huawei Inc.; emails: 198808xc@gmail.com, {chenxin180, bikaifeng1, weilonghu1, chenzhengsu1, xiaoan1, jianlong.chang1}@huawei.com, zxphistory@gmail.com, tian.q1@huawei.com; Y. Xu, Shanghai Jiao Tong University; email: yuhuixu@sjtu.edu.cn; L. Wang, Beijing University of Posts and Telecommunications; email: wanglanfei@bupt.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
0360-0300/2021/10-ART183 $15.00
https://doi.org/10.1145/3473330

ACM Computing Surveys, Vol. 54, No. 9, Article 183. Publication date: October 2021.
Despite the success of these efforts, researchers quickly encountered the difficulty that the network design space has become much more complex than that in the early deep learning era. Several troubles emerged. For example, the manually designed networks do not generalize to various scenarios, e.g., a network designed for holistic image classification may produce inferior performance in dense image prediction tasks (semantic segmentation, image super-resolution, etc.). In addition, it is sometimes required to design networks under specific hardware constraints (e.g., FLOPs, memory, latency, etc.), and manual design is often insufficient to explore a large number of possibilities. Motivated by these factors, researchers started to seek for the solution that can flexibly adjusted to different scenarios in minimal efforts.

**Automated machine learning** (AutoML [305]) appeared recently as a practical tool that enables researchers and even engineers without expertise in machine learning to use off-the-shelf algorithms conveniently for different purposes. As a subarea of AutoML, **neural architecture search (NAS)** focuses on designing effective neural networks in an automatic manner [257, 347, 414]. By defining a large set of possible network architectures (the set is often referred to as the search space), NAS can explore and evaluate a large number of networks that have never been studied before. Nowadays, NAS has achieved remarkable performance gain beyond manually designed architectures, in particular, in the scenarios that computational costs are constrained, e.g., in the mobile setting [131] of ImageNet classification.

In the early age, NAS algorithms often sampled a large number of architectures from the search space and then trained each of them from scratch to validate its performance. Despite their effectiveness in finding high-quality architectures, these approaches often require heavy computational overhead (e.g., tens of thousands of GPU days [414]), which hinders them from being transplanted to various applications. To alleviate this issue, researchers proposed to re-use the weights of previously optimized architectures [24] or share computation among different but relevant architectures [245] sampled from the search space. This direction eventually led to the idea that the search space is formulated into an over-parameterized super-network, so that the sampled sub-architectures get evaluated without additional optimization [23, 195]. Consequently, the search costs have been reduced by several orders of magnitudes, e.g., recent advances allow an effective architecture to be found within 0.1 GPU-days on CIFAR10 [358] or 2.0 GPU-days on ImageNet [21].

This article focuses on the weight-sharing methodology for NAS, and investigate the critical weakness of it, **instability**, which refers to the phenomenon that individual runs of search can lead to very different performance. We owe the instability to the optimization gap between the super-network and its sub-architectures—in other words, the search procedure tries to optimize the super-network, but a well-optimized super-network does not necessarily produce high-quality sub-architectures. This is somewhat similar to the well-known over-fitting problem in machine learning, and thus can be alleviated from different aspects. We try to summarize the existing efforts into a unified framework, and thus we can answer the questions of where the community is, what the main difficulties are, what to do in the near future.

We noticed previous surveys [87, 125, 147, 259, 300, 337] that used three factors (i.e., search space, search strategy, evaluation method) to categorize NAS approaches. We inherit this framework but merges the latter two factors, since they are highly coupled in recent works. Differently, this article focuses on the weight-sharing methodology and investigate it from the perspective of shrinking the optimization gap. As a disclaimer, this article discusses the modern neural architecture search methods, i.e., in the context of deep learning, so we do not cover the early work on finding efficient neural networks [14, 72, 288, 367].

The remainder of this article is organized as follows. Section 2 first formulates NAS into the problem of finding the optimal sub-architecture(s) in a super-network, and introduces both individual and weight-sharing approaches to explore the search space efficiently. Then, Section 3 is the
main part of this article, in which we formalize the optimization gap to be the main challenge of weight-sharing NAS, based on which, we review a few popular but preliminary solutions to shrink the gap. Next, in Section 4, we discuss a few other directions that are related to NAS. Finally, in Section 5, we look into the future and put forward a few open problems in this research field.

To complete this survey, we collected more than 500 papers from https://www.automl.org/automl/literature-on-neural-architecture-search/ (as the version of July 31, 2020, one month prior to the article submission). We skimmed these papers one by one, and removed those papers of relatively low quality. We also found ∼20 important papers that fall in the scope of NAS but were not in the original list. This made up the final list that contains ∼300 papers.

2 NEURAL ARCHITECTURE SEARCH

2.1 Overall Framework

In this section, we provide a generalized formulation for NAS and build up a notation system that will be used throughout this article. The background of NAS is identical to regular machine learning tasks, in which a dataset \( D \) with \( N \) elements is given, and each element in it is denoted by an input-output pair \((x_n, y_n)\). The goal is to find a model that works best on the dataset and, more importantly, transfers well to the testing scenario. For later convenience, we use \( D_{\text{train}}, D_{\text{val}}, \) and \( D_{\text{test}} \) to indicate the training, validation, and testing subsets of \( D \), respectively. There are some standard datasets for NAS. For image classification, the most popular datasets are CIFAR10 \(^{[165]}\) and ImageNet-1K \(^{[69, 263]}\). For natural language modeling, the most popular ones are Penn Treebank and WikiText-2 \(^{[223]}\).

NAS starts with defining a search space, \( S \). Each element \( S \in S \) is a network architecture that receives an input \( x \) and delivers an output \( y \), which we denote as \( y = f^S(x; \omega) \) where \( \omega \) denotes the network weights (e.g., in the convolutional layers). Note that the form and dimensionality of \( \omega \) depends on \( S \). We use \( \langle S, \omega^*(S) \rangle \) to denote an architecture equipped with the optimal network weights. The goal of NAS is to find the optimal architecture \( S^* \), which produces the best performance (e.g., classification accuracy) on the testing set:

\[
S^* = \arg\max_{S \in S} \text{Eval}(\langle S, \omega^*(S) \rangle; D_{\text{test}}),
\]

where \( \text{Eval}(\langle S, \omega^*(S) \rangle; D_{\text{test}}) \) is the function that measures how \( \langle S, \omega^*(S) \rangle \) behaves on the testing set, \( D_{\text{test}} \). But, in real practice, \( D_{\text{test}} \) is hidden in the search procedure, so the training and validation sets are often used instead, and thus the objective becomes

\[
S^* = \arg\max_{S \in S} \text{Eval}(\langle S, \omega^*(S) \rangle; D_{\text{val}}),
\]

\[
\text{s.t. } \omega^*(S) = \arg\min_{\omega} \mathcal{L}(\omega(S); D_{\text{train}}),
\]

where \( \mathcal{L}(\omega(S); D_{\text{train}}) \) is the loss function (e.g., cross-entropy) computed on the training set.

Most often, the search space \( S \) is sufficiently large to enable powerful architectures to be found. Therefore, it is unlikely that an exhaustive search method can explore the large space with reasonable computational costs. Therefore, a practical solution is to use a heuristic search method, which defines \( \mathcal{B} \), a probabilistic distribution on \( S \), and adjusts the probability, \( p(S) \), that an element \( S \) is sampled according to the rewards from those already sampled architectures. This is to say, there is a more important factor, the search strategy, which determines the way of adjusting the probabilistic distribution. Note that each search strategy is equipped with an evaluation method that judges the quality of a sampled architecture.

In what follows, we briefly review the popular choices of the search space and search strategy.
Left: An example of the search space. For simplicity, we show a macro-architecture where each cell is allowed to receive input from two precursors, and a micro-architecture (cell) where the type and existence of the inter-node connections are searchable.

Right: The relationship between four representative search spaces introduced in Sections 2.2.2–2.2.5, respectively. For the two most popular search spaces to date, MobileNet and DARTS, we summarize the advantages and disadvantages below, and point out some extensions.

2.2 Search Space

The search space, $S$, determines the range of architecture search. Increasing the scale of the search space, $|S|$, is both intriguing and challenging. On the one hand, a larger space enables the search algorithm to cover more architectures so that the chance of finding a powerful architecture is increased. On the other hand, a larger search space makes the search algorithm more difficult to converge, because $\text{Eval}(S, \omega; D_{val})$ can be a very complicated function with respect to $S$—more importantly, there is no guarantee of any of its properties in optimization. Hence, the number of samples required for depicting the function is often positively correlated to $|S|$.

There exist many types of search spaces. There are often two steps of constructing a search space, namely, the macro- and micro-architectures of the space. As shown in Figure 1, the macro-architecture determines the overall backbone of the network, and the micro-architecture determines the details of each network unit. Sometimes, the micro-architectures are referred to as cells and the number of cells can vary between the search and re-training phases or to adjust different tasks. Each NAS algorithm searches for either the macro- or micro-architecture but not necessarily both of them. Today, most NAS methods fix the macro-architecture and investigate better micro-architectures, which largely constrains the flexibility of the search space.

In what follows, we instantiate the macro- and micro-architectures using some popular examples. Note that some methods considered different search spaces for vision and language modeling, and we will mainly focus on the vision part for which the search space is often more complicated. The relationship among these popular search spaces is illustrated in Figure 1.

2.2.1 The EvoNet Search Space. EvoNet [257] is one of the first works that investigated an open search space, i.e., the architectures are not limited within the combination of a fixed number of components, and instead, the architecture is allowed to grow arbitrarily with requirement. Thus, the search space is latently defined by a set of actions on the architecture, e.g., inserting a convolutional layer to a specific position, altering the kernel size of a convolutional layer, adding a skip connection between two layers, and so on, and some actions can be reversely performed, e.g., removing the convolutional layer from a specific position.
In an open search space, there is not a fixed number of architectures, because the network shape (e.g., depth) is only constrained by the hardware platform—even provided the length of the search procedure and the hardware limit, the accurate space size is difficult to estimate. A very loose estimation is obtained by assuming that the search procedure starts with a fixed architecture and has $T$ steps in total. Thus, if each step can take up to $C$ actions, then the search space size satisfies $|S| \leq 1 + C + C^2 + \ldots + C^T$.

Other open search spaces. There are other efforts that tried to explore an open space [194, 324]. As we shall explain in Section 5, open-space search is likely to be the future trend of NAS [256], but the community needs a robust search strategy that can efficiently explore the challenging spaces.

2.2.2 The NAS-RL Search Space. NAS-RL [414] is the work that defined the modern formulation of NAS. It also sculptured the framework that uses the macro- and micro-architectures to determine the network. The densely connected macro-architecture has $N$ layers, each of which can be connected to any of its precursors. Besides the direct precursor, other connections can be present or absent, bringing a total of $2^{(N-1)(N-2)/2}$ possibilities. Each layer has a micro-architecture involving the hyper-parameters of the filter height, filter width, stride height, stride width, and the number of filters. For simplicity, each hyper-parameter is chosen from a pre-defined set, e.g., the number of filters is one among $\{24, 36, 48, 64\}$. Let $C$ be the number of combinations for each layer, then the total number of architectures is $|S| = C^N \times 2^{N(N-1)/2}$. In the original paper, $C$ varies with different search options and the largest $C$ is 576.

The search space for language modeling. The goal is to construct a cell that connects the neighboring states of a recurrent network, in which there are three input variables and two output variables. Similarly, the recurrent controller is used to determine the architecture with a fixed number of intermediate nodes. For each node that is related to the input data and the hidden state, there is an operator (e.g., add, elem-mult, etc.) and an activation function (e.g., tanh, ReLU, sigmoid, etc.), and the controller also determines how to connect the memory states to the temporary variables inside the tree.

Related search spaces. Genetic-CNN [347] shared the similar ideology of building the architecture in a macro-then-micro manner. The architecture is composed of a fixed number of stages and each stage has a fixed number of layers. Pooling layers are inserted between neighboring stages. Only the regular $3 \times 3$-convolution is used, but the inter-layer connectivity in the same stage can be searched. The number of architectures in each stage is $2^{N(N-1)/2}$ where $N$ is the number of layers.

2.2.3 The NASNet Search Space. NASNet [415] introduced the idea that the macro-architecture contains repeatable cells, so that the search procedure is performed within a relatively shallow network but the final architecture can be much deeper. In addition, the densely connected macro-architecture is simplified into bi-chain-styled, i.e., each layer steadily receives input from two precursors. Regarding the micro-architecture, each cell contains $N$ hidden nodes, indexed from 0 to $N-1$, and the two input nodes are indexed as $-2$ and $-1$. Each hidden node is connected to two nodes with smaller indices, with the operator of each connection chosen from a pre-defined set of $C_1$ candidates (e.g., dil-conv-$3 \times 3$, sep-conv-$5 \times 5$, identity, max-pool-$3 \times 3$, etc.) and the summarization function chosen from a pre-defined set of $C_2$ candidates (e.g., sum, concat, product, etc.). The outputs of all intermediate nodes are concatenated into the output of the cell. $C_1$ is 13 in NASNet, but is often reduced to a smaller number in the follow-up methods, because researchers noticed that there exist some weak operators that are rarely used (e.g., max-pool-$7 \times 7$ and $1 \times 7$-then-$7 \times 1$-conv), plus, shrinking the search space can reduce the cost of search. We denote $C_3$ as
the number of different topologies in a cell, which is computed via \( \binom{2}{2} \times \binom{3}{2} \times \ldots \times \binom{N+1}{2} \). The number of architectures in a normal cell or a reduction cell is thus \((C_1^2 \times C_2)^N \times C_3\).

**Related search spaces.** Many follow-up methods have used the same search space but simplified either the candidate set or the cell topology. For example, PNASNet [193] and AmoebaNet [255] reduced \( C_1 \) to 8, and DARTS [195] reduced \( N \) from 5 to 4. In addition, DARTS did not use individual summarization functions for different nodes and used sum as the default choice.

### 2.2.4 The DARTS Search Space

DARTS [195] has a very similar search space to NASNet and follows a weight-sharing search pipeline (see Section 2.3) so that a flexible super-network is constructed. In the DARTS space, the set of candidate operators have been defined as sep-conv-3 \( \times \) 3, dil-conv-3 \( \times \) 3, dil-conv-5 \( \times \) 5, max-pool-3 \( \times \) 3, avg-pool-3 \( \times \) 3, and skip-connect. A dummy zero operator is added to the search procedure but is not allowed to appear in the final architecture. To ease the formulation of architecture search, each layer receives input from an arbitrary number of previous layers in the same cell though only two of them are finally preserved. Provided that 4 intermediate cells are present, the number of architectures for both the normal and reduction cells is \((\binom{2}{2})^4 \times (\binom{3}{2})^5 \times 7^8 \approx 1.0 \times 10^9\), and the entire space has around \(1.1 \times 10^{18}\) architectures.

**The search space for language modeling.** Similar to NAS-RL, the DARTS space is also used for finding an improved recurrent cell in the language models. The input sources, the number of intermediate nodes, and the connectivity between these nodes are adjusted accordingly, but still, the outputs of all intermediate nodes are concatenated.

**Related search spaces.** The super-network formulation of DARTS has offered opportunities to create some new search spaces. For example, one can shrink the search space by reducing the number of operators (e.g., only sep-conv-3 \( \times \) 3 and skip-connect used [377]), or expanding the search space by enabling multiple operators to be preserved in each edge [55, 404], adding new operators [323], allowing the cells of the same type to be individual from each other [20], and relax the constraints that each layer receives input from two previous layers [21].

### 2.2.5 The MobileNet Search Space

An alternative way [28, 341] of simplifying the NAS-RL search space is to borrow the design of MobileNet. The macro-architecture is set to be chain-styled, i.e., each layer receives input from its direct precursor. The micro architecture is also constrained to be one of the MobileNet (MB) cells [131]. Changeable options include the type of separable-convolution [56] (the regular convolution can be replaced by de-convolution), the type of skip-layer connection (can be either identity or some kinds of pooling), the number of channels, the kernel size, the expansion ratio (how the intermediate layer of the MB cell magnitudes the number of basic number of channels). Beyond these, the number of layers within each cell (or referred to as a block) can also be searched [90, 301]. So, the number of architectures in each cell is basically \( C^1 + C^2 + \ldots + C^L \), where \( C \) is the number of combinations of each layer and \( L \) is the maximal number of layers in a cell. The above number is further powered up by \( N \), the number of cells.

**Related search spaces.** MNASNet [301] was the predecessor of the MobileNet search space, where there are more searchable options including the number of cells in each stage, i.e., \( L \) is relatively large. After that, researchers soon noticed that (i) there are duplicates in using different numbers of layers in neighboring cells and (ii) preserving a small number of MB cells is sufficient to achieve good performance. So, a few follow-up methods [60, 130, 302] have reduced \( L \) into 1 and reduced \( C \) into 6 (i.e., MB cells with a 3 \( \times \) 3, 5 \( \times \) 5, or 7 \( \times \) 7 convolution, and an expansion ratio
of 3 or 6). The MobileNet space is also friendly to augmentation, including introducing mixed convolution [39, 220, 303], allowing dense connections [90, 113], or adding attention blocks [183].

2.2.6 Other Search Spaces. Besides the previous search spaces that were widely explored, there exist other interesting methods. For example, some early efforts have either applied the technique of network transformation [40] or morphism [329] to generate complex architectures from the basic one [24, 187, 224], or made use of shared computation to combine exponentially many architectures with cheap computation [268, 322, 395]. Researchers have also designed other methods for architecture design such as repeating one-of-many choices [8, 64] or defining a language for encoding [230, 231]. However, in the current era, increasing the search space arbitrarily seems not the optimal choice, since the search strategy is not sufficiently robust and may produce weak results that are on par with random search [349].

There are efforts in applying NAS methods to other tasks, for which the search spaces need to adjust accordingly. For object detection, there are different kinds of efforts toward designing a powerful architecture, including (i) transferring powerful backbones in classification [304], (ii) searching for a new backbone [33, 48, 82], (iii) considering the way of fusing multi-stage features [102, 355], and (iv) integrating multiple factors into the search space [110, 365]. Regarding semantic segmentation in which spatial resolution is more sensitive [291], the most interesting part seems to design of the down-sampling and up-sampling path [191, 434], though searching for efficient blocks is also verified useful [189, 273, 330]. NAS is also applied for image generation, in particular, designing network architectures for the generative adversarial networks (GAN) [98, 104, 320]. Recently, this method has been extended to conditional GAN where the generator varies among classes and thus the search space becomes exponentially larger [406]. Beyond the search spaces that are extendable (see Sections 2.2.2 and 2.2.4), there were efforts that directly search for powerful architectures for RNN design [141, 238, 254, 271] or language modeling [36, 282, 326, 339]. Table 1 provides a summary of popular search spaces.

2.3 Search Strategy and Evaluation Method

We combine the review of search strategies and evaluation methods into one part, because these two factors are closely correlated, e.g., the individual heuristic search strategies often lead to an separate evaluation on each sampled architecture, while the weight-sharing search strategies often integrate architecture optimization into architecture search.

2.3.1 Individual Heuristic Search Strategies. The most straightforward idea of architecture search is to enumerate all possible architectures from the search space and evaluate them individually. However, since the search space is often very large so that exhaustive search is intractable, researchers turned to heuristic search. The fundamental idea is to make use of the relationship between architectures, so that if an architecture reports good/bad performance, it is natural to tune up/down the probability of sampling its close neighbors. The algorithm starts with formulating a function, \( p : S \mapsto \mathbb{R} \), so that each architecture \( s \in S \) in the search space corresponds to a probability density that it is sampled, and the function gets updated after the algorithm gets rewards from the evaluation result of the sampled architectures.

There are two popular choices of the heuristic algorithm, both of which maintain a latent form of the probabilistic distribution \( p(\cdot) \). The first type, named the genetic algorithm or evolutionary algorithm, constructs a family of architectures during the search procedure [257, 347], and allows each architecture (or named an individual) to generate other architectures sharing similar properties. Typical genetic operations include crossover, mutation, selection or elimination, and so on. The other type follows a different path, known as reinforcement learning [8, 414, 415], which constructs an architecture using a learnable policy, and updates the parameters in the policy.
| Search Space/Task       | Important Work | Other Work |
|-------------------------|----------------|------------|
| EvoNet [257] and open   | [194, 257]     | [256]      |
| NAS-RL [414]           | [414]          | [13, 409]  |
| NASNet [415]           | [16, 193, 211, 245, 255, 402, 415] | [1, 3, 6, 46, 49, 112, 168, 187, 198, 215, 241, 243, 266, 276, 325, 336, 387, 389, 403, 411] |
| DARTS [195]            | [44, 78, 79, 115, 177, 186, 195, 229, 351, 358, 377, 404] | [20, 30–32, 55, 62, 73, 107, 129, 133, 142, 155, 170, 173, 174, 209, 237, 324, 331, 352, 364, 366, 384, 399, 400] |
| MobileNet [28, 341]    | [25, 28, 60, 67, 116, 286, 301–303, 341] | [88, 90, 117, 273, 338, 354] |
| Other Spaces           | [8, 9, 23, 24, 64, 86, 158, 230, 287, 290, 347, 349] | [10, 27, 50–53, 70, 74, 84, 85, 95, 101, 120, 132, 145, 146, 150, 152, 153, 156, 157, 203, 208, 213, 216, 235, 245, 250, 262, 265, 267, 275, 277, 280, 293–297, 316–319, 328, 332, 334, 335, 342, 380, 397, 412] |
| Object Detection       | [48, 102, 304] | [33, 89, 110, 148, 149, 353, 355, 365] |
| Semantic Segmentation  | [37, 191]      | [89, 189, 232–234, 340, 391] |
| Generative Models      | [104]          | [97, 98, 178, 307, 308, 406] |
| Medical Image Analysis | [330]          | [7, 11, 75, 109, 161, 225, 361, 373, 394, 413] |
| Adversarial Learning   | –              | [75, 111, 313] |
| Low-level Vision       | –              | [58, 61, 114, 171, 179, 197, 226, 283, 311, 381] |
| Video Processing       | –              | [242, 246, 248, 264] |
| Multi-task Learning    | –              | [99, 201, 244, 375] |
| Language Modeling      | [282]          | [34, 36, 151, 182, 217, 218, 314, 326, 339] |
| Speech Recognition     | –              | [12, 71, 176] |
| Graph Networks         | –              | [100, 398, 405] |
| Recommendation         | –              | [54, 180, 284, 396] |
| Miscellaneous          | [224] (captioning) | [139] (attribute), [249, 385] (person re-id), [184] (inpainting), [363] (pose), [388] (reconstruction), [393] (knowledge graph), [4] (style transfer), [22] (saliency), [386] (privacy), [128] (deep image prior), [135] (counting), [383] (text recognition), [164] (discussion) |

Note that (i) there are two categorization methods, i.e., by space configuration (NAS-RL, NASNet, MobileNet, DARTS, open, others) and by non-classification task (detection, segmentation, etc.), and (ii) one work may appear in multiple rows, because it deals with more than one space or tasks. We use the number of Google scholar citations and the number per time unit (due July 31, 2020) to judge if this work is an “important work” (around 10% top-cited papers are considered important).

According to the final performance. There have been efforts in combining the evolutionary algorithm and reinforcement learning for NAS [46], or comparing the efficiency between these two options [217] and/or against the random search baseline [255].

Regardless of the heuristic algorithm being used, such search methods suffer the problem of heavy computational overhead. This is mainly due to the need of re-training each sampled architecture from scratch. There are a few standard ways of alleviating the computational burden in the search procedure, including using a smaller number of channels, stacking fewer cells, or exploring...
on a smaller dataset. However, it can still take long for the algorithm to thoroughly traverse the search space and find a satisfying architecture, in particular, when the training dataset is large.

**The random search baseline.** When the probability function does not change with the evaluation results, the individual heuristic algorithm degenerates to random search in which the probability of each architecture being sampled is equal and does not change with time. Note that random search can find any high-quality architecture given sufficient computational costs, but it is verified less efficient than heuristic search [255] or weight-sharing search [17] methods. Random search is widely used as a baseline test of search efficiency, e.g., evaluating the difficulty of the search space and thus validating the high cost-performance ratio of the proposed approach [21, 177, 195, 377].

### 2.3.2 Weight-sharing Heuristic Search Strategies

A smart solution of fast search lies in sharing computation among different architectures, as many of them have similar properties, e.g., beyond a well-trained architecture, only one layer is added or one layer is made wider. In these scenarios, it is possible to copy the weights of the trained network with a small part being transformed [24], so that the number of epochs required for optimizing the new network is significantly reduced.

Moving one step forward, an elegant solution of weight-sharing search is to construct the search space into a super-network [23, 245, 268], from which each architecture can be sampled as a sub-architecture, and different sub-architectures share corresponding modules. Consider a network that has a fixed topology with $L$ edges and each edge has $K$ options to be considered. The super-network contains $L K$ units to be optimized, but after the optimization, there are up to $K^L$ sub-architectures that can be sampled from the super-network. The search procedure is therefore accelerated, because much computation has been shared among the sub-architectures.

However, it is worth emphasizing that sampling a sub-architecture with the weights directly copied from the super-network does not reflect its accurate performance on the validation set, because the weight-sharing training procedure was focused on optimizing the super-network as an entire but not the individual sub-architectures. This difficulty is named the optimization gap, which we will formalize in Section 3.2. To alleviate the gap, typical efforts include fine-tuning the sub-architectures for more accurate performance prediction [245], facilitating the fairness during super-network training [60], and modeling the search space using graph-based methods [45].

**Relationship to individual heuristic search.** Note that, despite the super-network that organizes the search space in a more efficient manner, there still needs a standalone module to sample sub-architectures from the super-network. That is to say, either the evolutionary algorithm [60, 116] or the reinforcement learning algorithm [16, 245] can be used. Due to the effectiveness of sub-architecture sampling, it is also possible to split the search space into subspaces and enumerate all architectures in each of them [45].

### 2.3.3 Weight-sharing Differentiable Search Strategies

An alternative way of weight-sharing search is to allow a relaxed formulation of the super-network [195, 211, 277] so that it is differentiable to both the network weights (mainly the convolutional kernels and batch normalization [144] coefficients) and architectural parameters (modeling the likelihood that the candidate edges and/or operators are preserved in the final architecture). The most intriguing property of this formulation is that the architectural parameters can be optimized together with the network weights, so that at the end of the search procedure, the optimal sub-architecture is directly derived without the need of an additional sampling procedure. Therefore, the computational costs are further reduced.

The most popular example of differentiable search is DARTS [195], where the goal is to find the optimal sub-architecture for the normal and reduction cells as defined in Section 2.2.4. At the start, each edge is individually initialized so that all candidate operators have the same weight.
(e.g., $1/C$ where $C$ is the number of candidate operators). Then, during the search procedure, these architectural parameters are updated in a continuous space, so that the stronger (more effective) operators gradually take the lead. At the end of the search procedure, each edge undergoes a so-called discretization process, during which the strongest operator is preserved (and assigned with the weight of 1.0) while others are eliminated. The preserved sub-architecture (the architectural parameters are fixed) is fed into the re-training procedure for the optimal network weights.

**Relationship to weight-sharing heuristic search.** First, we emphasize that weight-sharing differentiable search also works on the MobileNet search space [28, 341], though it is more commonly used to explore the DARTS space. Essentially, differentiable search serves as an alternative way of approximately evaluating exponentially many architectures at the same time. The evaluation procedure is embedded into the search procedure and thus not explicitly performed. Therefore, the issue of optimization gap also exists in this framework. In particular, due to the inevitable approximation error of gradient computation, the differentiable search procedure can sometimes run into a dramatic failure [20, 186], and researchers developed various methods to alleviate it, including early termination [186, 377] or using progressive optimization [44, 173] to reduce the discretization error [306]. We will summarize these methods from another perspective in Section 3.3.

2.3.4 **Predictor-based Search Methods.** Besides the aforementioned search methods, there are also efforts in exhaustively enumerating all architectures in the search space (often relatively smaller) and then using them as the ground-truth to test the efficiency of different search strategies. The most popular benchmarks include NAS-Bench-101 [368] that contains ~423K architectures evaluated on CIFAR10 and NAS-Bench-201 [80] that contains 15,526 architectures evaluated on CIFAR10, CIFAR100, and ImageNet-16-120. We also noticed an extension to the NLP scope [163].

These benchmarks offer the opportunities of training predictors, i.e., the input is an encoded neural architecture and the output is the network accuracy [68]. There have been efforts of exploring the internal relationship between architectures (an effective way is to design a good encoding method for the architectures [96, 236, 332]) for better prediction performance, in particular, under a limited number of sampled architectures [276, 357]. The trained predictors can inspire algorithms beyond the toy search spaces [45, 236]. We will discuss more about this issue in Section 3.3.5.

Table 2 provides a summary of the frequently used search strategies, where we find that weight-sharing search is the current mainstream and future direction.

2.4 **Summary**

There is a close relationship between the search space and the search strategy together with evaluation method. When the search space is small and “tidy” (i.e., similar architectures are closely correlated in performance), individual heuristic search methods often work very well. However, when the search space becomes larger and more flexible, individual search methods often require heavy computational burden that is not affordable to most researchers and developers. Thus, since researchers believe that larger search spaces are the future trend of NAS, it has become an important topic to design efficient and stabilized weight-sharing search strategies. In the next part, we will owe the instability of weight-sharing architecture search to the optimization gap and discuss the solution to this problem.

As a side note, there have been research [251, 408] on designing the search space for architecture search or design. From the perspective of NAS, this is similar to the idea that gradually removing the candidate connections/operators during the search procedure [21, 44, 173], or undergoing a two-stage NAS algorithm that the network topology and operators are determined separately [20, 133, 373]. We carefully put forward the opinion that these multi-stage algorithms
reflect the instability and immaturity of NAS algorithms, and we advocate for an algorithm that can explore a very large search space in an end-to-end manner.

In evaluating the quality of the search space and/or algorithms, choosing a good benchmark as well as a set of proper hyper-parameters is very important for NAS research. We suggest the readers to refer to Reference [190] for more practices NAS research. Though, it was believed that evaluating the performance of NAS methods is often hard [272, 362, 372]. Different settings beyond supervised learning have been investigated in NAS, including semi-supervised learning [210], self-supervised learning [160], unsupervised learning [192], incremental learning [138], federated learning [121], and so on, showing the promising transferability of NAS methods. Last but not least, there are several toolkits for AutoML [154, 159, 305] that facilitate the reproducibility of NAS.

### 3 TOWARD STABILIZED WEIGHT-SHARING NAS

#### 3.1 An Alternative Formulation of Weight-sharing NAS

In this section, we delve deep into the weight-sharing NAS algorithms and try to answer the key question: why do these methods suffer search instability? Here, by *instability* we mean that the same algorithm can produce quite different results, in terms of either the searched architecture or the test accuracy, when it is executed for several times [177]. Weight-sharing NAS methods start with defining a super-network that contains all learnable parameters in the search space. This involves using a vectorized form, \( \alpha \), to encode each architecture \( \mathcal{S} \in \mathcal{S} \), so that the super-network is parameterized using both \( \alpha \) and \( \omega \). We follow the convention [195] to rewrite the super-network as \( y = F(x; \alpha, \omega) \). Throughout the remaining part of this article, we refer to \( \alpha \) and \( \omega \) as the architectural parameters and network weights, respectively. Though \( \alpha \) takes discrete values, a common practice is to slack the search space and correspond each element of \( \alpha \) to the probability that an operator appears in the final sub-architecture. Hence, finding
the optimal sub-architecture $S^*$ comes down to finding the optimal architectural parameters $\alpha^*$ and performing discretization (mapping the possibly continuous $\alpha^*$ to a discrete form that corresponds to a valid element of $S^*$), and thus Equation (2) becomes

$$S^* = \text{disc}(\alpha^*), \quad \alpha^* = \arg\min_{\alpha} \mathcal{L}(\omega^*(\alpha), \alpha; D_{\text{val}}),$$

subject to $\omega^*(\alpha) = \arg\min_{\omega} \mathcal{L}(\omega, \alpha; D_{\text{train}}),$

where $\mathcal{L}(\omega, \alpha; D) = \mathbb{E}_{(x, y^*) \in D} [\text{CE}(F(x; \alpha; \omega), y^*)]$ are the expected cross-entropy loss (computed in any training subset). We expect Eval$(S, \omega(S)|D_{\text{val}}) = 1 - \mathcal{L}(\omega, \alpha; D_{\text{val}})$ when $\alpha$ corresponds to $S$, but $\mathcal{L}(\omega, \alpha; D_{\text{train}})$ is more flexible in dealing with the continuous form of $\alpha$. Both $D_{\text{train}}$ and $D_{\text{val}}$ are subsets of the training data but they are not necessarily different, e.g., in one-level optimization methods $[21, 174]$, both $D_{\text{train}}$ and $D_{\text{val}}$ are the entire training set. The discretization function, $\text{disc}(\cdot)$, does not take an essential operation when the search space is not slacked, e.g., in the weight-sharing heuristic search, $\alpha$ does not take continuous values and $S^*$ is identical to $\alpha^*$.

Note that Equation (3) can be solved using non-weight-sharing methods where $\omega(\alpha)$ have individual weights for different $\alpha$. For weight-sharing methods, the key to optimize Equation (3) is to share $\omega(\alpha)$ over different $\alpha$ and perform the optimization with respect to $\omega$ and $\alpha$ over all sub-architectures. In other words, this is to achieve an amortized optimum over all sub-architectures and $\alpha$ plays the role of the amortizing distribution. With this formulation, we provide an alternative perspective in discriminating weight-sharing heuristic and differentiable search. The heuristic search methods often keep all elements of $\alpha$ identical, i.e., during training the super-network, the distribution $\Psi$ remains uniform over the entire search space. A standalone heuristic search procedure is required to find the best sub-architecture. On the contrary, the differentiable search methods slack the search space and update $\alpha$ and $\omega$ simultaneously, assuming that $\alpha$ gradually approaches the one-hot distribution corresponding to the best sub-architecture. So, after the super-network is well-trained, the best sub-architecture is often determined using a greedy algorithm on $\alpha$ while $\omega$ is often directly discarded.

### 3.2 The Devil Lies in the Optimization Gap

Both types of weight-sharing search algorithms suffer the instability issue, in particular, the search results are sensitive to random initialization and/or hyper-parameters. We owe this problem to the optimization gap, i.e., a well-optimized super-network does not always produce high-quality sub-architectures. Mathematically, it implies that optimizing Equation (3) yields an $\alpha^*$ and the corresponding $S^* = \text{disc}(\alpha^*)$, but there exists another architecture $S'$ that achieves far higher accuracy. Below, we show examples of how the optimization gap appears as different phenomena according to the type of weight-sharing NAS algorithms.

For weight-sharing heuristic search, the optimization gap can be quantified by sampling a set of sub-architectures, $\{S_1, S_2, \ldots, S_M\}$, and observing the true accuracy (by training each of them from scratch) and estimated accuracy (predicted by the super-network). Unfortunately, the relationship is often quite weak, e.g., the Kendall’s $\tau$-coefficient between the rank lists according to the true and estimated accuracy is close to zero $[45, 392]$ when the candidates have similar complexity. This is partly due to the strategy that optimizes part of the super-network in each iteration $[28, 60, 116]$, so that whether a sub-architecture performs well in the final evaluation depends on the frequency that it appears, in particular, the final iterations of the training procedure. This is mostly random and uncontrollable.

However, for weight-sharing differentiable search, the optimization gap can be quantified by observing the accuracy of the optimal sub-architecture without re-training the network weights, $\omega$. It is shown in Reference $[306]$ that even when the super-network achieves around 90% accuracy
Fig. 2. The optimization gap between the goals of weight-sharing optimization and individual optimization, where the latter is accurate while the former is approximated. We summarize four major reasons to compose of the optimization gap, sorted by their appearance in the NAS flowchart and corresponding to the contents in Sections 3.3.1–3.3.4, respectively. Note that weight-sharing heuristic methods may not involve the mathematical error, because $\nabla_\alpha \mathcal{L}(\omega^*(\alpha), \alpha; D_{\text{val}})$ is often not computed.

(in the CIFAR10 training set), the pruned sub-architecture, without parameter re-training, often reports less than 20% accuracy in the same dataset, which is dramatically low. This implies that the super-network is trained without “knowing” that a discretization stage will take effect to produce the final sub-architecture.

Summarizing the above, the optimization gap likely adds a non-observable noise, $\epsilon(\alpha)$, to the true accuracy of $\text{Eval}(\alpha) \leq 1 - \mathcal{L}(\omega^*(\alpha), \alpha; D_{\text{val}})$. Sometimes, for two candidates, say $\alpha_1$ and $\alpha_2$, the noise impacts even heavier, i.e., $|\epsilon(\alpha_1) - \epsilon(\alpha_2)| > |\text{Eval}(\alpha_1) - \text{Eval}(\alpha_2)|$. Due to the randomness of $\epsilon(\alpha)$, the judgment becomes unreliable. This can largely downgrade the search results, or cause the results sensitive to initialization or hyper-parameters.

3.3 Shrinking the Optimization Gap
Essentially, the optimization gap is the consequence of improving search speed with the price of lower evaluation accuracy. Shrinking the optimization gap forms the core challenge of weight-sharing NAS, for which researchers have proposed various solutions. Before going into details, we point out that the optimization gap is inevitable, because the network weights are shared by exponentially many sub-architectures and the optimization procedure focuses on minimizing the amortized loss over the entire search space, not a specific sub-architecture.

We first summarize the typical reasons that cause the optimization gap into four categories, shown in Figure 2. First, when the differentiable methods are used, the end-to-end training procedure may suffer dramatic errors in estimating the gradients with respect to $\alpha$. Second, the search procedure may cause the super-network fall into a local optimum, in particular, the joint optimization of $\alpha$ and $\omega$ increases the complexity of the high-dimensional space. Third, the discretization process that derives the optimal sub-architecture may incur a dramatic drop of the network training accuracy. Fourth, there are some differences in the network shape or hyper-parameters between the search and evaluation phases, making the searched architecture difficult in deployment.\(^1\) Note that most weight-sharing NAS approaches may involve more than one of the above issues. In what follows, we review the solutions to shrink the optimization gap by considering

\(^1\)The main reason is that in the search stage, the so-called super-network architecture needs to cover exponentially many architectures (e.g., each layer is formulated as the weighted sum of several different modules) and it is larger than the searched architecture (each layer only contains takes the determined module). Therefore, during the search stage, one needs to either reduce the network width or batch size, causing the optimization gap.
these factors individually or applying a learning-based approach that deals with the entire system as a black-box.

3.3.1 Fixing the Mathematical Error. The optimization error mostly exists in the differentiable search method, in particular, DARTS [195], in which a bi-level optimization method is used for updating $\omega$ and $\alpha$ separately. The reason for bi-level optimization lies in the imbalance between $\alpha$ and $\omega$. Most often, the number of learnable parameters of $\alpha$ is typically hundreds or thousands, but that of $\omega$ is often millions. So, minimizing Equation (3) can easily bias toward optimizing $\omega$ due to the higher dimension of the parameter space. To avoid such bias, DARTS [195] proposed to evaluate the network accuracy with respect to $\alpha$ and $\omega$ separately (i.e., $D_{\text{train}} \cup D_{\text{val}} = \emptyset$), which leads to the iterative optimization:

$$\omega_{t+1} \leftarrow \omega_t - \eta_\omega \cdot \nabla_\omega \mathcal{L}(\omega_t, \alpha_t; D_{\text{train}}),$$

$$\alpha_{t+1} \leftarrow \alpha_t - \eta_\alpha \cdot \nabla_\alpha \mathcal{L}(\omega_{t+1}, \alpha_t; D_{\text{val}}),$$

where $t$ is the iteration index starting with 0, and $\eta_\alpha$ and $\eta_\omega$ are learning rates. Bi-level optimization strategy avoids the bias, but the correctness of bi-level optimization depends on (i) $\omega_{t+1}$ has arrived at the optimum, i.e., $\omega_{t+1} = \omega^*(\alpha_t)$, which is very difficult to satisfy especially when the dimensionality of $\omega$ is high; (ii) the gradient with respect to $\alpha$ is accurately computed, but this is computationally intractable due to the requirement of computing $H_\omega^{-1}$, the inverse Hessian matrix of $\omega$. The solution that DARTS [195] used is to ignore the optimality of $\omega_{t+1}$ and use the identity matrix to approximate $H_\omega^{-1}$, but, as pointed out in References [20, 377], this leads to dramatic errors in gradient computation and forms the major reason for mode collapse, e.g., the searched sub-architecture are mostly occupied by the skip-connect operator and perform quite bad [20, 186, 377, 407].

A straightforward solution to alleviate the search instability is to perform early termination [186, 195, 323, 377] or further constrain the property of the sub-architecture, e.g., containing a fixed number of skip-connect operators [44, 186, 323]. But, not allowing the search procedure to converge brings the problem that the search result is sensitive to initialization, which reflects in the unsatisfying stability, e.g., most differentiable search methods can run into bad sub-architectures with a probability of 20%.–80%.

There are some work that tried to stabilize bi-level optimization from the mathematical perspective. Typical efforts include using a better approximation ($H_\omega$ to replace $H_\omega^{-1}$) to achieve an estimation error bound [20], applying adversarial perturbation as regularization to smooth the search space [43], or applying mixed-level optimization to bypass the dramatic approximation [122]. While these methods do not fix the error completely, an alternative path of research focuses on replacing bi-level optimization with one-level optimization, which is free of the mathematical burden, but the optimization bias toward $\omega$ needs to be addressed. For this purpose, researchers proposed to partition the operator set into parameterized and non-parameterized groups to facilitates the fair comparison between similar operators [174]. A more elegant solution is provided by Reference [21], which shows that simply adding regularization (e.g., data augmentation or Dropout) or using a larger dataset (e.g., ImageNet [69]) in the search procedure is sufficient to balance $\alpha$ and $\omega$ and improve one-level optimization. Nevertheless, since the

---

2A direct consequence of such bias is that $\alpha$ is not well optimized, which introduces random noise to NAS. In an extreme situation, the super-network can achieve satisfying accuracy by only optimizing $\omega$ (i.e., $\alpha$ remains the randomly initialized status), but this delivers zero information to NAS.

3It was proved [20] that, even in optimizing a toy super-network, the angle between the true and estimated gradient vectors of $\alpha$ can be larger than 90°, i.e., the optimization is going toward an uncontrolled, wrong direction.
validation set no longer exists, one-level optimization may suffers the difficulty of judging the extent that the super-network fits the training set, which may incur a higher risk of over-fitting.

3.3.2 **Avoiding the Sub-optima of Super-network.** Training a super-network is not easy, because the space determined by $\alpha$ is complicated and its property changes significantly with $\omega$. This can be understood as the competition among different operators, and thus whether an operator is competitive largely depends on how its weights (part of $\omega$) has been optimized. That being said, the operators that are easy to optimize (e.g., non-parameterized ones like skip-connect and pooling and the ones with fewer learnable weights like convolution with small kernels) may dominate in the early stage, which makes it even more difficult for the powerful operators (e.g., convolution of large kernels) to catch up with them [44]. In other words, the search algorithm runs into a local minimum in the space of $\alpha$.

One of the effective solutions is to warm up the search procedure [44, 229, 358, 359], i.e., freezing the update of $\alpha$ in the starting stage and allowing $\omega$ (mostly the weights of all convolution operators) to be well initialized. In essence, this is to postpone the optimization of $\alpha$. In the heuristic search methods, it behaves as performing fair sampling [60, 116] for super-network training followed by a standalone sampling procedure starting with a uniform distribution on $S$. Besides, there are also other kinds of regularization including partitioning the operators into parameterized and non-parameterized groups to avoid unfair competition [152, 174], using operator-level Dropout to weaken the superiority of non-parameterized operators [44], and dynamically adjusting the candidate operator set to allow some perturbation in $S$ [229].

Note that bi-level optimization also belongs to the larger category of regularization, which decomposes the joint optimization into the subspaces of $\alpha$ and $\omega$. This explains why regularization plays an important role in one-level optimization [21]. Finally, note that regularization often incurs slower convergence speed, but fortunately, the search cost of weight-sharing methods is relatively low and the increased overhead is often acceptable.

3.3.3 **Reducing the Discretization Loss.** After the super-network has been well optimized, discretization plays an important role in deriving the optimal sub-architecture from the super-network, i.e., $S^* = \text{disc}(\alpha^*)$ as in Equation (3). A straightforward solution is to directly eliminate the weak operators from the super-network, but it has been shown that discretization can dramatically downgrades the training accuracy of the super-network (without re-training), especially for the differentiable search methods [62, 306]. There are mainly three solutions. First, converting $\alpha$ from continuous to discrete so that the search procedure is always optimizing the discretized sub-architectures. This strategy is used in both heuristic [60, 116] and differentiable [31, 32, 351] search methods, and it can be interpreted as a kind of super-network regularization. Second, pushing the values of $\alpha$ toward either ends so that the eliminated elements have very low weights [62, 237, 306]. Third, weakening one-time discretization as gradual network pruning [21, 173, 237, 399] so that the super-network is not pulled too far from the normal status ($\omega$ is close to the optimum) and recovers after some training epochs (before the next pruning operation).

Note that, when the computational overhead is not considered, the safest way to reduce the discretization loss is to re-train all sub-architectures individually. Obviously, this strategy makes the algorithm degenerate to the heuristic search method with the distribution $\mathcal{P}$ provided by the weight-sharing search procedure and the advantage in efficiency no longer exists. To make better use of the search procedure, a common practice is to fine-tune the top-ranked sub-architecture for a short period of time [24, 333]. The length of fine-tuning is an adjustable parameter and plays the role of a tradeoff between purely weight-sharing methods (when the length is zero) and purely individual methods (when the length goes to infinity).
3.3.4 Bridging the Deployment Gap. The last step is to deploy the sub-architecture to the target scenario. In weight-sharing NAS, re-training is often required to achieve better performance, but there may be significant differences between the search and re-training settings. The major reasons are: (i) the search stage may need additional memory to save all network components and thus the network shape is shrunk; and (ii) the re-training stage may need to fit different constraints (e.g., the mobile setting \cite{131} that requires the network FLOPs is just below 600M). A practical way is to adjust the network shape (e.g., the depth, width, and input resolution) and change the hyper-parameters (e.g., the number of epochs, the learning rate, whether some regularization methods have been applied, etc.) accordingly. This causes a deployment gap. Conceptually, the algorithm finds the optimal sub-architecture under the setting used in the search procedure, but this does not imply its optimality under the hyper-parameters used in the re-training stage.\footnote{For example, if the search stage contains much fewer cells, then the searched cell architecture may have very deep connections, which increases the difficulty of convergence in the re-training stage. Similarly, if the search stage is equipped with lighter regularization (e.g., Dropout), then the searched architecture may contain fewer learnable parameters, which downgrades its upper-bound in the re-training stage.} Note that even with the same network shape, the difference in hyper-parameters can lead to unstable search results, while this factor is often concealed by other deployment gaps \cite{20}.

The first revealed deployment gap is named the depth gap \cite{44}, i.e., in the search space that contains repeatable cells, the searched sub-architecture can be extended in depth to achieve higher accuracy. However, it was shown that the optimal sub-architecture varies under different depths, and it proposed a progressive method to close the depth gap. Despite the progressively growing super-network, a more essential solution to the depth gap is to directly search in the target network shape. This is natural for most heuristic search methods \cite{60, 116, 130, 301} while the differentiable search methods can encounter the difficulty of training very deep super-networks—this is mainly due to the mathematical error \cite{20} elaborated in Section 3.3.1. Going one step further, researchers have been investigating the relationship between the shallow super-network and the deep sub-architectures \cite{140, 222, 369}, and a likely conclusion is that they can benefit from each other.

Another important deployment gap comes from transferring the searched architecture to other tasks, e.g., from image classification to object detection or semantic segmentation. Though many NAS methods have shown the ability of direct architecture transfer \cite{60, 358, 415}, we argue that this is not the optimal solution, since the optimal architecture may be task-specific. An example is that EfficientDet \cite{304} is difficult to be trained yet does not show dominating detection accuracy, although it borrows the architecture and pre-trained weights from EfficientNet \cite{302}, the state-of-the-art image classification architecture. To improve the transferability of the searched architectures, researchers have applied meta-learning \cite{42, 261, 274} (see Section 4.3) or performed network adaptation \cite{89}, but we point out that many aspects remain unexplored in this new direction.

3.3.5 Learning-based Approaches. To handle the complicated and coupled factors of the optimization gap, researchers have also investigated learning-based approaches that judge the quality of each sub-architecture based on both the super-network and other kinds of auxiliary information. We point out that this is closely related to the heuristic search methods that keep the distribution $\Psi$ of the search space uniform. The difference lies in that the heuristic search methods often update a controller to latently assign a probability to each sub-architecture, but the learning-based methods often encode each sub-architecture into a vector and then use a model to predict the accuracy explicitly.
The performance of learning-based approaches largely depends on the way of representing the sub-architectures, as the encoding method determines the relationship between different sub-architectures [96, 236, 332]. Sometimes, the auxiliary information does not come from a standalone predictor, but from a super-network, e.g., via knowledge distillation [25, 310]. Unlike formulating the entire space for the toy-sized NAS benchmarks [80, 368], in a large search space, the predictor is often embedded into the search procedure to guide the super-network optimization [45, 140, 333]. Besides improving accuracy, the predictor can also filter out weak candidates earlier and thus accelerate the search procedure [401, 411].

We emphasize that the learning-based approaches tend to consider NAS as a black-box predictor and use a small number of training samples to approximate the accuracy distribution on the search space. This is somewhat dangerous as the actual property of the distribution remains unclear. From a higher perspective, these approaches alleviate the optimization gap globally, but do not guarantee to capture the local property very well (i.e., the gap still exists).

3.4 Unsolved Problems

Despite the aforementioned efforts in shrinking the optimization gap, we notice that several important problems of weight-sharing NAS remain unsolved.

First, weight-sharing NAS methods are built upon the assumption that the search space is relatively smooth. However, the actual property of the search space remains unclear. For example, does the search space contain exponentially many sub-architectures that are equally good? Given one of the best sub-architecture, $S^*$, is it possible that the neighborhood of $S^*$ is mostly filled with bad sub-architectures? These issues become increasingly important as the search space gets more complicated. But, due to the huge computational costs to accurately evaluate all sub-architectures (except for the toy search spaces [80, 368, 379]), this problem remains mostly uncovered.

Second, weight-sharing NAS methods facilitate different operators and connections to compete with each other, but it is unknown if the comparison criterion is made fair. For example, almost all search spaces involve the competition among convolutions with various kernel sizes (e.g., $3 \times 3$ vs. $5 \times 5$ or $7 \times 7$), and sometimes, network compression forces competition among different channel and/or bit widths. Although the ‘heavy’ operators are intuitively stronger than the ‘light’ operators, they often suffer slower convergence during the training procedure. That is to say, the search result is sensitive to the time point that comparison is made between these operators. This can cause considerable uncertainty especially in very large search spaces.

Third, weight-sharing NAS methods are able to prune a large super-network into small sub-architectures, but may encounter difficulties in the reverse direction, i.e., deriving a large and/or deep sub-architecture from a relatively small super-network. This is necessary when higher recognition accuracy is desired, and existing approaches are mostly rule-based, including copying the repeatable cells (see Section 3.3.4) and applying simple magnification rules upon the base network. We expect weight-sharing NAS methods to have the ability of extending the search space when needed, which may require the assistance of dynamic routing networks (see Section 4.2).

4 RELATED RESEARCH DIRECTIONS

We briefly review some related research directions to NAS, including (i) compressing the network toward a higher cost-performance ratio, (ii) designing dynamic routing network architectures to

---

5Non-weight-sharing methods can achieve this goal [256, 257], but the price is to re-train the enlarged network every time, bringing heavy computational overheads.
fit different input data, (iii) learning efficient learning strategies (meta-learning), and (iv) looking for the optimal hyper-parameters for network training.

4.1 Model Compression

As the deep networks become more and more complicated, deploying them to edge devices (e.g., mobile phones) requires a new direction named model compression [118, 143]. Two of the most popular techniques are network pruning [119], which eliminates less useful channels or operators from a trained network, and network quantization [41, 105], which uses low-precision (e.g., 1-bit [253]) computation to replace the frequently used FP32 (i.e., 32-bit floating point number) operations. Both methodologies have been verified to achieve a better tradeoff between accuracy and complexity.

Conceptually, model compression shares a similar formulation to NAS, i.e., the generalized formulation in Section 2.1 directly applies with either a regularization term for model complexity or a hard constraint for the maximal resource. Therefore, NAS approaches are often easily transferred for model compression, including pruning [345], quantization [342], and joint optimization [47]. Sometimes, the searched configuration or connectivity mask can improve recognition accuracy [2]. Yet, the optimization gap also exists in these scenarios, and the methods introduced in Section 3.3 are still useful, e.g., super-network regularization [345], network fine-tuning [26], re-training [106], and so on. In addition, the unfairness issue is significant when the operators of different channel and/or bit widths exist [29]. Nevertheless, it is believed that NAS will form an important module in the future direction named software-hardware co-design.

4.2 Dynamic Routing Networks

Most neural architectures are static, i.e., not changing with input data. As a counterpart to save computation, researchers have investigated the option of adjusting the network architecture according to test data [196]. This strategy, named dynamic routing, brings various benefits, including higher amortized efficiency by assigning heavy computation to easy samples and light computation to hard samples [136] information for higher recognition accuracy [185]. There are also discussion on the training strategy for the dynamic routing networks [175, 219].

From a generalized perspective, a dynamic network can be considered as a variant of a static network. Note that even for a static network, the input data can cause part of neurons (after the activation function, e.g., ReLU [228]) to output a value of zero, that is, the connections to these neurons are eliminated. Therefore, the difference lies in that dynamic networks are switching off larger units (e.g., network layers or blocks [344]). This implies that NAS methods, with simple modification, can be applied for dynamic network design [376]. More importantly, dynamic routing networks may pave the way of weight-sharing NAS in an enlargeable search space [64], which we believe is a new research direction.

4.3 Meta-learning

Meta-learning [269, 270] is sometimes referred to as "learning to learn," which aims to formulate the common practices in a learning procedure, so that the learned model can quickly adapt to new tasks and/or environments. It originates from the idea of mimicking the learning behavior of human beings [167], yet eases the machine learning algorithms to deploy to various scenarios, e.g., the few-shot learning setting [315] that allows the model to learn new concepts with a small number of training samples. There are mainly three types of meta-learners, namely, model-based,
metric-based, and optimization-based methods, differing from each other in the way of formulating the discriminative function. For reviews of meta-learning, please refer to References [172, 312].

Meta-learning has a longer history than NAS, and modern NAS was considered a special case of meta-learning [9, 23, 85], namely, learning to design a computational model that assists learning. In particular, meta-learning plays an important role in the early age of NAS, when the NAS methods need to learn from a small number of sub-architectures that are sampled and individually evaluated. As weight-sharing NAS methods become more and more popular, meta-learning can be used for various purposes, including (i) enhancing the few-shot learning ability of NAS methods [81], (ii) allowing NAS methods to transfer to different tasks [261], and (iii) providing complementary information to shrink the optimization gap [52, 83].

4.4 Hyper-parameter Optimization

Hyper-parameter optimization (HPO) is another important direction of AutoML, with the aim of finding proper hyper-parameters to improve the quality of model training. This is a classic problem in machine learning [63], but in the deep learning era, the optimization goal has been focused on the important settings that impact network training including the learning rate [5, 204], mini-batch size [227], weight decay [205], momentum [298], or a combination of them [221]. For more thorough reviews of HPO, please refer to References [19, 93, 374].

We point out that NAS is essentially a kind of HPO, i.e., the network architecture is encoded as a set of hyper-parameters. Indeed, the NAS approaches in the early years have mostly inherited the ideology of HPO methods including random search [19], heuristic search [18, 281], and meta-learning [94]. However, NAS approaches have been transplanted to HPO [15], and sometimes NAS and HPO are performed together [162, 378]. But, one of the important challenges that HPO algorithms have encountered is the much slower optimization speed compared to the weight-sharing NAS algorithms. This is because each step of HPO requires a complete training procedure, unlike NAS that updates the architectural parameters, \( \alpha \), in each step of the search procedure. Essentially, this is due to the missing of the greedy optimization assumption: during NAS training, it is always a good choice to minimize the loss function in a short period of time, but this does not hold for HPO training. A direct consequence is the limited degree of freedom of the HPO search space. We expect the future research of HPO can alleviate this difficulty and thus achieve higher flexibility of optimizing hyper-parameters. The recent efforts to combine NAS and HPO [76] provided a promising direction.

4.5 Automated Data Augmentation

An important subarea of HPO is automated data augmentation. Data augmentation is a technique that enlarges the training set using transformed copies of the existing training images. The assumption is that slightly perturbed (e.g., rotated, flipped, distorted, etc.) samples share the same label as the original one. AutoAugment [65] first formulated data augmentation into an HPO problem in a large search space and applied reinforcement learning for finding the optimal strategy. It achieved good practice in many image classification datasets, but the search cost is very high. There are a few follow-ups that tried to accelerate AutoAugment using approximated methods [66, 127, 188]. Without careful control, the aggressively augmented training samples may contaminate the

6Borrowed from http://metalearning-symposium.ml/files/vinyals.pdf.
7A clear example lies in the learning rate. If the goal is to cause the loss function drop rapidly, then the algorithm may choose to use a small learning rate at the start of training, but this strategy is not good for the global optimality.
training procedure, for which researchers have proposed various solutions [327, 346, 390]. There is a survey [278] that reviews a wide range of data augmentation techniques for deep learning.

5 CONCLUSIONS AND FUTURE PROSPECTS

Neural architecture search (NAS) is an important topic of automated machine learning (AutoML) that aims to find replacement for the manually designed architectures. With three-year rapid development, NAS shows the potential of becoming a standard tool of deep learning algorithms applied to different AI problems. Currently, NAS has contributed significantly to some state-of-the-art computer vision algorithms, but NAS also suffers significant drawbacks that hinders its broader applications.

However, in the past months, the research of NAS seems to have slowed down, arguably due to both the limited search space and unstable search algorithms. These factors are highly correlated. The conservative design of the search spaces makes it difficult to evaluate and compare the performance of different search algorithms; however, the instability of the search algorithms obstacles the exploration in larger search spaces. Today, two most popular search spaces, the DARTS search space (Section 2.2.4) and the MobileNet search space (Section 2.2.5) have been so thoroughly explored that some tricks (e.g., preserving a fixed number of skip-connect operators in each cell [44, 186]) can produce highly competitive search results, or some inaccurate conclusions can be made [239, 349], yet these results do not show significant advantages over the random search methods with a reasonable amount of trials.

As the last but not least part of this survey, we share our opinions on the future directions of NAS. We believe that the future prospect of NAS lies in whether researchers can propose a new framework to systematically solve the aforementioned problems, in particular, designing more flexible search spaces and developing reliable search algorithms. These two factors facilitate each other, but in the current time point (July 31, 2020), the former seems more urgent.

A good search space for NAS research is expected to satisfy four conditions. First, the space is sufficiently large to include both chain-styled [279] and multi-path [299] networks, popular hand-designed modules (e.g., the residual block [124], group convolution [348], depthwise separable convolution [56], shuffle modules [214], attention units [134, 382], activation functions [252, 323], etc.), and the combination of them [131]. According to our rough estimation, the cardinality of such a search space is expected to exceed $10^{1000}$, but the largest search space to date (GOLD-NAS [21]) only has a cardinality of $10^{110} – 10^{170}$. Second, the space should be adjustable, allowing the search algorithm to increase or decrease the model size without restarting the search procedure from scratch. Third, the space can be applied to a wide range of vision problems without heavy modifications—preferably, the searched architecture transfers to different tasks with merely light-weighted heads to be replaced (or searched), like HRNet [321]. Fourth, the best architecture in the space should achieve significantly better accuracy-complexity tradeoff than that found by random search (with reasonable computational costs) or manual design (including pure manual design and adding simple rules to the search procedure).

Based on the search space, a good search algorithm should, first of all, be able to explore the search space without getting lost, e.g., running into a set of sub-optimal architectures that are even worse than random search or manual design. We believe that various network training methods will be useful, including knowledge distillation [108, 126, 200, 289, 310], automated data augmentation (see Section 4.5), dynamic routing networks (see Section 4.2), and transfer learning [207, 240]. In addition, the algorithm should allow different constraints (e.g., hardware factors like latency [212, 341, 359] and other factors [92, 286]) to be embedded. We believe that such NAS algorithms, once developed, can largely facilitate the development of deep learning and even the next generation of AI computing.
ACKNOWLEDGMENTS

The authors thank the colleagues in Huawei Noah's Ark Lab, in particular, Prof. Tong Zhang, Dr. Wei Zhang, Dr. Chunjing Xu, Jianzhong He, and Zewei Du, for instructive discussions. We would also like to thank the organizers of AutoML.org for providing an awesome list of AutoML papers: https://www.automl.org/automl/literature-on-neural-architecture-search/.

REFERENCES

[1] George Adam and Jonathan Lorraine. 2019. Understanding neural architecture search techniques. Retrieved from https://arXiv:1904.00438.
[2] Karim Ahmed and Lorenzo Torresani. 2018. Maskconnect: Connectivity learning by gradient descent. In Proceedings of the European Conference on Computer Vision.
[3] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. 2019. Adaptive stochastic natural gradient method for one-shot neural architecture search. Retrieved from https://arXiv:1905.08537.
[4] Jie An, Haoyi Xiong, Jun Huan, and Jiebo Luo. 2020. Ultrafast photorealistic style transfer via neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.
[5] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent. In Advances in Neural Information Processing Systems. MIT Press.
[6] Randy Ardywibowo, Shahin Boluki, Xinyu Gong, Zhangyang Wang, and Xiaoning Qian. 2020. NADS: Neural architecture distribution search for uncertainty awareness. Retrieved from https://arXiv:2006.06646.
[7] Woong Bae, Seungho Lee, Yeha Lee, Beomhee Park, Minku Chung, and Kyu-Hwan Jung. 2019. Resource optimized neural architecture search for 3D medical image segmentation. In Medical Image Computing and Computer-Assisted Intervention. Springer.
[8] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing neural network architectures using reinforcement learning. In Proceedings of the International Conference on Learning Representations.
[9] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Accelerating neural architecture search using performance prediction. Retrieved from https://arXiv:1705.10823.
[10] Prasanna Balapratasha, Romain Egele, Misha Salim, Stefan Wild, Venkatram Vishwanath, Fangfang Xia, Tom Brettin, and Rick Stevens. 2019. Scalable reinforcement-learning-based neural architecture search for cancer deep learning research. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.
[11] Maria Baldeon-Calisto and Susana K. Lai-Yuen. 2020. AdaResU-Net: Multiobjective adaptive convolutional neural network for medical image segmentation. Neurocomputing 392 (2020), 325–340.
[12] Ahmed Baruwa, Mojeeb Abisiga, Ibrahim Gbadebesin, and Afeez Fakunle. 2019. Leveraging end-to-end speech recognition with neural architecture search. Retrieved from https://arXiv:1912.05946.
[13] Pouya Bashivan, Mark Tensen, and James J. DiCarlo. 2019. Teacher guided architecture search. In Proceedings of the International Conference on Computer Vision.
[14] Justin Bayer, Daan Wierstra, Julian Togelius, and Jürgen Schmidhuber. 2009. Evolving memory cell structures for sequence learning. In Proceedings of the International Conference on Artificial Neural Networks.
[15] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. 2017. Neural optimizer search with reinforcement learning. In Proceedings of the International Conference on Machine Learning.
[16] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In Proceedings of the International Conference on Machine Learning.
[17] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. 2020. Can weight sharing outperform random architecture search? An investigation with TuNAS. In Proceedings of the Conference on Computer Vision and Pattern Recognition.
[18] James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the International Conference on Machine Learning.
[19] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyperparameter optimization. In Advances in Neural Information Processing Systems. MIT Press.
[20] KaiFeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. 2019. Stabilizing DARTS with amended gradient estimation on architectural parameters. Retrieved from https://arXiv:1910.11831.
[21] Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. 2020. GOLD-NAS: Gradual, one-level, differentiable. Retrieved from https://arXiv:2007.03331.

[22] Simone Bianco, Marco Buzzelli, Gianluigi Ciocca, and Raimondo Schettini. 2020. Neural architecture search for image saliency fusion. Info. Fusion 57 (2020), 89–101.

[23] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. 2017. SMASH: One-shot model architecture search through hypernetworks. Retrieved from https://arXiv:1708.05344.

[24] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient architecture search by network transformation. In Proceedings of the AAAI Conference on Artificial Intelligence.

[25] Han Cai, Chuang Gan, and Song Han. 2020. Once for all: Train one network and specialize it for efficient deployment. In Proceedings of the International Conference on Learning Representations.

[26] Han Cai, Tianzhe Wang, Zhanghao Wu, Kuan Wang, Ji Lin, and Song Han. 2019. On-device image classification with proxyless neural architecture search and quantization-aware fine-tuning. In Proceedings of the International Conference on Computer Vision Workshops.

[27] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. 2018. Path-level network transformation for efficient architecture search. In Proceedings of the International Conference on Machine Learning.

[28] Han Cai, Ligeng Zhu, and Song Han. 2019. Proxylessnas: Direct neural architecture search on target task and hardware. In Proceedings of the International Conference on Learning Representations.

[29] Zhaowei Cai and Nuno Vasconcelos. 2020. Rethinking differentiable search for mixed-precision neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[30] Fabio Maria Carlucci, Pedro Esperanca, Rasul Tutunov, Marco Singh, Victor Gabillon, Antoine Yang, Hang Xu, Zewei Chen, and Jun Wang. 2019. MANAS: Multi-agent neural architecture search. Retrieved from https://arXiv:1909.01051.

[31] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. 2019. Probabilistic neural architecture search. Retrieved from https://arXiv:1902.05116.

[32] Jianlong Chang, Yiwen Guo, Gaofeng Meng, Shiming Xiang, Chunhong Pan et al. 2019. DATA: Differentiable architecture approximation. In Advances in Neural Information Processing Systems. MIT Press.

[33] Bo Chen, Golnaz Ghiasi, Hanxiao Liu, Tsung-Yi Lin, Dmitry Kalenichenko, Hartwig Adam, and Quoc V. Le. 2020. Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[34] Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. 2020. Adabert: Task-adaptive bert compression with differentiable neural architecture search. Retrieved from https://arXiv:2001.04246.

[35] Hanlin Chen, Baocang Zhang Li’an Zhuo, Xiaowu Zheng, Jianzhang Liu, David Doermann, and Rongrong Ji. 2020. Binarized neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.

[36] Junkun Chen, Kaiyu Chen, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Exploring shared structures and hierarchies for multiple nlp tasks. Retrieved from https://arXiv:1808.07658.

[37] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. 2018. Searching for efficient multi-scale architectures for dense image prediction. In Advances in Neural Information Processing Systems. MIT Press.

[38] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. 40, 4 (2017), 834–848.

[39] Shoufa Chen, Yunpeng Chen, Shuicheng Yan, and Jiashi Feng. 2019. Efficient differentiable neural architecture search with meta kernels. Retrieved from https://arXiv:1912.04749.

[40] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. In Proceedings of the International Conference on Learning Representations.

[41] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. In Proceedings of the International Conference on Machine Learning.

[42] Xin Chen, Yawen Duan, Zewei Chen, Hang Xu, Zihao Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. 2020. CATCH: Context-based meta reinforcement learning for transferrable architecture search. In Proceedings of the European Conference on Computer Vision.

[43] Xiangning Chen and Cho-Jui Hsieh. 2020. Stabilizing differentiable architecture search via perturbation-based regularization. Retrieved from https://arXiv:2002.05283.

[44] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In Proceedings of the International Conference on Computer Vision.

[45] Xin Chen, Lingxi Xie, Jun Wu, Longhui Wei, Yuhui Xu, and Qi Tian. 2020. Fitting the search space of weight-sharing NAS with graph convolutional networks. Retrieved from https://arXiv:2004.08423.
[46] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. 2019. Renas: Reinforced evolutionary neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[47] Yukang Chen, Gaofeng Meng, Qian Zhang, Xinbang Zhang, Liangchen Song, Shiming Xiang, and Chunhong Pan. 2018. Joint neural architecture search and quantization. Retrieved from https://arXiv:1811.09426.

[48] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. 2019. DetNAS: Backbone search for object detection. In Advances in Neural Information Processing Systems. MIT Press.

[49] Yushi Chen, Kaiqiang Zhu, Lin Zhu, Xin He, Pedram Ghamisi, and Jón Atli Benediktsson. 2019. Automatic design of convolutional neural network for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. 57, 9 (2019), 7048–7066.

[50] An-Chieh Cheng, Jin-Dong Dong, Chi-Hung Hsu, Shu-Huan Chang, Min Sun, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. 2018. Searching toward pareto-optimal device-aware neural architectures. In Proceedings of the International Conference on Computer-Aided Design.

[51] An-Chieh Cheng, Chieh Hubert Lin, Da-Cheng Juan, Wei Wei, and Min Sun. 2020. InstaNAS: Instance-aware neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.

[52] Hsin-Pai Cheng, Tunhou Zhang, Yukun Yang, Feng Yan, Shiyu Li, Harris Teague, Hai Li, and Yiran Chen. 2019. SwiftNet: Using graph propagation as meta-knowledge to search highly representative neural architectures. Retrieved from https://arXiv:1906.08305.

[53] Hsin-Pai Cheng, Tunhou Zhang, Yukun Yang, Feng Yan, Harris Teague, Yiran Chen, and Hai Li. 2019. Msnet: Structural wired neural architecture search for internet of things. In Proceedings of the International Conference on Computer Vision Workshops.

[54] Weiwu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Differentiable neural input search for recommender systems. Retrieved from https://arXiv:2006.04466.

[55] Minsu Cho, Mohammadreza Soltani, and Chinmay Hegde. 2019. One-shot neural architecture search via compressive sensing. Retrieved from https://arXiv:1906.02869.

[56] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[57] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. 2019. Scarlettnas: Bridging the gap between scalability and fairness in neural architecture search. Retrieved from https://arXiv:1908.06022.

[58] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. 2019. Fast, accurate and lightweight super-resolution with neural architecture search. Retrieved from https://arXiv:1901.07261.

[59] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2020. MoGA: Searching beyond MobileNetV3. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing.

[60] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2020. MoGA: Searching beyond MobileNetV3. In Proceedings of the International Conference on Computer Vision.

[61] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. 2019. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. Retrieved from https://arXiv:1907.01845.

[62] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. 2019. Multi-objective reinforced evolution in mobile neural architecture search. Retrieved from https://arXiv:1901.01074.

[63] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. 2020. Fair darts: Eliminating unfair advantages in differentiable architecture search. In Proceedings of the European Conference on Computer Vision.

[64] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. 2017. Adanet: Adaptive structural learning of artificial neural networks. In Proceedings of the International Conference on Machine Learning.

[65] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. Autoaugment: Learning augmentation policies from data. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[66] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the Computer Vision and Pattern Recognition Workshops.

[67] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yuning Hu, Yiming Wu, Yangqing Jia et al. 2019. Chamnet: Towards efficient network design through platform-aware model adaptation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[68] Boyang Deng, Junjie Yan, and Dahua Lin. 2017. Peephole: Predicting network performance before training. Retrieved from https://arXiv:1712.03351.

[69] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[70] Georgi Dikov, Patrick van der Smagt, and Justin Bayer. 2019. Bayesian learning of neural network architectures. Retrieved from https://arXiv:1901.04436.
[71] Shaojin Ding, Tianlong Chen, Xinyu Gong, Weiwei Zha, and Zhangyang Wang. 2020. AutoSpeech: Neural architecture search for speaker recognition. Retrieved from https://arXiv:2005.03215.
[72] Shifei Ding, Hui Li, Chunyang Su, Junzhuo Yu, and Fengxiang Jin. 2013. Evolutionary artificial neural networks: A review. Artific. Intell. Rev. 39, 3 (2013), 251–260.
[73] Hongwei Dong, Bin Zou, Lamei Zhang, and Siyu Zhang. 2020. Automatic design of CNNs via differentiable neural architecture search for PolSAR image classification. IEEE Trans. Geosci. Remote Sens. 58, 9 (2020), 6362–6375.
[74] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. 2018. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In Proceedings of the European Conference on Computer Vision.
[75] Nanqing Dong, Min Xu, Xiaodan Liang, Yiliang Jiang, Wei Dai, and Eric Xing. 2019. Neural architecture search for adversarial medical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 828–836.
[76] Xuanyi Dong, Mingxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc V. Le. 2020. AutoHAS: Differentiable hyper-parameter and architecture search. Retrieved from https://arXiv:2006.03656.
[77] Xuanyi Dong and Yi Yang. 2019. Network pruning via transformable architecture search. In Advances in Neural Information Processing Systems. MIT Press.
[78] Xuanyi Dong and Yi Yang. 2019. One-shot neural architecture search via self-evaluated template network. In Proceedings of the International Conference on Computer Vision.
[79] Xuanyi Dong and Yi Yang. 2020. Searching for a robust neural architecture in four GPU hours. In Proceedings of the Conference on Computer and Communication Security. IEEE Press.
[80] Xuanyi Dong and Yi Yang. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. In Proceedings of the International Conference on Learning Representations.
[81] Sivan Dovish, Eli Schwartz, Chao Xue, Rogerio Feris, Alex Bronstein, Raja Giryes, and Leonid Karlinsky. 2019. MetAdapt: Meta-learned task-adaptive architecture for few-shot classification. Retrieved from https://arXiv:1912.00412.
[82] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. 2020. SpineNet: Learning scale-permutated backbone for recognition and localization. In Proceedings of the Conference on Computer Vision and Pattern Recognition.
[83] Alina Dubatovka, Efi Kokiooupolu, Luciano Sbaiz, Andrea Gesmundo, Gabor Bartok, and Jesse Berent. 2019. Ranking architectures using meta-learning. Retrieved from https://arXiv:1911.11481.
[84] Jayanta K. Dutta, Jiayi Liu, Unmesh Kurup, and Mohak Shah. 2018. Effective building block design for deep convolutional neural networks using search. Retrieved from https://arXiv:1801.08577.
[85] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2017. Simple and efficient architecture search for convolutional neural networks. Retrieved from https://arXiv:1711.04528.
[86] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Efficient multi-objective neural architecture search via lamarckian evolution. In Proceedings of the International Conference on Learning Representations.
[87] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. J. Mach. Learn. Res. 20, 55 (2019), 1–21.
[88] Jiemin Fang, Yukang Chen, Xinbang Zhang, Qian Zhang, Chang Huang, Gaofeng Meng, Wenyu Liu, and Xinggang Wang. 2019. EAT-NAS: Elastic architecture transfer for accelerating large-scale neural architecture search. Retrieved from https://arXiv:1901.05884.
[89] Jiemin Fang, Yuzhu Sun, Kangjian Peng, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. 2020. Fast neural network adaptation via parameter remapping and architecture search. In Proceedings of the International Conference on Learning Representations.
[90] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. 2020. Densely connected search space for more flexible neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.
[91] Igor Fedorov, Ryan P. Adams, Matthew Mattina, and Paul Whatmough. 2019. Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. In Advances in Neural Information Processing Systems. MIT Press.
[92] Javier Fernandez-Marques, Paul N. Whatmough, Andrew Mundy, and Matthew Mattina. 2020. Searching for winograd-aware quantized networks. Retrieved from https://arXiv:2002.10711.
[93] Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In Automated Machine Learning. Springer, Cham, 3–33.
[94] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing bayesian hyperparameter optimization via meta-learning. In Proceedings of the AAAI Conference on Artificial Intelligence.
[95] Ben Fielding and Li Zhang. 2018. Evolving image classification architectures with enhanced particle swarm optimisation. IEEE Access 6 (2018), 68560–68575.
[96] David Friede, Jovita Lukasik, Heiner Stuckenschmidt, and Margret Keuper. 2019. A variational-sequential graph autoencoder for neural architecture performance prediction. Retrieved from https://arXiv:1912.05317.

[97] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. 2020. AutoGAN-distiller: Searching to compress generative adversarial networks. Retrieved from https://arXiv:2006.08198.

[98] Chen Gao, Yumeng Chen, Si Liu, Zhenxiong Tan, and Shunicheng Yan. 2020. Adversarialnas: Adversarial neural architecture search for gans. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[99] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. 2020. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[100] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. Retrieved from https://arXiv:1904.09981.

[101] Yonatan Geifman and Ran El-Yaniv. 2019. Deep active learning with a neural architecture search. In Advances in Neural Information Processing Systems. MIT Press.

[102] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[103] Ross Girshick. 2015. Fast r-cnn. In Proceedings of the International Conference on Computer Vision.

[104] Yunchao Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. 2019. Autogan: Neural architecture search for generative adversarial networks. In Proceedings of the International Conference on Computer Vision.

[105] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. Retrieved from https://arXiv:1412.6115.

[106] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[107] Sam Green, Craig M. Vineyard, Ryan Helinski, and Cetin Kaya Koç. 2019. RAPDARTS: Resource-aware progressive differentiable architecture search. Retrieved from https://arXiv:1911.05704.

[108] Jindong Gu and Volker Tresp. 2020. Search for better students to learn distilled knowledge. Retrieved from https://arXiv:2001.11612.

[109] Dazhou Guo, Dakai Jin, Zhuotun Zhu, Tsung-Ying Ho, Adam P. Harrison, Chun-Hung Chao, Jing Xiao, and Le Lu. 2020. Organ at risk segmentation for head and neck cancer using stratified learning and neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[110] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. 2020. Hit-detector: Hierarchical trinity architecture search for object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[111] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. 2020. When NAS meets robustness: In search of robust architectures against adversarial attacks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[112] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. 2019. Irlas: Inverse reinforcement learning for architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[113] Ronghao Guo, Chen Lin, Chuming Li, Keyu Tian, Ming Sun, Lu Sheng, and Junjie Yan. 2020. Powering one-shot topological NAS with stabilized share-parameter proxy. In Proceedings of the European Conference on Computer Vision.

[114] Yong Guo, Yongsheung Luo, Zhenhao He, Jin Huang, and Jian Chen. 2020. Hierarchical neural architecture search for single image super-resolution. Retrieved from https://arXiv:2003.04619.

[115] Long Guo, Yin Zheng, Mingkui Tan, Qi Chen, Jian Chen, Peilin Zhao, and Junzhou Huang. 2019. Nat: Neural architecture transformer for accurate and compact architectures. In Advances in Neural Information Processing Systems. MIT Press.

[116] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2019. Single path one-shot neural architecture search with uniform sampling. Retrieved from https://arXiv:1904.00420.

[117] Song Han, Han Cai, Ligeng Zhu, Ji Lin, Kuan Wang, Zhijian Liu, and Yujun Lin. 2019. Design automation for efficient deep learning computing. Retrieved from https://arXiv:1904.10616.

[118] Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In Proceedings of the International Conference on Learning Representations.

[119] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems. MIT Press.

[120] Shayan Hassantabar, Xiaoliang Dai, and Niraj K. Jha. 2019. STEERAGE: Synthesis of neural networks using architecture search and grow-and-prune methods. Retrieved from https://arXiv:1912.05831.

[121] Chaoyang He, Murali Annavaram, and Salman Avestatehr. 2020. FedNAS: Federated deep learning via neural architecture search. Retrieved from https://arXiv:2004.08546.
[122] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. 2020. Milenas: Efficient neural architecture search via mixed-level reformulation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[123] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In Proceedings of the International Conference on Computer Vision.

[124] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[125] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2019. AutoML: A survey of the state-of-the-art. Retrieved from https://arXiv:1908.00709.

[126] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Retrieved from https://arXiv:1503.02531.

[127] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In Proceedings of the International Conference on Machine Learning.

[128] Kary Ho, Andrew Gilbert, Hailin Jin, and John Collomosse. 2020. Neural architecture search for deep image prior. Retrieved from https://arXiv:2001.04776.

[129] Hyeong Gwon Hong, Pyunghwan Ahn, and Junmo Kim. 2019. EDAS: Efficient and differentiable architecture search. Retrieved from https://arXiv:1912.01237.

[130] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan et al. 2019. Searching for mobilenetv3. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[131] Andrew G. Howard, Menglong Zhou, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. Retrieved from https://arXiv:1704.04861.

[132] Chi-Hung Hsu, Shu-Huan Chang, Jhao-Hong Liang, Hsin-Ping Chou, Chun-Hao Liu, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juang. 2018. Monas: Multi-objective neural architecture search using reinforcement learning. Retrieved from https://arXiv:1806.10332.

[133] Hanzhang Hu, John Langford, Rich Caruana, Saurajit Mukherjee, Eric J. Horvitz, and Debadeepta Dey. 2019. Efficient forward architecture search. In Advances in Neural Information Processing Systems.

[134] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[135] Yutao Hu, Xiaolong Jiang, Xuhui Liu, Baoshang Zhang, Jungong Han, Xianbin Cao, and David Doermann. 2020. NAS-Count: Efficient convolutional neural networks for mobile applications. Retrieved from https://arXiv:2003.00217.

[136] Shenyang Huang, Vincent François-Lavet, and Guillaume Rabusseau. 2019. Neural architecture search for class-incremental learning. Retrieved from https://arXiv:1909.06686.

[137] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. 2018. Gnas: A greedy neural architecture search method for multi-attribute learning. In ACM Proceedings of the International Conference on Multimedia.

[138] Shenyang Huang, Vincent François-Lavet, and Guillaume Rabusseau. 2019. Neural architecture search for class-incremental learning. Retrieved from https://arXiv:1909.06686.

[139] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. 2018. Gnas: A greedy neural architecture search method for multi-attribute learning. In ACM Proceedings of the International Conference on Multimedia.

[140] Sian-Yao Huang and Wei-Ta Chu. 2020. PONAS: Progressive one-shot neural architecture search for very efficient deployment. Retrieved from https://arXiv:2003.05112.

[141] Zhisheng Huang and Bing Xiang. 2019. Wenet: Weighted networks for recurrent network architecture search. Retrieved from https://arXiv:1904.03819.

[142] Andrew Hundt, Varun Jain, and Gregory D. Hager. 2019. sharpdarts: Faster and more accurate differentiable architecture search. Retrieved from https://arXiv:1903.09900.

[143] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. Retrieved from https://arXiv:1602.07360.

[144] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning.

[145] William Irwin-Harris, Yanan Sun, Bing Xue, and Mengjie Zhang. 2019. A graph-based encoding for evolutionary convolutional neural network architecture design. In Proceedings of the IEEE Congress on Evolutionary Computation.

[146] Roxana Istrate, Florian Scheidegger, Giovanni Mariani, Dimitrios Nikolopoulos, Costas Bekas, and A. Cristiano I. Malossi. 2019. Tapas: Train-less accuracy predictor for architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.
Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap

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

Chenhan Jiang, Shaoju Wang, Xiaodan Liang, Hang Xu, and Nong Xiao. 2020. ElixirNet: Relation-aware network architecture adaptation for medical lesion detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 11093–11100.

Chenhan Jiang, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. 2020. SP-NAS: Serial-to-parallel backbone search for object detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Weiwen Jiang, Xinyi Zhang, Edwin H.-M. Sha, Lei Yang, Qingfeng Zhuge, Yiyu Shi, and Jingtong Hu. 2019. Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search. In *Proceedings of the Annual Design Automation Conference*.

Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In *Proceedings of the Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*.

Yang Jiang, Cong Zhao, Zeyang Dou, and Lei Pang. 2019. Neural architecture refinement: A practical way for avoiding overfitting in NAS. Retrieved from https://arXiv:1905.02341.

Haifeng Jin, Qingquan Song, and Xia Hu. 2018. Efficient neural architecture search with network morphism. Retrieved from https://arXiv:1806.10282.

Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *SIGKDD Proceedings of the International Conference on Knowledge Discovery & Data Mining*.

Xiaojie Jin, Jiang Wang, Joshua Slocum, Ming-Hsuan Yang, Shengyang Dai, Shuiyong Yan, and Jiashi Feng. 2019. Rc-darts: Resource constrained differentiable architecture search. Retrieved from https://arXiv:1912.12814.

Francisco Erivaldo Fernandes Junior and Gary G. Yen. 2019. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evolution. Comput.* 49 (2019), 62–74.

Purushotham Kamath, Abhishek Singh, and Debo Dutta. 2018. Neural architecture construction using envelopenets. Retrieved from https://arXiv:1803.06744.

Kirthivasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. 2018. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*. MIT Press.

Kirthivasan Kandasamy, Karun Raju Vysyaraaju, Willie Neiswanger, Biswajit Paria, Christopher R. Collins, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. 2020. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *J. Mach. Learn. Res.* 21, 81 (2020), 1–27.

Sapir Kaplan and Raja Giryes. 2020. Self-supervised neural architecture search. Retrieved from https://arXiv:2007.01500.

Sungwoong Kim, Ildoo Kim, Sunghin Lim, Woohyuk Baek, Chiheon Kim, Hyungjoo Cho, Boogeon Yoon, and Taesup Kim. 2019. Scalable neural architecture search for 3D medical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Aaron Klein and Frank Hutter. 2019. Tabular benchmarks for joint architecture and hyperparameter optimization. Retrieved from https://arXiv:1905.04970.

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

Efikokipoulou, Anja Hauth, Luciano Sbaiz, Andrea Gesmundo, Gabor Bartok, and Jesse Berent. 2019. Fast task-aware architecture inference. Retrieved from https://arXiv:1902.05781.

Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. MIT Press.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behav. Brain Sci.* 40 (2017).

Kevin A. Laube and Andreas Zell. 2019. Shufflenasnets: Efficient cnn models through modified efficient neural architecture search. In *Proceedings of the International Joint Conference on Neural Networks*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

Heung-Chang Lee, Do-Guk Kim, and Bohyung Han. 2020. Efficient decoupled neural architecture search by structure and operation sampling. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*.

Royson Lee, Lukasz Dudziak, Mohamed Abdelfattah, Stylianos I. Venieris, Hyeji Kim, Hongkai Wen, and Nicholas D. Lane. 2020. Journey towards tiny perceptual super-resolution. In *Proceedings of the European Conference on Computer Vision*. 

ACM Computing Surveys, Vol. 54, No. 9, Article 183. Publication date: October 2021.
[172] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. 2015. Metalearning: A survey of trends and technologies. *Artif. Intell. Rev.* 44, 1 (2015), 117–130.

[173] Guohao Li, Guoqchong Qian, Itzel C. Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2020. Sgas: Sequential greedy architecture search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[174] Guilin Li, Xing Zhang, Zitong Wang, Zhenguo Li, and Tong Zhang. 2019. StacNAS: Towards stable and consistent optimization for differentiable neural architecture search. Retrieved from https://arXiv:1909.11926.

[175] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. 2019. Improved techniques for training adaptive deep networks. In *Proceedings of the International Conference on Computer Vision*.

[176] Jixiang Li, Chuming Liang, Bo Zhang, Zhao Wang, Fei Xiang, and Xiangxiang Chu. 2019. Neural architecture search on acoustic scene classification. Retrieved from https://arXiv:1912.12825.

[177] Liam Li and Ameet Talwalkar. 2019. Random search and reproducibility for neural architecture search. Retrieved from https://arXiv:1902.07638.

[178] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. 2020. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[179] Ruoteng Li, Robby T. Tan, and Loong-Fah Cheong. 2020. All in one bad weather removal using architectural search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[180] Ting Li, Junbo Zhang, Kainan Bao, Yuxuan Liang, Yexin Li, and Yu Zheng. 2020. Autost: Efficient neural architecture search for spatio-temporal prediction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[181] Xin Li, Yiming Zhou, Zheng Pan, and Jiashì Feng. 2019. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[182] Yinqiao Li, Chi Hu, Yuhao Zhang, Nuo Xu, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and Changliang Li. 2020. Learning architectures from an extended search space for language modeling. Retrieved from https://arXiv:2005.02593.

[183] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lin, Linjie Yang, Chixing Yu, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L. Yuille. 2020. Neural architecture search for lightweight non-local networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[184] Yaoman Li and Irwin King. 2019. Architecture search for image inpainting. In *Proceedings of the International Symposium on Neural Networks*. MIT Press.

[185] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. 2020. Learning dynamic routing for semantic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[186] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. 2019. Darts+: Improved differentiable architecture search with early stopping. Retrieved from https://arXiv:1909.06035.

[187] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. 2018. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

[188] Sungbin Lim, Ildoo Kim, Taesup Kim, Chieheon Kim, and Sungwoong Kim. 2019. Fast autoaugment. In *Advances in Neural Information Processing Systems*. MIT Press.

[189] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. 2020. Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[190] Marius Lindauer and Frank Hutter. 2019. Best practices for scientific research on neural architecture search. Retrieved from https://arXiv:1909.02453.

[191] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

[192] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. 2020. Are labels necessary for neural architecture search? In *Proceedings of the European Conference on Computer Vision*.

[193] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision*.

[194] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2018. Hierarchical representations for efficient architecture search. In *Proceedings of the International Conference on Learning Representations*.

[195] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. Darts: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations*.

[196] Luanli Liu and Jia Deng. 2018. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[197] Peng Liu, Mohammad D. El Bashra, Yangjunyi Li, Yao Xiao, Pina C. Sanelli, and Ruogu Fang. 2019. Deep evolutionary networks with expedited genetic algorithms for medical image denoising. *Med. Image Anal.* 54 (2019), 306–315.
Advances in Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap

[198] Peiye Liu, Bo Wu, Huadong Ma, Pavan Kumar Chundi, and Mingoo Seok. 2019. MemNet: Memory-efficiency guided neural architecture search with augment-trim learning. Retrieved from https://arXiv:1907.09569.

[199] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision.

[200] Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. 2020. Search to distill: Pearls are everywhere but not the eyes. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[201] Yifan Liu, Bohan Zhuang, Chunhua Shen, Hao Chen, and Wei Yin. 2019. Training compact neural networks via auxiliary overparameterization. Retrieved from https://arXiv:1909.02214.

[202] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[203] Pablo Ribaltas Lorenzo and Jakub Nalepa. 2018. Memetic evolution of deep neural networks. In Proceedings of the Genetic and Evolutionary Computation Conference.

[204] Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In Proceedings of the International Conference on Learning Representations.

[205] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In Proceedings of the Conference on Learning Representations.

[206] David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60, 2 (2004), 91–110.

[207] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. 2020. Neural architecture transfer. Retrieved from https://arXiv:2005.05859.

[208] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference.

[209] Renqian Luo, Tao Qin, and Enhong Chen. 2019. Understanding and improving one-shot neural architecture optimization. Retrieved from https://arXiv:1909.10815.

[210] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Semi-Supervised neural architecture search. Retrieved from https://arXiv:2002.10389.

[211] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural architecture optimization. In Advances in Neural Information Processing Systems. MIT Press.

[212] Li Linya Zhang, Yuqing Yang, Yuhang Jiang, Wenwu Zhu, and Yunxin Liu. 2020. Fast Hardware-aware neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops.

[213] Li Linya, Zhang, Yuqing, Yang, Yuan, Zhu, and Yunxin Liu. 2020. Fast Hardware-aware neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops.

[214] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision.

[215] Vladimir Macko, Charles Weill, Hanna Mazzawi, and Javier Gonzalez. 2019. Improving neural architecture search image classifiers via ensemble learning. Retrieved from https://arXiv:1903.06256.

[216] Marcus Märtens and Dario Izzo. 2019. Neural network architecture search with differentiable cartesian genetic programming for regression. In Proceedings of the Genetic and Evolutionary Computation Conference.

[217] Krysztof Maziarz, Mingxing Tan, Andrey Khorlin, Kuang-Yu Samuel Chang, and Andrea Gesmundo. 2018. Evolutionary-neural hybrid agents for architecture search. Retrieved from https://arXiv:1811.09828.

[218] Hanna Mazzawi, Xavi Gonzalvo, Aleks Kracun, Prashant Sridhar, Niranjan Subrahmanya, Ignacio Lopez-Moreno, Hyun-Jin Park, and Patrick Violette. 2019. Improving keyword spotting and language identification via neural architecture search at scale. In Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH’19).

[219] Mason McGill and Pietro Perona. 2017. Deciding how to decide: Dynamic routing in artificial neural networks. Retrieved from https://arXiv:1703.06217.

[220] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. 2020. Atomnas: Fine-grained end-to-end neural architecture search. In Proceedings of the International Conference on Learning Representations.

[221] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2016. Towards automatically tuned neural networks. In Proceedings of the Workshop on Automatic Machine Learning.

[222] Rang Meng, Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. 2020. Neural inheritance relation guided one-shot layer assignment search. Retrieved from https://arXiv:2002.12580.

[223] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. Retrieved from https://arXiv:1609.07843.
[244] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy et al. 2019. Evolving deep neural networks. In Proceedings of the Conference on Artificial Intelligence in the Age of Neural Networks and Brain Computing. 293–312.

[225] Aliashgar Mortazi and Ulas Bagci. 2018. Automatically designing CNN architectures for medical image segmentation. In Proceedings of the International Workshop on Machine Learning in Medical Imaging.

[226] Marcin Mozejko, Tomasz Latkowski, Łukasz Treszczotko, Michał Szafrańik, and Krzysztof Trojanowski. 2020. Superkernel neural architecture search for image denoising. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops.

[227] Norman Mu, Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. 2018. Parameter re-initialization through cyclical batch size schedules. Retrieved from https://arXiv:1812.01216.

[228] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the International Conference on Machine Learning.

[229] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. 2019. Xnas: Neural architecture search with expert advice. In Advances in Neural Information Processing Systems. MIT Press.

[230] Renato Negri and Geoff Gordon. 2017. Deeparchitect: Automatically designing and training deep architectures. Retrieved from https://arXiv:1704.08792.

[231] Renato Negri, Matthew Gormley, Geoffrey J. Gordon, Darshan Patil, Nghia Le, and Daniel Ferreira. 2019. Towards modular and programmable architecture search. In Advances in Neural Information Processing Systems. MIT Press.

[232] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. 2019. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[233] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. 2020. Architecture search of dynamic cells for semantic video segmentation. In Proceedings of the Winter Conference on Applications of Computer Vision.

[234] Vladimir Nekrasov, Chunhua Shen, and Ian Reid. 2020. Template-based automatic search of compact semantic segmentation architectures. In Proceedings of the Winter Conference on Applications of Computer Vision.

[235] Kien Nguyen, Clinton Fookes, and Sridha Sridharan. 2020. Constrained design of deep iris networks. IEEE Trans. Image Process. 29 (2020), 7166–7175.

[236] Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. 2020. A generic graph-based neural architecture encoding scheme for predictor-based NAS. In Proceedings of the European Conference on Computer Vision.

[237] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. 2020. Asap: Architecture search, anneal and prune. In Proceedings of the International Conference on Artificial Intelligence and Statistics.

[238] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating recurrent neural network memory structures using neuro-evolution. In Proceedings of the Genetic and Evolutionary Computation Conference.

[239] T. Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter A. N. Bosman. 2020. Local search is a remarkably strong baseline for neural architecture search. Retrieved from https://arXiv:2004.08996.

[240] Rameswar Panda, Michele Merler, Mayoori Jaiswal, Hui Wu, Kandan Ramakrishnan, Ulrich Finkler, Chun-Fu Chen, Minsk Cho, David Kung, Rogerio Feres et al. 2020. NASTransfer: Analyzing architecture transferability in large scale neural architecture search. Retrieved from https://arXiv:2006.13314.

[241] Ramakanth Pasunuru and Mohit Bansal. 2019. Continual and multi-task architecture search. In Proceedings of the Annual Meeting of the Association for Computational Linguistics.

[242] Wei Peng, Xiaopeng Hong, and Guoying Zhao. 2019. Video action recognition via neural architecture searching. In Proceedings of the International Conference on Image Processing.

[243] Juan-Manuel Pérez-Rúa, Moez Baccouche, and Stephane Pateux. 2018. Efficient progressive neural architecture search. Retrieved from https://arXiv:1808.00391.

[244] Juan-Manuel Pérez-Rúa, Valentin Viezeuf, Stéphane Pateux, Moez Baccouche, and Frédéric Jurie. 2019. Mfas: Multimodal fusion architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[245] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. In Proceedings of the International Conference on Machine Learning.

[246] A. J. Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S. Ryoo. 2019. Evolving space-time neural architectures for videos. In Proceedings of the International Conference on Computer Vision.

[247] Jonas Prellberg and Oliver Kramer. 2018. Lamarckian evolution of convolutional neural networks. In Proceedings of the International Conference on Parallel Problem Solving from Nature.

[248] Zhaofan Qiu, Ting Yao, Yiheng Zhang, Yongdong Zhang, and Tao Mei. 2019. Scheduled differentiable architecture search for visual recognition. Retrieved from https://arXiv:1909.10236.

[249] Ruijie Quan, Xuanyi Dong, Yu Wu, Linchao Zhu, and Yi Yang. 2019. Auto-reid: Searching for a part-aware convnet for person re-identification. In Proceedings of the International Conference on Computer Vision.
[250] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollar. 2019. On network design spaces for visual recognition. In Proceedings of the International Conference on Computer Vision.

[251] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. 2020. Designing network design spaces. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[252] Fajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. Retrieved from https://arXiv:1710.05941.

[253] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In Proceedings of the European Conference on Computer Vision.

[254] Aditya Rawal and Risto Miikkulainen. 2018. From nodes to networks: Evolving recurrent neural networks. Retrieved from https://arXiv:1803.04439.

[255] 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.

[256] Esteban Real, Chen Liang, David R. So, and Quoc V. Le. 2020. AutoML-Zero: Evolving machine learning algorithms from scratch. Retrieved from https://arXiv:2003.03384.

[257] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In Proceedings of the International Conference on Machine Learning.

[258] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[259] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2020. A comprehensive survey of neural architecture search: Challenges and solutions. Retrieved from https://arXiv:2006.02903.

[260] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems. MIT Press.

[261] J. Gomez Robles and Joaquin Vanschoren. 2019. Learning to reinforcement learn for neural architecture search. Retrieved from https://arXiv:1911.03769.

[262] Raanan Y. Rohdekar, Shami Nisimov, Yaniv Gurwicz, Guy Koren, and Gal Novik. 2018. Constructing deep neural networks by bayesian network structure learning. In Advances in Neural Information Processing Systems. MIT Press.

[263] 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. Int. J. Comput. Vision 115, 3 (2015), 211–252.

[264] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. 2016. Advodispnet: Improving disparity estimation with automl. In Proceedings of the AAAI Conference on Artificial Intelligence.

[265] Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. 2020. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In Proceedings of the International Conference on Learning Representations.

[266] Tom Schaul and Jürgen Schmidhuber. 2010. Metalearning. Scholarpedia 5, 6 (2010), 4650.

[267] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2019. Evaluating the search phase of neural architecture search. Retrieved from https://arXiv:1902.09701.

[268] Cristina Saltori, Subhankar Roy, Nicu Sebe, and Giovanni Iacca. 2019. Regularized evolutionary algorithm for dynamic neural topology search. In Proceedings of the International Conference on Image Analysis and Processing.

[269] Pedro Savarese and Michael Maire. 2019. Learning implicitly recurrent CNNs through parameter sharing. Retrieved from https://arXiv:1902.09701.

[270] Tom Schaul and Jürgen Schmidhuber. 2010. Metalearning. Scholarpedia 5, 6 (2010), 4650.

[271] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2019. Evaluating the search phase of neural architecture search. Retrieved from https://arXiv:1902.08142.

[272] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. 2019. Squeezeenas: Fast neural architecture search for faster semantic segmentation. In Proceedings of the International Conference on Computer Vision workshops.

[273] Albert Shaw, Wei Wei, Weiyang Liu, Le Song, and Bo Dai. 2019. Meta architecture search. In Advances in Neural Information Processing Systems. MIT Press.

[274] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. 2019. Searching for accurate binary neural architectures. In Proceedings of the International Conference on Computer Vision Workshops.

[275] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T. Kwok, and Tong Zhang. 2019. Multi-objective neural architecture search via predictive network performance optimization. Retrieved from https://arXiv:1911.09336.
[277] Richard Shin, Charles Packer, and Dawn Song. 2018. Differentiable neural network architecture search. In Proceedings of the International Conference on Learning Representations: Workshops.

[278] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. J. Big Data 6, 1 (2019), 60.

[279] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations.

[280] Sean C. Smithson, Guang Yang, Warren J. Gross, and Brett H. Meyer. 2016. Neural networks designing neural networks: Multi-objective hyper-parameter optimization. In Proceedings of the International Conference on Computer-Aided Design.

[281] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems. MIT Press, 2951–2959.

[282] David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In Proceedings of the International Conference on Machine Learning.

[283] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. 2020. Efficient residual dense block search for image super-resolution. In Proceedings of the AAAI Conference on Artificial Intelligence.

[284] Qingquan Song, Dehua Cheng, Hanning Zhou, Jiyan Yang, Yuandong Tian, and Xia Hu. 2020. Towards automated neural interaction discovery for click-through rate prediction. Retrieved from https://arXiv:2007.06434.

[285] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (2014), 1929–1958.

[286] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymeropoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. 2019. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases.

[287] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. Nature Mach. Intell. 1, 1 (2019), 24–35.

[288] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. Evolution. Comput. 10, 2 (2002), 99–127.

[289] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2019. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. Retrieved from https://arXiv:1912.07768.

[290] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A genetic programming approach to designing convolutional neural network architectures. In Proceedings of the Genetic and Evolutionary Computation Conference.

[291] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[292] Li Sun, Xiaoyi Yu, Liuan Wang, Jun Sun, Hiroya Inakoshi, Ken Kobayashi, and Hiromichi Kobashi. 2019. Automatic neural network search method for open set recognition. In Proceedings of the International Conference on Image Processing.

[293] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G. Yan, and Mengjie Zhang. 2019. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. IEEE Trans. Evolution. Comput. 24, 2 (2019), 350–364.

[294] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yan. 2018. Automatically evolving CNN architectures based on blocks. Retrieved from https://arXiv:1810.11875.

[295] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yan. 2018. A particle swarm optimization-based flexible convolutional autoencoder for image classification. IEEE Trans. Neural Netw. Learn. Syst. 30, 8 (2018), 2295–2309.

[296] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yan. 2019. Evolving deep convolutional neural networks for image classification. IEEE Trans. Evolution. Comput. 23, 1 (2019), 89–103.

[297] Yanan Sun, Gary G. Yan, and Zhang Yi. 2018. Evolving unsupervised deep neural networks for learning meaningful representations. IEEE Trans. Evolution. Comput. 23, 1 (2018), 89–103.

[298] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In Proceedings of the International Conference on Machine Learning.

[299] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[300] El-Ghazali Talbi. 2020. Optimization of deep neural networks: A survey and unified taxonomy. Technical Report. University of Lille and INRIA.

[301] 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 Conference on Computer Vision and Pattern Recognition.
Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. 2018. Evolving deep convolutional neural networks by variable-
Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra et al. 2016. Matching networks for one-shot learn-
Tom Véniat, Olivier Schwander, and Ludovic Denoyer. 2019. Stochastic adaptive neural architecture search for key-
Danilo Vasconcellos Vargas and Shashank Kotyan. 2019. Evolving robust neural architectures to defend from adver-
Gerard Jacques van Wyk and Anna Sergeevna Bosman. 2019. Evolutionary neural architecture search for image
Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. 2020. Multi-fidelity
Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human pose estimation via deep neural networks. In
Proceedings of the Conference on Computer Vision and Pattern Recognition.
Yunjie Tian, Chang Liu, Lingxi Xie, Jianbin Jiao, and Qixiang Ye. 2020. Discretization-aware architecture search. Retrieved from https://arXiv:2007.03154.
Yuesong Tian, Li Shen, Guinan Su, Zhifeng Li, and Wei Liu. 2020. AlphaGAN: Fully differentiable architecture search for generative adversarial networks. Retrieved from https://arXiv:2006.09134.
Yuan Tian, Qin Wang, Zhiwu Huang, Wen Li, Dengxin Dai, Minghao Yang, Jun Wang, and Olga Fink. 2020. Off-policy reinforcement learning for efficient and effective GAN architecture search. Retrieved from https://arXiv:2007.09180.
Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human pose estimation via deep neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.
Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. 2020. Multi-fidelity neural architecture search with knowledge distillation. Retrieved from https://arXiv:2006.08341.
Gerard Jacques van Wyk and Anna Sergeevna Bosman. 2019. Evolutionary neural architecture search for image restoration. In Proceedings of the International Joint Conference on Neural Networks.
Joaquin Vanschoren. 2018. Meta-learning: A survey. Retrieved from https://arXiv:1810.03548.
Danilo Vasconcellos Vargas and Shashank Kotyan. 2019. Evolving robust neural architectures to defend from adversarial attacks. Retrieved from https://arXiv:1906.11667.
Tom Véniat, Olivier Schwander, and Ludovic Denoyer. 2019. Stochastic adaptive neural architecture search for keyword spotting. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing.
Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra et al. 2016. Matching networks for one shot learning. In Advances in Neural Information Processing Systems. MIT Press.
Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. 2018. Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In Proceedings of the IEEE Congress on Evolutionary Computation.
Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. 2018. A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. In Proceedings of the Australasian Joint Conference on Artificial Intelligence.
Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. 2019. A hybrid GA-PSO method for evolving architecture and short connections of deep convolutional neural networks. In Proceedings of the Pacific Rim International Conference on Artificial Intelligence.
Bin Wang, Bing Xue, and Mengjie Zhang. 2019. Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks. Retrieved from https://arXiv:1907.12659.
Hanchao Wang and Jun Huan. 2019. Agan: Towards automated design of generative adversarial networks. Retrieved from https://arXiv:1906.11080.
Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang et al. 2020. Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (2020).
Jingdong Wang, Zhen Wei, Ting Zhang, and Wenjun Zeng. 2016. Deeply fused nets. Retrieved from https://arXiv:1605.07716.
Lanfei Wang, Lingxi Xie, Tianyi Zhang, Jun Guo, and Qi Tian. 2019. Scalable NAS with factorizable architectural parameters. Retrieved from https://arXiv:1912.13256.
Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. 2019. Sample-efficient neural architecture search by learning action space. Retrieved from https://arXiv:1906.06832.
Linnan Wang, Yiyang Zhao, Yuu Jimai, Yuandong Tian, and Rodrigo Fonseca. 2019. Alphax: Exploring neural architectures with deep neural networks and monte carlo tree search. Retrieved from https://arXiv:1903.11059.
Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyou Kou, Yunhai Tong, Mao Yang, and Lidong Zhou. 2020. TextNAS: A neural architecture search space tailored for text representation. In Proceedings of the AAAI Conference on Artificial Intelligence.
Longhui Wei, An Xiao, Lingxi Xie, Xin Chen, Xiaopeng Zhang, and Qi Tian. 2020. Circumventing outliers of autoaugment with knowledge distillation. In Proceedings of the European Conference on Computer Vision.
[328] Tao Wei, Changhu Wang, and Chang Wen Chen. 2017. Modularized morphing of neural networks. Retrieved from https://arXiv:1701.03281.

[329] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. 2016. Network morphism. In Proceedings of the International Conference on Machine Learning.

[330] Yu Weng, Tianbao Zhou, Yuje Li, and Xiaoyu Qiu. 2019. Nas-unet: Neural architecture search for medical image segmentation. IEEE Access 7 (2019), 44247–44257.

[331] Yu Weng, Tianbao Zhou, Lei Liu, and Chunlei Xia. 2019. Automatic convolutional neural network search for image classification under different scenes. IEEE Access 7 (2019), 38495–38506.

[332] Colin White, Willie Neiswanger, Sam Nolen, and Yash Savani. 2020. A study on encodings for neural architecture search. Retrieved from https://arXiv:2007.04965.

[333] Colin White, Willie Neiswanger, and Yash Savani. 2019. Bananas: Bayesian optimization with neural architectures for neural architecture search. Retrieved from https://arXiv:1910.11858.

[334] Martin Wistuba. 2017. Finding competitive network architectures within a day using UCT. Retrieved from https://arXiv:1712.07420.

[335] Martin Wistuba. 2018. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases.

[336] Martin Wistuba and Tejaswini Pedapati. 2019. Inductive transfer for neural architecture optimization. Retrieved from https://arXiv:1903.03536.

[337] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. 2019. A survey on neural architecture search. Retrieved from https://arXiv:1905.01392.

[338] Alexander Wong, Zhong Qiu Lin, and Brendan Chwyl. 2019. Attonets: Compact and efficient deep neural networks for the edge via human-machine collaborative design. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops.

[339] Catherine Wong, Neil Housby, Yifeng Lu, and Andrea Gesmundo. 2018. Transfer learning with neural autml. In Advances in Neural Information Processing Systems. MIT Press.

[340] Ken C. L. Wong and Mehdi Moradi. 2019. SegNAS3D: Network architecture search with derivative-free global optimization for 3D image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention.

[341] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[342] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. 2018. Mixed precision quantization of convnets via differentiable neural architecture search. Retrieved from https://arXiv:1812.00990.

[343] Huihai Wu, Junge Zhang, and Kaili Huang. 2019. SparseMask: Differentiable connectivity learning for dense image prediction. In Proceedings of the International Conference on Computer Vision.

[344] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. 2018. Blockdrop: Dynamic inference paths in residual networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[345] Xia Xiao, Ziyang Wang, and Sanguthevar Rajasekaran. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In Advances in Neural Information Processing Systems. MIT Press.

[346] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. 2020. Adversarial examples improve image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[347] Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In Proceedings of the International Conference on Computer Vision.

[348] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[349] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019. Exploring randomly wired neural networks for image recognition. In Proceedings of the International Conference on Computer Vision.

[350] Saining Xie, Alexander Wong, Zhong Qiu Lin, and Brendan Chwyl. 2019. Attonets: Compact and efficient deep neural networks for the edge via human-machine collaborative design. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops.

[351] Tao Wei, Changhu Wang, and Chang Wen Chen. 2016. Network morphism. In Proceedings of the International Conference on Machine Learning.

[352] Yu Weng, Tianbao Zhou, Lei Liu, and Chunlei Xia. 2019. Automatic convolutional neural network search for image classification under different scenes. IEEE Access 7 (2019), 38495–38506.

[353] Colin White, Willie Neiswanger, Sam Nolen, and Yash Savani. 2020. A study on encodings for neural architecture search. Retrieved from https://arXiv:2007.04965.

[354] Colin White, Willie Neiswanger, and Yash Savani. 2019. Bananas: Bayesian optimization with neural architectures for neural architecture search. Retrieved from https://arXiv:1910.11858.

[355] Martin Wistuba and Tejaswini Pedapati. 2019. Inductive transfer for neural architecture optimization. Retrieved from https://arXiv:1903.03536.

[356] Martin Wistuba. 2018. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases.

[357] Martin Wistuba and Tejaswini Pedapati. 2019. Inductive transfer for neural architecture optimization. Retrieved from https://arXiv:1903.03536.

[358] Martin Wistuba. 2017. Finding competitive network architectures within a day using UCT. Retrieved from https://arXiv:1712.07420.
[354] Yunyang Xiong, Ronak Mehta, and Vikas Singh. 2019. Resource constrained neural network architecture search: Will a submodularity assumption help?. In Proceedings of the International Conference on Computer Vision.

[355] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. 2019. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In Proceedings of the International Conference on Computer Vision.

[356] Yixing Xu, Yunhe Wang, Kai Han, Hanting Chen, Yehui Tang, Shangling Jui, Chunjing Xu, Qi Tian, and Chang Xu. 2019. Renas: Architecture ranking for powerful networks. Retrieved from https://arXiv:1910.01523.

[357] Yixing Xu, Yunhe Wang, Kai Han, Shangling Jui, Chunjing Xu, Qi Tian, and Chang Xu. 2020. ReNAS: Relativistic evaluation of neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.

[358] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2020. Pd-darts: Partial channel connections for memory-efficient architecture search. In Proceedings of the International Conference on Learning Representations.

[359] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Bowen Shi, Qi Tian, and Hongkai Xiong. 2020. Latency-aware differentiable neural architecture search. Retrieved from https://arXiv:2001.06392.

[360] Shen Yan, Biyi Fang, Faen Zhang, Yu Zheng, Xiao Zeng, Mi Zhang, and Hui Xu. 2019. Hm-nas: Efficient neural architecture search via hierarchical masking. In Proceedings of the International Conference on Computer Vision Workshops.

[361] Xingang Yan, Weiwen Jiang, Yiyu Shi, and Cheng Zhuo. 2020. MS-NAS: Multi-scale neural architecture search for medical image segmentation. Retrieved from https://arXiv:2007.06151.

[362] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. 2020. NAS evaluation is frustratingly hard. In Proceedings of the International Conference on Learning Representations.

[363] Sen Yang, Wankou Yang, and Zhen Cui. 2019. Pose neural fabrics search. Retrieved from https://arXiv:1909.07068.

[364] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. 2020. Cars: Continuous evolution for efficient neural architecture search. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[365] Lewei Yao, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. 2020. SM-NAS: Structural-to-modular neural architecture search for object detection. In Proceedings of the AAAI Conference on Artificial Intelligence.

[366] Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. 2020. Efficient neural architecture search via proximal iterations. In Proceedings of the AAAI Conference on Artificial Intelligence.

[367] Xin Yao. 1999. Evolving artificial neural networks. Proc. IEEE 87, 9 (1999), 1423–1447.

[368] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. Nas-bench-101: Towards reproducible neural architecture search. In Proceedings of the International Conference on Machine Learning.

[369] Hongyuan Yu and Houwen Peng. 2020. Cyclic differentiable architecture search. Retrieved from https://arXiv:2006.10724.

[370] Jiahui Yu and Thomas Huang. 2019. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. Retrieved from https://arXiv:1903.11728.

[371] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. 2020. Bignas: Scaling up neural architecture search with big single-stage models. In Proceedings of the European Conference on Computer Vision.

[372] Kaicheng Yu, Christian Scuito, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2020. Evaluating the search phase of neural architecture search. In Proceedings of the International Conference on Learning Representations.

[373] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L. Yuille, and Daguang Xu. 2020. C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

[374] Tong Yu and Hong Zhu. 2020. Hyper-parameter optimization: A review of algorithms and applications. Retrieved from https://arXiv:2003.05689.

[375] Ye Yu, Yingmin Li, Shuai Che, Niraj K. Jha, and Weifeng Zhang. 2020. Software-defined design space exploration for an efficient DNN accelerator architecture. IEEE Trans. Comput. 70, 1 (2020), 45–56.

[376] Zhihang Yuan, Bingzhe Wu, Zheng Liang, Shiwan Zhao, Weichen Bi, and Guangyu Sun. 2019. S2DNAS: Transforming static CNN model for dynamic inference via neural architecture search. Retrieved from https://arXiv:1911.07033.

[377] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. 2020. Understanding and robustifying differentiable architecture search. In Proceedings of the International Conference on Learning Representations.

[378] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. 2018. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. Retrieved from https://arXiv:1807.06906.

[379] Arber Zela, Julien Siems, and Frank Hutter. 2020. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In Proceedings of the International Conference on Learning Representations.
Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2020. Practical block-wise neural network architecture generation. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2021. BlockQNN: Efficient block-wise neural network architecture generation. IEEE Trans. Pattern Anal. Mach. Intell. 43, 7 (2021), 2314–2328.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. 2019. BayesNAS: A bayesian approach for neural architecture search. In Proceedings of the International Conference on Machine Learning.

Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-gnn: Neural architecture search of graph neural networks. Retrieved from https://arXiv:1909.03184.
[406] Peng Zhou, Lingxi Xie, Xiaopeng Zhang, Bingbing Ni, and Qi Tian. 2020. Searching towards class-aware generators for conditional generative adversarial networks. Retrieved from https://arXiv:2006.14208.

[407] Pan Zhou, Caiming Xiong, Richard Socher, and Steven C. H. Hoi. 2020. Theory-inspired path-regularized differential network architecture search. Retrieved from https://arXiv:2006.16537.

[408] Xin Zhou, Dejing Dou, and Boyang Li. 2019. Searching for stage-wise neural graphs in the limit. Retrieved from https://arXiv:1912.12860.

[409] Yanqi Zhou, Siavash Ebrahimi, Sercan Ö. Arik, Haonan Yu, Hairong Liu, and Greg Diamos. 2018. Resource-efficient neural architect. Retrieved from https://arXiv:1806.07912.

[410] Yizhou Zhou, Xiaoyan Sun, Chong Luo, Zheng-Jun Zha, and Wenjun Zeng. 2020. Posterior-guided neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence.

[411] Yanqi Zhou, Peng Wang, Sercan Arik, Haomian Yu, Syed Zawad, Feng Yan, and Greg Diamos. 2019. EPNAS: Efficient progressive neural architecture search. In Proceedings of the British Machine Vision Conference.

[412] Hui Zhu, Zhulin An, Chuanguang Yang, Kaiqiang Xu, Erhu Zhao, and Yongjun Xu. 2019. EENA: Efficient evolution of neural architecture. In Proceedings of the International Conference on Computer Vision Workshops.

[413] Zhuotun Zhu, Chenxi Liu, Dong Yang, Alan Yuille, and Daguang Xu. 2019. V-nas: Neural architecture search for volumetric medical image segmentation. In Proceedings of the International Conference on 3D Vision.

[414] Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In Proceedings of the International Conference on Learning Representations.

[415] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition.

Received September 2020; revised April 2021; accepted June 2021