Curriculum-NAS: Curriculum Weight-Sharing Neural Architecture Search

Yuwei Zhou
zhou-yw21@mails.tsinghua.edu.cn
Tsinghua University

Xin Wang[*]
xin_wang@tsinghua.edu.cn
Tsinghua University

Hong Chen
h-chen20@mails.tsinghua.edu.cn
Tsinghua University

Xuguang Duan
dxg18@mails.tsinghua.edu.cn
Tsinghua University

Chaoyu Guan
zhou-yw21@mails.tsinghua.edu.cn
Tsinghua University

Wenwu Zhu[*]
wwzhu@tsinghua.edu.cn
Tsinghua University

ABSTRACT

Neural Architecture Search (NAS) is an effective way to automatically design neural architectures for various multimedia applications. Weight-sharing, as one of the most popular NAS strategies, has been widely adopted due to its search efficiency. Existing weight-sharing NAS methods overlook the influence of data distribution and treat each data sample equally. Contrastively, in this paper, we empirically discover that different data samples have different influences on architectures, e.g., some data samples are easy to fit by certain architectures but hard by others. Hence, there exist architectures with better performances on early data samples being more likely to be discovered in the whole NAS searching process, which leads to a suboptimal searching result. To tackle this problem, we propose Curriculum-NAS, a curriculum training framework on weight-sharing NAS, which dynamically changes the training data weights during the searching process. In particular, Curriculum-NAS utilizes the multiple subnets included in weight-sharing NAS to jointly assess data uncertainty, which serves as the difficulty criterion in a curriculum manner, so that the potentially optimal architectures can obtain higher probability of being fully trained and discovered. Extensive experiments on several image and text datasets demonstrate that our Curriculum-NAS can bring consistent improvement over existing weight-sharing NAS. The code is available online at https://github.com/zhouyw16/curriculum-nas.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**.

KEYWORDS

neural architecture search, curriculum learning, data uncertainty

ACM Reference Format:
Yuwei Zhou, Xin Wang, Hong Chen, Xuguang Duan, Chaoyu Guan, and Wenwu Zhu. 2022. Curriculum-NAS: Curriculum Weight-Sharing Neural Architecture Search. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), Oct. 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3548271

1 INTRODUCTION

Neural architecture search (NAS) has gained increasing attention in the community recently. For its ability to automate the process of neural architecture designing, NAS has been widely applied to various multimedia tasks including image classification [33, 43, 62], object detection [9, 20, 41], semantic segmentation [7, 34, 40, 58], text representation [52], neural machine translation [48], language model [36, 42, 61], etc. Existing neural architecture search algorithms can be mainly divided into two branches [15, 18], the multi-trial methods and the weight-sharing methods, based on whether there exists a supernet with shared parameters. The multi-trial methods [3, 27, 35, 39, 43, 54, 60, 61] rely on a large number of trials on training different individual architectures till convergence to find the suitable neural architectures, which brings prohibitive computational cost. In contrast, weight-sharing methods [1, 4, 5, 8, 16, 17, 19, 23, 31, 32, 36, 42, 55, 56] force all architectures to share their trainable weights in order to avoid the computational bottleneck of repeat training from scratch in the searching procedure.

Recently, the weight-sharing strategy has been widely adopted in various NAS algorithms including random search based ones [32], reinforcement learning based ones [42], differentiable ones [36], evolution based ones [23] and embedding based ones [37], etc. The sharing of trainable weights and non-repeated training procedure greatly reduce training computation cost and significantly improve the searching efficiency. Specifically, the weight-sharing approach features a supernet which subsumes all possible architectures in a predefined search space. The trained weights of the supernet are inherited and shared by each of its subnets. The procedure of searching architectures is the combination of one-shot training on the supernet and evaluation on its multiple subnets. Therefore, the weight-sharing strategy speeds up NAS by hundreds or even thousands of times and helps to popularize NAS in more areas.

Despite the success of current weight-sharing NAS algorithms, they fail to consider the influence of data distribution on searching architectures and merely treat all data samples equally. To explain this problem, we conduct a pre-experiment on the influence of data on various architectures. Concretely, we sample 5 common neural networks [25, 26, 44, 46, 57] for image classifications and 5 architectures from the DARTS search space [36] and plot the similarity of their errors on training and test samples of CIFAR-10 [29]. The details of the 10 networks above are listed in the supplementary. As is illustrated in Figure 1, the similarity scores between arbitrary two architectures are all less than 0.4. This phenomenon indicates architecture is one of the reasons that affect how well a model fits

[*] Corresponding authors.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10.
https://doi.org/10.1145/3503161.3548271

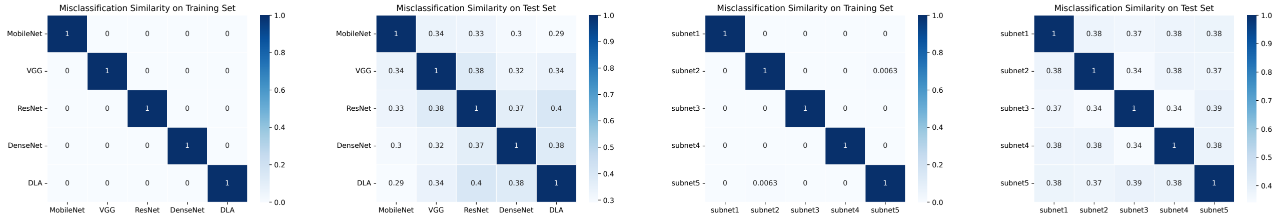


Figure 1: Our observation of the misclassification similarity, which is defined as follows: sim $= \frac{|W_i \cap W_j|}{|W_i \cup W_j|}$**, where** $W = \{y_{pred} \neq y_{true}\}$ **is the misclassification set. The former two subfigures show the misclassification similarity of common neural networks on training set and test set of CIFAR-10. And the latter two show the randomly sampled subnets from DARTS searching space.**

data and the ability of architectures to fit data is distinct. Correspondingly, the influence of data on architectures is different, e.g., data samples may be easy for some architectures while difficult for others. As a result, there are always some architectures that can perform better on early data samples. In weight-sharing NAS where all architectures and weights are nested in one supernet, those with better early performance are more than likely to maintain their high probability of being sampled [12] because they may improve themselves at the cost of suppressing other architectures which are sharing weights with them [13]. Therefore, some potentially optimal architectures may not have sufficient opportunities to be fully trained and discovered in the whole searching process, which leads to sub-optimal results and deteriorating performance.

In this paper, we propose to solve the problem by upweighting and valuing the data which is more important and worth learning, so that the subnets with relatively underwhelming performance can catch up with those having better early performance, and thus the promising ones can be discovered and searched with much higher probabilities. This idea is inspired by human education, where students should devote more effort to important and representative problems. However, simultaneously determining the proper data weights and the best architecture poses great challenges to this task. As is illustrated in Figure 2, existing works are designed to only search for the architectures, failing to solve the above challenges.

To tackle the challenges, we propose a curriculum weight-sharing neural architecture search (Curriculum-NAS) in this paper. Specifically, we design a training schedule by reweighting data samples to realize a curriculum strategy for weight-sharing NAS. We first initiate data distribution to be the same as its original. Then, in every searching iteration, we sample multiple subnets with weights and architectures inherited from the supernet, and model the outputs of each data sample through multiple subnets as a Gaussian distribution with mean given by the ground truth. We denote the standard deviation of the Gaussian as the uncertainty of the data and softly select the more uncertain data by assigning higher weights to them. Next, we conduct the weight-sharing NAS with the adjusted data distribution and train the supernet till convergence. Finally, we evaluate the candidate subnets and search for the final architecture.

The key insight of our proposed Curriculum-NAS model is that uncertain data is important and has great influence on architectures, and the emphasis on it can help architectures with poor performance at the current searching stage to be better trained and those with great potential may obtain more probability of standing out in the evaluation stage. To verify the efficacy as well as the universality of our method, we apply it to 6 popular weight-sharing NAS algorithms of various types and conduct experiments on 4 datasets for image classification and language model tasks. The empirical results show that Curriculum-NAS can bring consistent improvement over existing weight-sharing NAS methods. Furthermore, we conduct a comparative study by tracing the rankings of searched architectures in several vanilla NAS algorithms and our Curriculum-NAS model to analyze the beneficial roles of our method. To summarize, our contributions are listed as follows,

- We propose Curriculum-NAS, a novel curriculum learning based training framework for weight-sharing NAS algorithms.
- We propose a novel mechanism to evaluate data uncertainty by utilizing multiple subnets included in weight-sharing NAS.
- We demonstrate that our curriculum learning based training framework can be applied to various popular weight-sharing NAS algorithms, which shows its universality.
- Empirical experiments on both image and text datasets show that our proposed Curriculum-NAS approach is able to bring consistent improvement over existing weight-sharing NAS baselines.

2 PRELIMINARY

For simplicity of description, we first review the general formulation of NAS. Let A be the architecture search space and $a \in A$ is a candidate architecture. In weight-sharing NAS, A is designed as a supernet architecture represented by a directed acyclic graph (DAG) and correspondingly a is a subnet architecture represented by a subgraph of the DAG. The trainable weights of every subnet $w(a)$ all come from $W(A)$, i.e. the trained weights of the supernet. To obtain the optimal architecture a^* from A, a bi-level optimization problem of architectures and weights has to be solved:

$$\begin{aligned} a^* &= \arg\max_{a} \mathrm{ACC}_{val}(w^*(a), a), \\ \text{s.t.} \quad w^*(a) &= \arg\min_{w} L_{train}(w, a). \end{aligned} \tag{1}$$

Particularly, in some differentiable NAS algorithms, the bi-level optimization problem can also be formulated as:

$$\begin{aligned} a^* &= \arg\min_{a} L_{val}(w^*(a), a), \\ \text{s.t.} \quad w^*(a) &= \arg\min_{w} L_{train}(w, a), \end{aligned} \tag{2}$$

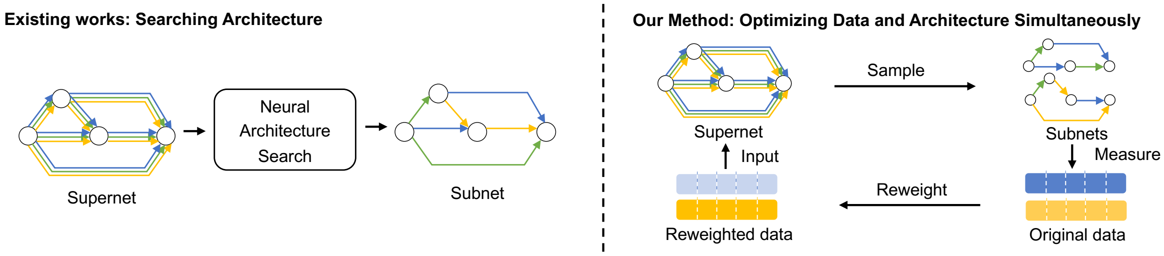

Existing works: Searching Architecture
Supernet
Neural Architecture Search
Subnet
Our Method: Optimizing Data and Architecture Simultaneously
Supernet
Sample
Subnets
Input
Measure
Reweight
Reweighted data
Original data

Figure 2: Existing works v.s. our method.

where $L_{train}(\cdot)$ and $L_{val}(\cdot)$ are the loss function on training and validation set respectively, $\text{ACC}_{val}(\cdot)$ is the accuracy on validation set, and $w^*(a)$ is the optimal weights of the subnet with the architecture a. The maximum $\text{ACC}_{val}(\cdot)$ and the minimum $L_{val}(\cdot)$ both represent the best performance on the validation set.

In the following sections, we will describe our method under the general NAS optimization process in Eq. (1).

3 METHOD: CURRICULUM-NAS

In this section, we propose our Curriculum-NAS, which applies curriculum learning to weight-sharing NAS by automatically adjusting the data distribution during the searching process. First, we formulate our method (Section 3.1). Then, we present our curriculum approach based on data uncertainty (Section 3.2), data reweighting (Section 3.3) and subnets sampling (Section 3.4). At last, we summarize the overall process of Curriculum-NAS (Section 3.5).

3.1 Formulation

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_i^N$, where each data sample x_i is attached with its label y_i. Let $v = \{v_i\}^N \in \mathcal{R}^N$ be a vector of data weights for each sample in dataset. Then our method is to collaborate the data weights into weight-sharing NAS algorithms:

$$a^* = \arg\max_a \text{ACC}_{val}(w^*(a,v), a), \tag{3a}$$

$$\text{s.t.} \quad w^*(a,v) = \arg\min_w L_{train}(w,a,v). \tag{3b}$$

As is shown in Eq. (3), the searching procedure takes data into consideration and learns a curriculum by reweighting it. The data weights impact the training procedure by scaling the loss:

$$L_{train}(w,a,v) = \sum_i^N v_i \cdot l_i(w,a), \tag{4}$$

where $l_i(w,a)$ is the loss of i^{th} data sample for architecture a parameterized by w. A weight value v_i close to zero means a soft discarding of the data sample, while a weight value equal to or even greater than one represents a selection of the data sample. Therefore, our Curriculum-NAS searches architectures in a curriculum manner where the selection of data is learned in every training iteration based on the assessment of multiple subnets sampled from the supernet, and at the same time, plays an important role in the optimization of the supernet.

3.2 Data Uncertainty

The key part of our method is how to reweight training data. As is mentioned in Section 1, each data sample has its influence on architectures. We measure it by data uncertainty, which could be reflected by the deviation among the outputs of multiple architectures. Concretely, we sample K subnets from the supernet and evaluate each data sample with these subnets, and then reweight the training data samples. We present the concrete process of calculating data uncertainty in this subsection.

Let $f_k = f_k(w_k, a_k)$ be a subnet with weights w_k and architecture a_k, where $1 \le k \le K$. For a data sample x, its output from the subnet f_k is $f_k(x)$ and its label is y. We assume the likelihood of a data sample's output given by its label is a Gaussian distribution and define the Bayesian probabilistic model as:

$$p(f(x)|y) = \mathcal{N}(y, \sigma^2), \tag{5}$$

where the label y is the mean, and the σ is the standard deviation which could be regarded as the uncertainty of a data sample to the architectures.

For K subnets outputs, we assume that they follow independent Gaussian distributions:

$$p(f_1(x), f_2(x), ..., f_K(x)|y) = \prod_{k=1}^{K} p(f_k(x)|y). \tag{6}$$

According to Maximum Likelihood Principle, we optimize the following objective:

$$\begin{aligned} &\max_\sigma \log p(f_1(x), f_2(x), ..., f_K(x)|y) \\ &= \min_\sigma - \sum_{k=1}^{K} \log p(f_k(x)|y). \end{aligned} \tag{7}$$

For regression loss function $L(y, f(x)) = (y - f(x))^2$, it has:

$$\begin{aligned} &- \sum_{k=1}^{K} \log p(f_k(x)|y) \\ &\propto \sum_{k=1}^{K} \frac{1}{2\sigma^2}(y - f_k(x))^2 + \log\sigma \\ &= \sum_{k=1}^{K} \frac{1}{2\sigma^2} L(y, f_k(x)) + \log\sigma. \end{aligned} \tag{8}$$

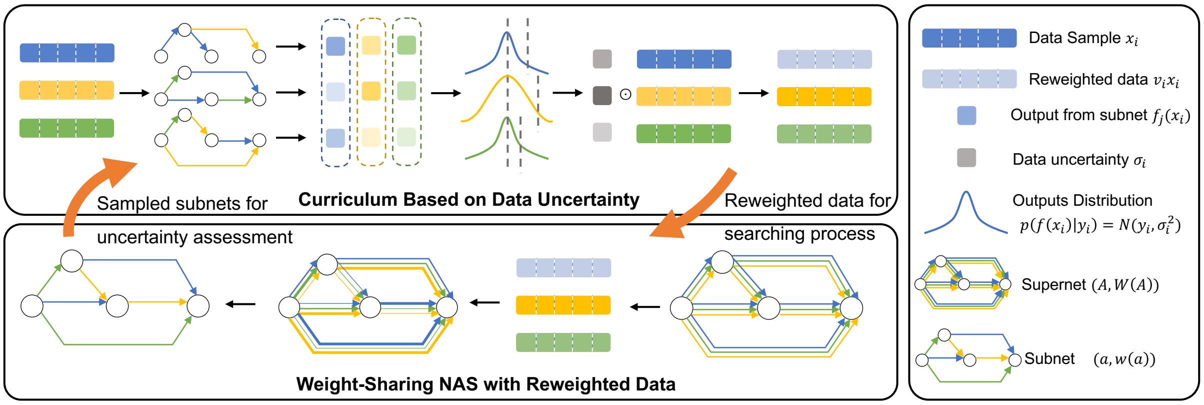

Sampled subnets for uncertainty assessment
Curriculum Based on Data Uncertainty
Reweighted data for searching process
Weight-Sharing NAS with Reweighted Data
Data Sample x_i
Reweighted data $v_i x_i$
Output from subnet $f_j(x_i)$
Data uncertainty σ_i
Outputs Distribution $p(f(x_i)|y_i) = N(y_i, \sigma_i^2)$
Supernet $(A, W(A))$
Subnet $(a, w(a))$

Figure 3: The framework of Curriculum-NAS. *Bottom:* The searching process of a weight-sharing NAS with reweighted data. *Upper:* The curriculum approach based on data uncertainty. A data sample has multiple outputs from multiple subnets and it is assumed that the outputs follow a Gaussian distribution with mean given by its label and standard deviation representing its uncertainty, which decides the weight of the data sample.

For classification loss function $L(y, f(x)) = -y \log \mathrm{Softmax} f(x)$, the likelihood can be interpreted as a scaled version of Boltzmann distribution [28]:

$$\begin{aligned} & -\sum_{k=1}^{K} \log p(f_k(x)|y, \sigma) \\ =& -\sum_{k=1}^{K} \log \mathrm{Softmax}(\frac{1}{\sigma^2} f_k(x)) \\ =& \sum_{k=1}^{K} -\frac{1}{\sigma^2}[f_k(x)]_y + \log \sum_{y'} \exp(\frac{1}{\sigma^2}[f_k(x)]_{y'}) \\ =& \sum_{k=1}^{K} \frac{1}{\sigma^2} L(y, f_k(x)) + \log \frac{\sum_{y'} \exp(\frac{1}{\sigma^2}[f_k(x)]_{y'})}{(\sum_{y'} \exp([f_k(x)]_{y'}))^{\frac{1}{\sigma^2}}} \\ \approx & \sum_{k=1}^{K} \frac{1}{\sigma^2} L(y, f_k(x)) + \log \sigma, \end{aligned} \tag{9}$$

where $[f(x)]_y$ represents the y^{th} element of the output vector and the approximately equal sign can become a equal one when $\sigma \longrightarrow 1$.

Without loss of generality, we take regression loss Eq. (8) as an example and directly calculate the value of σ by minimizing the following objective:

$$\begin{aligned} & \min \sum_{k=1}^{K} \left(\frac{1}{2\sigma^2} L(y, f_k(x)) + \log \sigma \right) \\ \Longleftrightarrow & \frac{\partial}{\partial \sigma} \left(\sum_{k=1}^{K} \left(\frac{1}{2\sigma^2} L(y, f_k(x)) + \log \sigma \right) \right) = 0 \\ \Longleftrightarrow & \sigma = \sqrt{\frac{1}{K} \sum_{k=1}^{K} L(y, f_k(x))} \end{aligned} \tag{10}$$

3.3 Data Reweighting

In this part, we map the uncertainty of a data sample to its weight in order to softly select data and schedule training. Here we borrow the idea of Hard Example Mining (HEM) [45] that harder data is more informative. Therefore, we assign higher weights to the data samples with higher uncertainty, which deserve greater importance and more emphasis. But unlike HEM, the weights of our method are dynamic to avoid the overfitting problem faced by HEM.

Besides, the weights of data samples should keep their scales instead of increasing or decreasing unlimitedly to prevent collapsing. For instance, if all subnets have the correct prediction and the uncertainty of each data becomes zero, the data weights should become one instead of zero, which makes our Curriculum NAS degenerate into the original NAS and perform at least no worse than that. More specifically, since the original data weights are $v = \{1\}^N$ and the sum of them is N, we keep the sum N at every training iteration. Therefore, we define the data weights as in Eq. (11), where $\sigma = \{\sigma_i\}_{i=1}^N$ is the uncertainty vector of all the data samples.

$$v = N \frac{\sigma}{\|\sigma\|_1}. \tag{11}$$

3.4 Subnets Sampling

In the above subsection, the K subnets are mentioned a lot, which is critical to evaluating the data uncertainty. And in this subsection, we present the method to sample these subnets from the supernet.

We evaluate the data uncertainty through several subnets instead of all subnets due to the limitation of computational complexity. Specifically, we pick up the top-K subnets in every training iteration through the following *architecture sampling* and *weights inheriting* stages, because we assume that the more confusing results among the excellent subnets are more worth learning.

Architecture Sampling. Let π be the searching policy, through which various NAS algorithms pick up their top-1 architecture:

$$\{a\} \sim \pi(A). \tag{12}$$

Then we define π_K to pick up the top-K architectures:

$$\{a_1, a_2, ..., a_K\} \sim \pi_K(A). \tag{13}$$

For different types of NAS algorithms, we present their corresponding sampling methods. For reinforcement learning algorithms like ENAS [42], we sample the first K architectures constructed by the controller. For differentiable algorithms like DARTS [36], we sort all combinations of operations by the product of their probabilities and sample the architectures with top-K products. For an evolutionary search algorithm like SPOS [23], we sample the searched architectures with top-K accuracy. For a random search algorithm [32], we simply sample the first K searched architectures.

Weights Inheriting. We directly derive the trainable weights of subnets from the supernet for two reasons. First, the current weights reflect the current learning state of the supernet. Second, inheriting instead of retraining largely saves computational costs. Therefore, we derive the weights connected to the architectures:

$$w_k = W(a_k). \tag{14}$$

3.5 Curriculum Weight-Sharing NAS

In the subsection, we present the complete process of Curriculum-NAS, an automatic approach to optimize architecture search and data selection for weight-sharing NAS based on data uncertainty. It is illustrated in figure 3 and elaborated in algorithm 1.

First, we initiate the architecture search space according to the required task and data and construct the weight-sharing supernet. We initiate the training data distribution the same as the original. In every iteration of the training stage, we sample subnets including network weights and architectures from the supernet and assess the uncertainty of each data sample to reweight them. Then we train the supernet based on the adjusted data distribution to update its trainable weights and architecture searching policy. After that, we evaluate the searched architectures through validation loss or accuracy. Finally, we retrain and test the final searched architecture on the test set to achieve the results.

To sum up, the above algorithm makes full use of the fact that multiple subnets are included naturally in NAS and evaluates data uncertainty with these subnets. The reweighted data based on uncertainty help the promising subnets be more likely to be searched.

Algorithm 1 Curriculum-NAS Algorithm

```
Require: The number of sampled subnets $K$, the original weight-sharing NAS algorithm like DARTS, ENAS, etc.
Initialize data weights $v = \{1\}^N$, supernet weights $W$, supernet architecture $A$, architecture searching policy $\pi$.
while not convergent do
    Sample $K$ subnets $f_k(w_k, a_k)$ via Eq. (13) and (14);
    Assess data uncertainty $\sigma$ with $K$ subnets via Eq. (10);
    Update data weights $v$ via Eq. (11);
    Update supernet weights $w \in W$ via Eq. (3b)
    Update architecture searching policy $\pi$ via the same way as the original weight-sharing NAS algorithm.
end while
Derive the final architecture $a^* \in A$ via Eq. (3a).
```

4 EMPIRICAL RESULTS

In this section, we introduce the experimental setup (Section 4.1), present the performance of our Curriculum-NAS (Section 4.2), analyze its efficacy through a comparative study (Section 4.4) and discuss the influence of sampled subnet number K (Section 4.5).

4.1 Experimental Setup

To validate the effectiveness of our method, we conduct experiments on both image classification and language model tasks.

Image Classification. We conduct our image classification experiments on the basis of *NATS-BENCH* [15], an extensive version of *NAS-BENCH-201* [18], which is an algorithm-agnostic benchmark for NAS algorithms evaluation. Its predefined skeleton includes stacks of cells, each of which is a complete directed acyclic graph with 4 nodes and 6 edges. Every edge is a choice among 5 operations: zeroize, skip connection, 1-by-1 convolution, 3-by-3 convolution, and 3-by-3 average pooling. Therefore, the size of the search space is $5^6 = 15625$.

The datasets where all of the architectures have been trained and evaluated in advance are CIFAR-10, CIFAR-100 and ImageNet-16-120. The three image classification datasets are split into training, validation and test set according to the proposed configuration in *NATS-BENCH* so that the comparisons among architectures are fair and justified.

- **CIFAR-10** [29]: A prevalent image classification dataset with 60 × 60 color images in 10 classes. In *NATS-BENCH* configuration, the original training set is split in half into new training and validation set.
- **CIFAR-100** [29]: The dataset just like CIFAR-10 but with images in 100 classes. In *NATS-BENCH* configuration, the original test set is split in half into new validation and test set.
- **ImageNet-16-120** [11, 14]: A down-sampled version of the prevalent image classification ImageNet. It consists of 16 color images in 120 classes. In *NATS-BENCH* configuration, the original test set is split in half into new validation and test set.

The performance on the three datasets is evaluated by top-1 accuracy. As the benchmark provides its API for querying the performances of architectures, it is convenient to compare NAS algorithms based on the reported results of the final searched architecture.

Language Model. Apart from the image classification task, we also conduct language model experiments on the basis of the same setting as [42] and [36]. We run the official codes published by their authors, apply our method to the original searching process and compare the results with/without our curriculum manner.

The dataset on which the experiments are performed is Penn Treebank(PTB).

- **PTB** [38]: A popular dataset for language model. We preprocess PTB in the same way as [42] and [36].

The performance on PTB is evaluated by perplexity. We report the best test perplexity based on the best architecture in the searching process and the best trainable parameters in the retraining process.

Comparable Methods. To evaluate the generality of our method on weight-sharing NAS, we take multiple kinds of state-of-the-art

Table 1: Dataset Statistics

Dataset	Task	Classes	Tokens	Training	Validation	Test
CIFAR-10	Image Classification	10	-	25000	25000	10000
CIFAR-100	Image Classification	100	-	50000	5000	5000
ImageNet-16-120	Image Classification	120	-	151700	3000	3000
PTB	Language Model	-	10000	42068	3370	3761

NAS algorithms as baselines, including random search algorithms, differentiable algorithms and reinforcement learning algorithms.

- **CL-X**: X represents one of the following baseline algorithms and CL-X is our method applied to the algorithm X.
- **RSPS** [32]: A random search algorithm with early-stopping and weight-sharing strategy. It is proposed to serve as a competitive NAS baseline.
- **DARTS-V1** [36]: The first-order DATRS, a differentiable algorithm with mixed operations encoded by continuous parameters. Its architecture gradient is given by $\nabla_a L_{val}(w, a)$.
- **DARTS-V2** [36]: The second-order DATRS with its architecture updated by descending $\nabla_a L_{val}(w - \xi \nabla_w L_{train}(w, a), a)$. It costs more time than DARTS-V1 but generally performs better.
- **GDAS** [17]: An algorithm with a differentiable sampler over the supernet to optimize the architecture.
- **SETN** [16]: A differentiable algorithm with a learnable evaluator to indicate the probability of each architecture having lower validation loss.
- **ENAS** [42]: A classical reinforcement learning based algorithm with a learnable controller to generate architectures and take the validation accuracy as the reward.

Hyperparameters Setting. To fairly assess our method, we take the exact same hyperparameters and configurations as the benchmark or the baselines did. Besides, we set the number of sampled subnets $K \in \{1, 3, 5, 7, 9, 15\}$ in our method. With the hyper-parameters above, we report the mean and standard deviation results of 3 runs with different fixed random seeds in image classification task and report the 1 run result in language model task due to the limitation of time and computation resources.

4.2 Results of Image Classification

We follow the image classification experiment setting strictly after the *NATS-BENCH*, apply our method to 6 weight-sharing NAS algorithms and evaluate the performances on 3 datasets. All algorithms train their supernets and search architectures on the training and validation set, and test their final searched architecture on test set.

The main results presented in Table 2 show that our method can be widely applied to various weight-sharing NAS algorithms and outperform these baselines on datasets with different scales. The improvement may result from the adjustment of the data distribution allowing the promising architectures to have more probability to be fully trained so that they can stand out in earlier stages.

Besides, it is observed that the degree of improvement depends on the baselines and datasets. For instance, CL-DARTS-V1 and CL-DARTS-V2 have their edges over DARTS-V1 and DARTS-V2 on large datasets (CIFAR-100 and ImageNet-16-120), while CL-GDAS performs much better than GDAS on the small dataset (CIFAR-10). It is most likely due to their different convergence rates, which is mentioned in *NATS-BENCH*. Although DARTS-V1, DARTS-V2 and GDAS all belong to differentiable NAS, the former two converge too quickly on CIFAR-10 and the latter one converges too slowly on CIFAR-100 and ImageNet-16-120. Except for these situations, our method brings consistent improvement and achieves good results.

Table 2: Image classification Accuracies(%). The results of the original NAS algorithms come from *NATS-BENCH* paper. The bold font represents the better performance. The row of *Optimal* presents the highest accuracy on each dataset.

Method	CIFAR-10	CIFAR-100	ImageNet-16-120
RSPS	91.05±0.66	68.26±0.96	40.69±0.36
CL-RSPS	**93.25±0.41**	**69.49±1.33**	**42.58±0.97**
DARTS-V1	**59.84±7.84**	61.26±4.43	37.88±2.91
CL-DARTS-V1	**59.84±7.83**	**67.19±0.65**	**42.50±1.46**
DARTS-V2	65.38±7.84	60.49±4.95	36.79±7.59
CL-DARTS-V2	**70.92±0.00**	**67.46±1.00**	**38.71±4.05**
GDAS	93.23±0.58	**68.17±2.50**	**39.40±0.00**
CL-GDAS	**93.54±0.10**	**68.17±0.33**	**39.40±0.00**
SETN	92.72±0.73	69.36±1.72	39.51±0.33
CL-SETN	**93.01±0.24**	**71.03±0.11**	**39.68±0.16**
ENAS	**93.76±0.00**	70.67±0.62	41.44±0.00
CL-ENAS	**93.76±0.00**	**70.75±0.11**	**42.67±1.73**
Optimal	94.37	73.51	46.20

Table 3: Language Model Perplexities. The results of the original NAS algorithms are obtained by running their official code published in the setting of fixed sequence length. The bold font represents the better performance.

Method	Valid PPL	Test PPL
DARTS-V1	62.61	60.08
CL-DARTS-V1	**61.31**	**58.78**
DARTS-V2	60.52	58.37
CL-DARTS-V2	**59.29**	**56.94**
ENAS	62.66	59.93
CL-ENAS	**61.58**	**58.38**

4.3 Results of Language Model

We basically follow the language model experiment after ENAS and DARTS respectively. However, the two baselines train their supernets with variable-length text sequences, which can hardly be regarded as a definite data sample. Therefore, we have to fix the

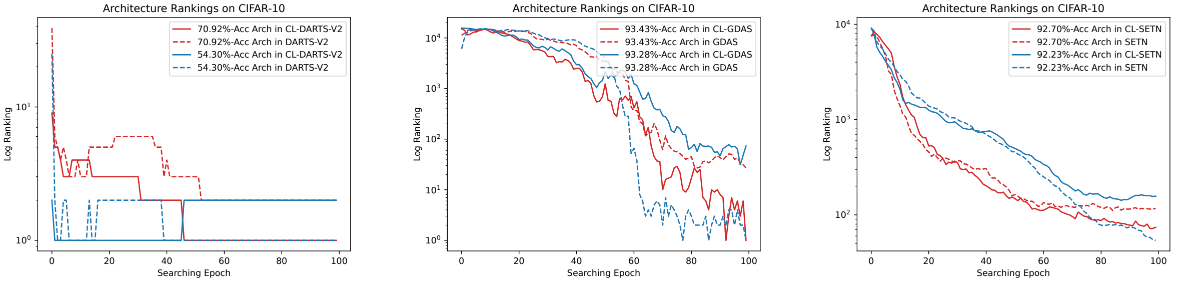


Figure 4: The rankings of final searched architectures from *left*: DARTS-V2, CL-DARTS-V2, *middle*: GDAS, CL-GDAS, and *right*: SETN, CL-SETN. Let the architecture searched by Curriculum-NAS be a_1 and by original NAS a_2. We care about the changing of their rankings in the two searching procedures respectively, so each subfigure contains 4 lines including *red solid*: a_1 in Curriculum-NAS, *red dash*: a_1 in original NAS, *blue dash*: a_2 in original NAS and *blue solid*: a_2 in Curriculum-NAS.

length of each text sequence in order to assess their uncertainty and assign appropriate weights. We apply our method to 3 weight-sharing NAS algorithms and evaluate the performances on PTB dataset. All methods train their supernets and search architectures on the training and validation set, and test their final searched architecture on test set after retraining from scratch.

We report the results in Table 3, which indicates our approach achieves improvements over DARTS-V1, DARTS-V2 and ENAS on valid and test perplexity.

4.4 Efficacy Analysis

To further analyze the efficacy of our method, we trace the ranking of the final architectures searched by our method and the baselines. We plot the changing of rankings on CIFAR-10 in Figure 4.

For instance, let a_1 and a_2 be the final architectures searched by CL-GDAS and GDAS respectively. Since a_1 has the accuracy of 93.43% on CIFAR-10 and a_2 only has 93.28%, we call a_1 a more promising architecture. The results in the middle of Figure 4 show that (1) a_1 in CL-GDAS (red solid) ranks ahead of a_2 in CL-GDAS (blue solid) most of the time, illustrating that the more promising one ranks ahead of the less promising one in our method. (2) a_1 in CL-GDAS (red solid) ranks ahead of a_1 in original GDAS (red dash) most of the time, illustrating that the more promising one in our method ranks ahead of itself in the baseline. The results of CL-DARTS-V2 and CL-SETN lead to the similar conclusion that a more promising architecture in our method ranks ahead most of the time especially in the later stage of training because our curriculum approach plays a gradual role in better training the more promising architecture. The results verify the efficacy of our method.

We only plot the ranking curves of DATRS, GDAS and ENAS instead of RSPS and ENAS because we are not able to get an exact architecture ranking from the latter two algorithms. Besides, the final searched architecture in SETN does not rank first in the last epoch because SETN picks up multiple candidate architectures and chooses the one with the highest validation accuracy.

4.5 Hyperparameter Sensitivity

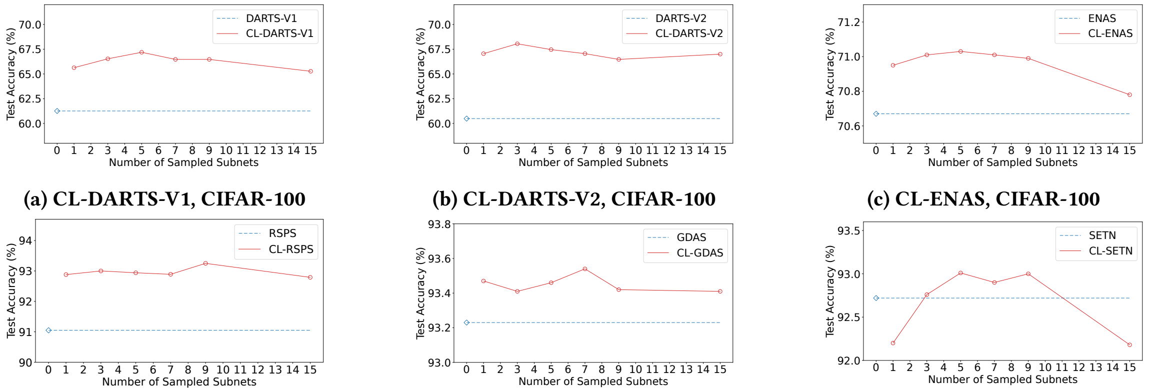
Another issue worth discussing is the number of sampled subnets K. We conduct the experiments with different K values and keep other hyperparameters the same. The results are shown in Figure 5, which illustrates that our method outperforms the baselines with different K in most cases on both CIFAR-10 and CIFAR-100. From the overall trend, we observe that when K is small like $K = 1$ or K is large like $K = 15$, the accuracy will be relatively low. The reason is likely due to the following reasons. When K is small, the uncertainty measured by too few subnets is not sufficient. Especially when $K = 1$, the uncertainty degenerates into the difficulty decided by one subnet. On the other hand, when K is large, the evaluation of too many subnets weakens the role of data uncertainty, e.g., all data weights tend to equal to one because of the normalization process in Equation 11 and our method degenerates into the original weight-sharing NAS algorithm. And the case can be worse when the really bad architectures are sampled and they dominate the uncertainty.

5 RELATED WORK

5.1 Neural Architecture Search

Neural architecture search (NAS) serves as an effective way to design neural architecture automatically. The first proposed NAS [61] is based on reinforcement learning, which trains a controller to generate all layers and their connections. It achieves great performance on multimedia datasets, but it costs too much computation and time. Therefore, the following NAS works strive to reduce the cost. PNAS [35] searches architectures in a progressive manner. NASNet [62] does not search architectures but cells, which stack up to form an architecture. NAO [37] embeds architectures into low-dimension space. ENAS [42] forces all architectures to share their trainable architectures to avoid repeat training from scratch, which is named weight-sharing NAS and is widely used in current NAS algorithms.

Weight-Sharing NAS. The appearance of weight-sharing NAS aims to alleviate the unaffordable time and computation cost of NAS. Besides the baselines mentioned in section 4.1, there are numerous weight-sharing NAS works. For example, SPOS [23] presents a single path one-shot approach with uniform sampling and a search procedure based on evolutionary algorithm. DenseNAS [19], P-DARTS [8], PC-DARTS [56], SGAS [31], etc. are inspired by DARTS and focus on its weakness. They respectively manage to speed up searching, expand the width and depth of models, and bridge the gap



(a) CL-DARTS-V1, CIFAR-100

(b) CL-DARTS-V2, CIFAR-100

(c) CL-ENAS, CIFAR-100

(d) CL-RSPS, CIFAR-10

(e) CL-GDAS, CIFAR-10

(f) CL-SETN, CIFAR-10

Figure 5: The influence of sampled subnets number K on accuracy. Each subfigure contains 2 lines including *red*: the accuracy of Curriculum-NAS with different K and *blue*: the accuracy of the original NAS.

between validation and test. To sum up, weight-sharing NAS indeed improves the efficiency of searching procedure significantly and maintains the efficacy basically. But it is still confronted with the problem of decoupled weights and the initially better architecture taking up the leading position. Therefore, we introduce curriculum learning into it to adjust data distribution in order to fully train the supernet and its promising subnets.

5.2 Curriculum Learning

Curriculum learning (CL) [2, 51] is a strategy of training from ease, imitating the procedure of human learning with curricula. It is widely used in deep learning to improve model generalization and accelerate training convergence. The simplest work of CL is named Baby Step [2, 49], which determines the difficulty and input order of data in advance. However, the predefined method is not flexible and satisfying. Therefore, the Self-Paced method [30] is proposed to select data samples automatically according to the training loss. Besides, there are Transfer Teacher [24, 53], Reinforcement Learning Teacher [21, 59], and other automatic CL frameworks based on the specific data, model and task [6, 47]. The key parts of CL are a difficulty measurer to judge the difficulty of data samples and a training scheduler to decide the input sequence of data. Our method actually belongs to the generalized definition of curriculum learning because we measure the data difficulty by its uncertainty and schedule the training process by softly selecting data. Our method also adopts the idea of Hard Example Mining (HEM) [45], a variant of CL, which assumes harder data is more informative and trains from the hardest data, so we assign higher weights to the more uncertain data.

Apart from this, it should be noted that Curriculum Learning has been combined with NAS recently. InstaNAS [10] increases the task difficulty through a reinforcement learning procedure. GTN [50] replaces each real training data with synthetic one in a curriculum manner, which however gives up real data distribution and is only applied on the basis of NAO [37]. CNAS [22] applies curriculum in search space instead of data. Therefore, our Curriculum-NAS is far different from the methods above.

5.3 Differences from Other Works

Multi-Task Loss. Its definition [28] is also related to uncertainty:

$$l_{multitask}(l_1, ..., l_n, \sigma_1, ..., \sigma_n) = \sum_{i=1}^{n} \frac{l_i}{2\sigma_i^2} + \log \sigma_i.$$

However, its uncertainty targets different tasks and is inversely related to the weights. Besides, the uncertainty is regarded as a trainable parameter to be optimized. The above two points are the major difference between our method and the multi-task loss.

Data Augmentation. Another related work to our method is data augmentation which also changes the data distribution. But it is not conflicted with our reweighting strategy because our method is a soft selection of data and augmentation is an extension of data, both of which can be applied simultaneously. More importantly, the augmentation is usually predefined or preprocessed but our data distribution optimization is conducted automatically in every training iteration.

6 CONCLUSION

In this paper, inspired by curriculum learning where data distribution and learning scheduler matter, we propose our Curriculum-NAS, a curriculum training framework on weight-sharing NAS based on data uncertainty. The method takes advantage of multiple subnets included in NAS to evaluate data uncertainty. Thanks to the soft selection of data based on uncertainty, the promising architectures are more than likely to be fully trained and the NAS can search for better architectures. A future direction worth studying is to introduce curriculum learning and optimize data distribution even in the evaluation or retraining stage to improve current state-of-the-art works.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China No. 2020AAA0106300 and National Natural Science Foundation of China No. 62250008, No. 62102222.

REFERENCES

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*. PMLR, 550–559.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

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

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344* (2017).

[5] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[6] Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. SuperLoss: A Generic Loss for Robust Curriculum Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[7] 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. *Advances in neural information processing systems* 31 (2018).

[8] 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 IEEE/CVF International Conference on Computer Vision*. 1294–1303.

[9] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. 2019. Detnas: Backbone search for object detection. *Advances in Neural Information Processing Systems* 32 (2019).

[10] 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*, Vol. 34. 3577–3584.

[11] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819* (2017).

[12] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2021. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12239–12248.

[13] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. 2020. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*. Springer, 465–480.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[15] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. 2021. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[16] Xuanyi Dong and Yi Yang. 2019. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3681–3690.

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

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

[19] 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10628–10637.

[20] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7036–7045.

[21] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *international conference on machine learning*. PMLR, 1311–1320.

[22] Yong Guo, Yaofo Chen, Yin Zheng, Peilin Zhao, Jian Chen, Junzhou Huang, and Mingkui Tan. 2020. Breaking the curse of space explosion: Towards efficient nas with curriculum search. In *International Conference on Machine Learning*. PMLR, 3822–3831.

[23] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*. Springer, 544–560.

[24] Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*. PMLR, 2535–2544.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[27] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. 2018. Neural architecture search with bayesian optimisation and optimal transport. *arXiv preprint arXiv:1802.07191* (2018).

[28] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

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

[30] M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *NIPS*, Vol. 1. 2.

[31] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2020. Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1620–1630.

[32] Liam Li and Ameet Talwalkar. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*. PMLR, 367–377.

[33] Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. 2019. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*.

[34] 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 IEEE/CVF conference on computer vision and pattern recognition*. 82–92.

[35] 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 (ECCV)*. 19–34.

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

[37] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural architecture optimization. *Advances in neural information processing systems* 31 (2018).

[38] Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Using Large Corpora* (1994), 273.

[39] Renato Negrinho and Geoff Gordon. 2017. Deeparchitect: Automatically designing and training deep architectures. *arXiv preprint arXiv:1704.08792* (2017).

[40] 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 IEEE/CVF conference on computer vision and pattern recognition*. 9126–9135.

[41] Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. 2019. Efficient neural architecture transformation search in channel-level for object detection. *Advances in Neural Information Processing Systems* 32 (2019).

[42] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.

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

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[45] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 761–769.

[46] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[47] Samarth Sinha, Animesh Garg, and Hugo Larochelle. 2020. Curriculum By Smoothing. *Advances in Neural Information Processing Systems* 33 (2020).

[48] David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*. PMLR, 5877–5886.

[49] Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 751–759.

[50] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. 2020. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*. PMLR, 9206–9216.

[51] Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[52] Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyu 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*, Vol. 34. 9242–9249.

[53] Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*. PMLR, 5238–5246.

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

[55] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* (2018).

[56] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2019. PC-DARTS: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737* (2019).

[57] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2403–2412.

[58] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. 2019. Customizable architecture search for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11641–11650.

[59] Mingjun Zhao, Haijiang Wu, Di Niu, and Xiaoli Wang. 2020. Reinforced Curriculum Learning on Pre-Trained Neural Machine Translation Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9652–9659.

[60] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2423–2432.

[61] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

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