UNIVERSITY OF CALIFORNIA

Los Angeles

Towards efficient, effective, and robust Neural Architecture Search methods

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Computer Science

by

Ruochen Wang

2021

ABSTRACT OF THE THESIS

Towards efficient, effective, and robust Neural Architecture Search methods

by

Ruochen Wang
Master of Science in Computer Science
University of California, Los Angeles, 2021
Professor Cho-Jui Hsieh, Chair

Recently, Neural Architecture Search (NAS) has attracted lots of attention for its potential to democratize deep learning. For a practical end-to-end deep learning platform, NAS plays a crucial role in discovering task-specific architecture depending on users' configurations (e.g., dataset, evaluation metric, etc.). Among various search paradigms, Differentiable Neural Architecture Search is one of the most popular NAS methods for its search efficiency and simplicity, accomplished by jointly optimizing the model weight and architecture parameters in a weight-sharing supernet via gradient-based algorithms. At the end of the search phase, the operations with the largest architecture parameters will be selected to form the final architecture, with the implicit assumption that the values of architecture parameters reflect the operation strength. Despite the search efficiency, the weight-sharing supernet also shows a tendency towards non-parametric operations, resulting in shallow architectures with degenerated performance. We provide both theoretical and empirical analysis of the poor generalization observed in Differentiable NAS, which links this issue to the failure of the magnitude-based selection. Following this inspiration, we discuss two lines of methods that greatly improve the effectiveness and robustness of Differentiable NAS: The first line

proposes an alternative perturbation-based architecture selection that is shown to identify better architectures in the search space, whereas the second line aligns the architecture parameter with the strength of underlying operations. To complete the picture, an alternative paradigm to the differential architecture search (predictor-based NAS) is also presented.

The thesis of Ruochen Wang is approved.

Baharan Mirzasoleiman

Quanquan Gu

Cho-Jui Hsieh, Committee Chair

University of California, Los Angeles

2021

To my parents . . .

for their unconditional love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Prof. Cho-Jui Hsieh, who guided me thoughtfully and patiently throughout the course of study. Moreover, I am grateful to my collaborators - Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Shoukang Hu - for their effort and friendship. I also own special thanks to my colleague Xuanqing Liu, who offered me invaluable suggestions during difficult times, as well as my mentor at Microsoft Research - Kai Chen, who introduces me to the field of Neural Architecture Search in the first place. The research summarizes in this thesis would not exist without their inputs and dedications.

CHAPTER 1

Introduction

Neural Architecture Search (NAS) has been drawing increasing attention in both academia and industry for its potential to automatize the process of discovering high-performance architectures, which have long been handcrafted. Early works on NAS deploy Evolutionary Algorithm [SM02, RMS17, LSV17] and Reinforcement Learning [ZL17, PGZ18, ZYW18] to guide the architecture discovery process. These pioneering methods require training a large number of discrete architectures, and hence incur significant amount of computation costs. In the attempt to improve the efficiency of NAS, several one-shot weight-sharing methods have been proposed that dramatically cut down the search cost [BLR18, GZM19, BKZ18].

As a particularly popular instance of one-shot methods, DARTS [LSY19] enables the search process to be performed with a gradient-based optimizer in an end-to-end manner. It applies continuous relaxation that transforms the categorical choice of architectures into continuous architecture parameters α. The resulting supernet can be optimized via gradient-based methods, and the operations associated with the largest architecture parameters are selected to form the final architecture. Despite its simplicity, several works cast doubt on the effectiveness of DARTS. For example, a simple randomized search [LT19] outperforms the original DARTS; [ZES20] observes that DARTS degenerates to networks filled with parametric-free operations such as the skip connection or even random noise, leading to the poor performance of the selected architecture.

While the majority of previous research attributes the failure of DARTS to its supernet optimization [ZES20, CH20, CWC21], little has been discussed about the validity of another

important assumption: *the value of α reflects the strength of the underlying operations.* In this paper, we conduct an in-depth analysis of this problem. Surprisingly, we find that in many cases, α does not really indicate the operation importance in a supernet. Firstly, the operation associated with larger α does not necessarily result in higher validation accuracy after discretization. Secondly, as an important example, we show mathematically that the domination of skip connection observed in DARTS (i.e. α_{skip} becomes larger than other operations.) is in fact a reasonable outcome of the supernet's optimization but becomes problematic when we rely on α to select the best operation.

If α is not a good indicator of operation strength, how should we select the final architecture from a pretrained supernet? We discuss two lines of solutions. The first line bypasses α and directly measures the operation strength based on its contribution to the supernet performance in an adversarial fashion. Concretely, we propose an alternative perturbation-based architecture selection method. Given a pretrained supernet, the best operation on an edge is selected and discretized based on how much it perturbs the supernet accuracy; The final architecture is derived edge by edge, with fine-tuning in between so that the supernet remains converged for every operation decision. The second line aims at aligning the architecture selection with differentiable architecture search, by formulating architecture search as a distribution learning problem, which induces implicit hessian regularization [CWC21]. Empirical evaluations show that both lines of methods are able to drastically improve the search performance and the robustness of differentiable NAS.

To make the picture a bit more complete, the remaining chapters of the paper discusses an alternative architecture search paradigm that does not rely on weight-sharing supernet to make search decisions, called predictor-based NAS. Predictor-based methods use a tiny subset of architectures to train a surrogate model to predict the accuracy of each individual architecture. This line of methods do not rely on a weight-sharing one-shot supernet, and thus is free from the various inductive biases of differentiable NAS methods.

CHAPTER 2

Understanding Differentiable NAS

Despite the search efficiency of Differentiable Neural Architecture Search (DARTS), several work finds that it generalizes poorly to a wide range of search spaces (Section 2.2). For example, [ZES20] observes that DARTS degenerates to networks filled with parametric-free operations such as the skip connection or even random noise, leading to the poor performance of the selected architecture. While previous analysis attributes the poor generalization of DARTS to the failure of supernet optimization, we show both empirically and mathematically that it is in fact caused by the magnitude-based architecture selection method (Section 2.3).

2.1 Differentiable Architecture Search Framework

We start by reviewing the formulation of DARTS. DARTS' search space consists of repetitions of cell-based microstructures. Every cell can be viewed as a DAG with N nodes and E edges, where each node represents a latent feature map x^i, and each edge is associated with an operation o (e.g. *skip_connect*, *sep_conv_3x3*) from the search space $\mathcal{O}$. Continuous relaxation is then applied to this search space: Concretely, every operation on an edge is activated during the search phase, with their outputs mixed by the architecture parameter α to form the final mixed output of that edge $\bar{m}(x^i) = \sum_{o \in \mathcal{O}} \frac{\exp \alpha_o}{\sum_{o'} \exp \alpha_{o'}} o(x^i)$. This particular formulation allows the architecture search to be performed in a differentiable manner: DARTS jointly optimizes α and model weight w with the following bilevel objective via

alternative gradient updates:

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*, \alpha) \qquad \text{s.t.} \qquad w^* = \arg\min_{w} \ \mathcal{L}_{train}(w, \alpha). \tag{2.1}$$

We refer to the continuous relaxed network used in the search phase as the **supernet** of DARTS. At the end of the search phase, the operation associated with the largest α_o on each edge will be selected from the supernet to form the final architecture. An important implicit assumption here is that the magnitude of $\alpha = (\alpha_o)_o$ represents the strength of the underlying operation.

2.2 Failure Mode Analysis of DARTS

Several works cast doubt on the generalization of DARTS. [ZES20] tests DARTS on four different search spaces and observes significantly degenerated performance, resulting in architectures filled with skip connections. They empirically find that the selected architectures perform poorly when DARTS' supernet falls into high curvature areas of validation loss (captured by large dominant eigenvalues of the Hessian $\nabla^2_{\alpha,\alpha}\mathcal{L}_{val}(w, \alpha)$). While [ZES20] relates this problem to the failure of supernet training in DARTS, we examine it from the architecture selection aspects of DARTS, and show that much of DARTS' robustness issue can be alleviated by a better architecture selection method.

2.3 The pitfall of magnitude-based architecture selection in DARTS

In this section, we put forward the opinion that the architecture parameter α does not necessarily represent the strength of the underlying operation in general, in direct contradiction to the claim made in DARTS. As an important example, we mathematically justify that the skip connection domination phenomenon observed in DARTS is reasonable by itself, and becomes problematic when combined with the magnitude-based architecture selection.

2.3.1 α may not represent the operation strength

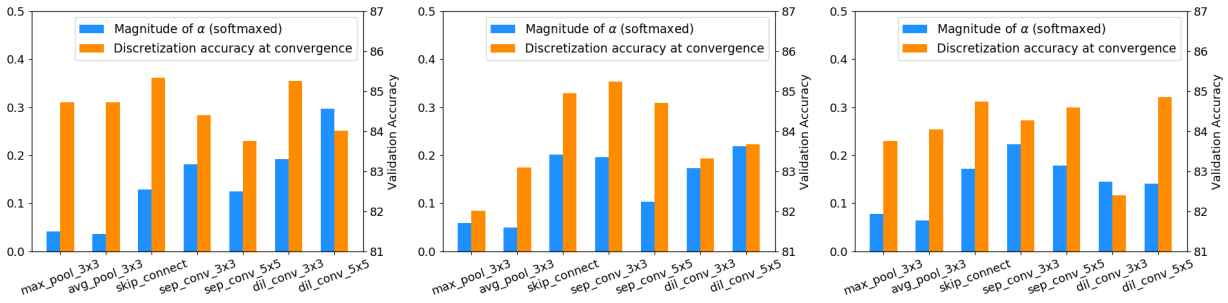
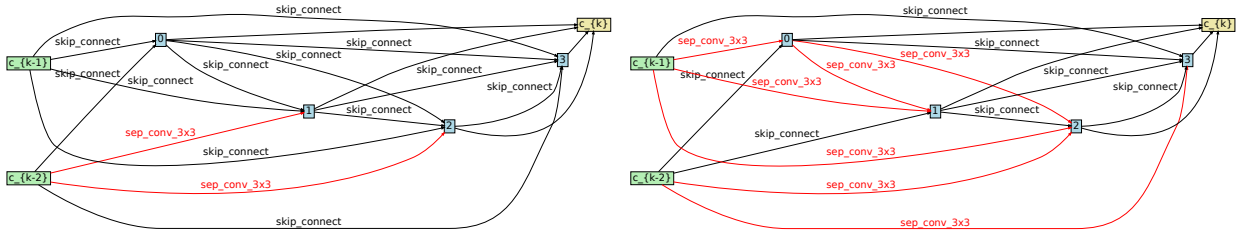


Figure 2.1: α vs discretization accuracy at convergence of all operations on 3 randomly selected edges from a pretrained DARTS supernet (one subplot per edge). The magnitude of α for each operation does not necessarily agree with its relative discretization accuracy at convergence.



(a) Magnitude (b) Strength

Figure 2.2: Operation strength on each edge of S2 *(skip_connect, sep_conv_3x3)*. (a). Operations associated with the largest α. (b). Operations that result in the highest discretization validation accuracy at convergence. Parameterized operations are marked red.

Following DARTS, existing differentiable NAS methods use the value of architecture parameters α to select the final architecture from the supernet, with the implicit assumption that α represents the strength of the underlying operations. In this section, we study the validity of this assumption in detail.

Consider one edge on a pretrained supernet; the strength of an operation on the edge can be naturally defined as the supernet accuracy after we discretize to this operation and

fine-tune the remaining network until it converges again; we refer to this as "discretization accuracy at convergence" for short. The operation that achieves the best discretization accuracy at convergence can be considered as the best operation for the given edge. Figure 2.1 shows the comparison of α (blue) and operation strength (orange) of randomly select edges on DARTS supernet. As we can see, the magnitude of α for each operation does not necessarily agree with their relative strength measured by discretization accuracy at convergence. Moreover, operations assigned with small αs are sometimes strong ones that lead to high discretization accuracy at convergence. To further verify the mismatch, we investigate the operation strength on search space S2, where DARTS fails dramatically due to excessive skip connections [ZES20]. S2 is a variant of DARTS search space that only contains two operations per edge (*skip_connect, sep_conv_3x3*). Figure 2.2 shows the selected operations based on α (left) and operation strength (right) on all edges on S2. From Figure 2.2a, we can see that $\alpha_{skip\_connect} > \alpha_{sep\_conv\_3x3}$ on 12 of 14 edges. Consequently, the derived child architecture will lack representation ability and perform poorly due to too many skip connections. However, as shown in Figure 2.2b, the supernet benefits more from discretizing to *sep_conv_3x3* than *skip_connect* on half of the edges.

There are several reason why α fails to capture the operation strength. Firstly, consider the second order approximation of the validation loss of a pretrained supernet:

$$\mathcal{L}_{val}(\hat{\alpha}, w) \approx \mathcal{L}_{val}(\alpha, w) + \alpha^T \nabla_\alpha \mathcal{L}_{val}(\alpha, w) + \frac{1}{2}\alpha^T \nabla^2_{\alpha^2} \mathcal{L}_{val}(\alpha, w)\alpha \tag{2.2}$$

$$= \mathcal{L}_{val}(\alpha, w) + \frac{1}{2}\alpha^T \nabla^2_{\alpha^2} \mathcal{L}_{val}(\alpha, w)\alpha \quad (\nabla_\alpha \mathcal{L}_{val}(\alpha, w) = 0 \text{ at convergence}) \tag{2.3}$$

Where $\hat{\alpha}$ is the perturbed architecture parameters and α is the current instance. We can see that the influence of α on the supernet's validation loss depends not only on α itself but also the Hessian matrix $\nabla^2_{\alpha^2} \mathcal{L}_{val}(\alpha, w)$. In magnitude-based architecture selection, the Hessian term is completely ignored, which corresponds to the case when $\nabla^2_{\alpha^2} \mathcal{L}_{val}(\alpha, w) = sI$. Secondly, architecture parameters are interdependent: select and discretizing one edge to an operation modifies the supernet, thereby affecting subsequent selection.

Table 2.1: Test accuracy before and after layer (edge) shuffling on CIFAR-10. For ResNet and VGG, we randomly swap two layers in each stage (defined as successive layers between two downsampling blocks. For DARTS supernet, we randomly swap two edges in every cell.

	VGG	**ResNet**	**DARTS**
Before	92.69	93.86	88.44
After	9.83 ± 0.33	83.2015 ± 2.03	81.09 ± 1.87

2.3.2 A case study: skip connection

Several works point out that DARTS tends to assign large α to skip connections, resulting in shallow architectures with poor generability [ZES20, LZS19, BHX19]. This "skip connection domination" issue is generally attributed to the failure of DARTS' supernet optimization. In contrast, we draw inspiration from research on ResNet [HZR16] and show that this phenomenon by itself is a reasonable outcome while DARTS refines its estimation of the optimal feature map, rendering α_{skip} ineffective in the architecture selection.

In vanilla networks (e.g., VGG), each layer computes a new level of feature map from the output feature map of the predecessor layer; thus, reordering layers at test time would dramatically hurt the performance [VWB16]. Unlike vanilla networks, [GSS17] and [VWB16] discover that successive layers in ResNet with compatible channel sizes are in fact estimating the same optimal feature map so that the outputs of these layers stay relatively close to each other at convergence; As a result, ResNet's test accuracy remains robust under layer reordering. [GSS17] refers to this unique way of feature map estimation in ResNet as the "unrolled estimation."

DARTS' supernet resembles ResNet, rather than vanilla networks like VGG, in both appearance and behavior. Appearance-wise, within a cell of DARTS' supernet, edges with skip connection are in direct correspondence with the successive residual layers in ResNet. Behavior-wise, DARTS' supernet also exhibits a high degree of robustness under edge shuffling. As shown in Table 2.1, randomly reordering edges on a pretrained DARTS' supernet

at test time also has little effect on its performance. This evidence indicates that DARTS performs unrolled estimation like ResNet as well, i.e., edges within a cell share the same optimal feature map that they try to estimate. In the following proposition, we apply this finding and provide the optimal solution of α in the sense of minimizing the variance of feature map estimation.

Proposition 1. [1] *Without loss of generality, consider one cell from a simplified search space consists of two operations: (skip, conv). Let m^* denotes the optimal feature map, which is shared across all edges according to the unrolled estimation view [GSS17]. Let $o_e(x_e)$ be the output of convolution operation, and let x_e be the skip connection (i.e., the input feature map of edge e). Assume m^*, $o_e(x_e)$ and x_e are normalized to the same scale. The current estimation of m^* can then be written as:*

$$\overline{m}_e(x_e) = \frac{\exp(\alpha_{conv})}{\exp(\alpha_{conv}) + \exp(\alpha_{skip})} o_e(x_e) + \frac{\exp(\alpha_{skip})}{\exp(\alpha_{conv}) + \exp(\alpha_{skip})} x_e, \tag{2.4}$$

where α_{conv} and α_{skip} are the architecture parameters defined in DARTS. The optimal α^*_{conv} and α^*_{skip} minimizing $var(\overline{m}_e(x_e) - m^*)$, the variance of the difference between the optimal feature map m^* and its current estimation $\overline{m}_e(x_e)$, are given by:

$$\alpha^*_{conv} \propto var(x_e - m^*) \tag{2.5}$$

$$\alpha^*_{skip} \propto var(o_e(x_e) - m^*). \tag{2.6}$$

We refer the reader to the original paper of DARTS-PT [WCC21] for a detailed proof. From eq. (2.5) and eq. (2.6), we can see that the relative magnitudes of α_{skip} and α_{conv} come down to which one of x_e or $o_e(x_e)$ is closer to m^* in variance:

- x_e (input of edge e) comes from the mixed output of the previous edge. Since the goal of every edge is to estimate m^* (unrolled estimation), x_e is also directly estimating m^*.

[1] *Proposition 1 unfolds the optimal α in principle and does not constraint the particular optimization method (i.e., bilevel, single-level, or blockwise update) to achieve it. Moreover, this proposition can be readily extended to various other search spaces since we can group all non-skip operations into a single $o_e(\cdot)$.*

- $o_e(x_e)$ is the output of a single convolution operation instead of the complete mixed output of edge e, so it will deviate from m^* even at convergence.

Therefore, in a well-optimized supernet, x_e will naturally be closer to m^* than $o_e(x_e)$, causing α_{skip} to be greater than α_{conv}.

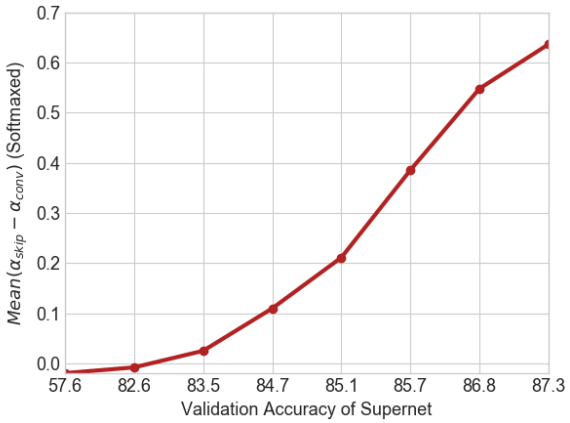


Figure 2.3: $mean(\alpha_{skip} - \alpha_{conv})$ (softmaxed) v.s. supernet's validation accuracy. The gap of $(\alpha_{skip} - \alpha_{conv})$ increases as supernet gets better.

Our analysis above indicates that the better the supernet, the larger the $(\alpha_{skip} - \alpha_{conv})$ gap (softmaxed) will become since x_e gets closer and closer to m^* as the supernet is optimized. This result is evidenced in Figure 2.3, where $mean(\alpha_{skip} - \alpha_{conv})$ continues to grow as the supernet gets better. In this case, although $\alpha_{skip} > \alpha_{conv}$ is reasonable by itself, it becomes an inductive bias to NAS if we were to select the final architecture based on α.

CHAPTER 3

Improving the Effectiveness and Robustness of Differentiable NAS

In the previous chapter, we identify that the poor generalization of DARTS, including skip connection domination, is caused by the failure of magnitude-based architecture selection. Inspired by this analysis, we introduce two lines of methods to solve this problem: 1) A perturbation-based architecture selection that bypasses architecture parameter α and measures the operation strength in an adversarial fashion (Section 3.1), and 2) improve the alignment between α and operation strength via Hessian Regularization.

3.1 Perturbation-based architecture selection

Instead of relying on the α value to select the best operation, we propose to directly evaluate operation strength in terms of its contribution to the supernet's performance. The operation selection criterion is laid out in section 3.1.1. In section 3.1.2, we describe the entire architecture selection process.

3.1.1 Evaluating the strength of each operation

In section 2.3.1, we define the strength of each operation on a given edge as how much it contributes to the performance of the supernet, measured by discretization accuracy. To avoid inaccurate evaluation due to large disturbance of the supernet during discretization, we fine-tune the remaining supernet until it converges again, and then compute its valida-

tion accuracy (discretization accuracy at convergence). The fine-tuning process needs to be carried out for evaluating each operation on an edge, leading to substantial computation costs.

To alleviate the computational overhead, we consider a more practical measure of operation strength: for each operation on a given edge, we mask it out while keeping all other operations, and re-evaluate the supernet. The one that results in the largest drop in the supernet's validation accuracy will be considered as the most important operation on that edge. This alternative criterion incurs much less perturbation to the supernet than discretization since it only deletes one operation from the supernet at a time. As a result, the supernet's validation accuracy after deletion stays close to the unmodified supernet, and thus it alleviates the requirement of tuning the remaining supernet to convergence. Therefore, we implement this measurement for the operation selection in this work.

Algorithm 1: Perturbation-based Architecture Selection

Input: A pretrained supernet S, Set of edges $\mathcal{E}$ from S, Set of nodes $\mathcal{N}$ from S

Result: Set of selected operations $\{o_e^*\}_{e \in \mathcal{E}}$

while $|\mathcal{E}| > 0$ **do**

  randomly select an edge $e \in \mathcal{E}$ (and remove it from $\mathcal{E}$);

  forall *operation o on edge e* **do**

    evaluate the validation accuracy of S when o is removed ($ACC_{\backslash o}$);

  end

  select the best operation for e: $o_e^* \leftarrow \arg\min_o ACC_{\backslash o}$;

  discretize edge e to o_e^* and tune the remaining supernet for a few epochs;

end

3.1.2 The complete architecture selection process

Our method operates directly on top of DARTS' pretrained supernet. Given a supernet, we randomly iterate over all of its edges. We evaluate each operation on an edge, and select the

best one to be discretized based on the measurement described in section 3.1.1. After that, we tune the supernet for a few epochs to recover the accuracy lost during discretization. The above steps are repeated until all edges are decided. Algorithm 1 summarizes the operation selection process. The cell topology is decided in a similar fashion. This simple method is termed "perturbation-based architecture selection (PT)" in the following sections.

3.1.3 Experimental Evaluation

In this section, we demonstrate that the perturbation-based architecture selection method is able to consistently find better architectures than those selected based on the values of α. The evaluation is based on the search space of DARTS and NAS-Bench-201 [DY20], and we show that the perturbation-based architecture selection method can be applied to several variants of DARTS.

3.1.3.1 Results on DARTS' CNN search space

We keep all the search and retrain settings identical to DARTS since our method only modifies the architecture selection part. After the search phase, we perform perturbation-based architecture selection following Algorithm 1 on the pretrained supernet. We tune the supernet for 5 epochs between two selections as it is enough for the supernet to recover from the drop of accuracy after discretization. We run the search and architecture selection phase with four random seeds and report both the best and average test errors of the obtained architectures.

As shown in Table 3.1, the proposed method (DARTS+PT) improves DARTS' test error from 3.00% to 2.61%, with manageable search cost (0.8 GPU days). Note that by only changing the architecture selection method, DARTS performs significantly better than many other differentiable NAS methods that enjoy carefully designed optimization process of the supernet, such as GDAS [DY19] and SNAS [XZL19]. This empirical result suggests that

architecture selection is crucial to DARTS: with the proper selection algorithm, DARTS remains a very competitive method.

Our method is also able to improve the performance of other variants of DARTS. To show this, we evaluate our method on SDARTS(rs) and SGAS [CH20, LQD20]. SDARTS(rs) is a variant of DARTS that regularizes the search phase by applying Gaussian perturbation to α. Unlike DARTS and SDARTS, SGAS performs progressive search space shrinking. Concretely, SGAS progressively discretizes its edges with the order from most to least important, based on a novel edge importance score. For a fair comparison, we keep its unique search space shrinking process unmodified and only replace its magnitude-based operation selection with ours. As we can see from Table 3.1, our method consistently achieves better average test errors than its magnitude-based counterpart. Concretely, the proposed method improves SDARTS' test error from 2.67% to 2.54% and SGAS' test error from 2.66% to 2.56%. Moreover, the best architecture discovered in our experiments achieves a test error of 2.44%, ranked top among other NAS methods.

3.1.3.2 Performance on NAS-Bench-201 search space

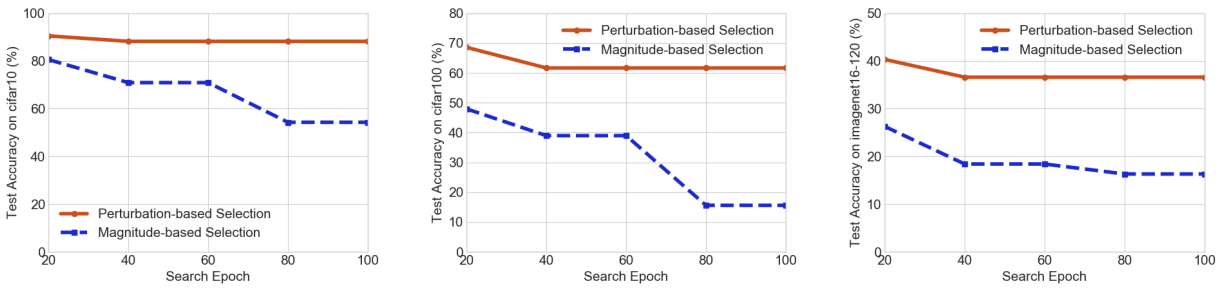


Figure 3.1: Trajectory of test accuracy on space NAS-Bench-201 and three datasets (Left: cifar10, Middle: cifar100, Right: Imagenet16-120). The test accuracy of our method is plotted by taking the snapshots of DARTS' supernet at corresponding epochs and run our selection method on top of it.

To further verify the effectiveness of the proposed perturbation-based architecture se-

Table 3.1: Test error of architectures discovered by perturbation-based selection on DARTS Space and CIFAR-10.

Architecture	**Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC [HLM17]	3.46	25.6	-	manual
NASNet-A [ZVS18]	2.65	3.3	2000	RL
AmoebaNet-A [RAH19]	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-B [RAH19]	2.55 ± 0.05	2.8	3150	evolution
PNAS [LZN18]*	3.41 ± 0.09	3.2	225	SMBO
ENAS [PGZ18]	2.89	4.6	0.5	RL
NAONet [LTQ18]	3.53	3.1	0.4	NAO
SNAS (moderate) [XZL19]	2.85 ± 0.02	2.8	1.5	gradient
GDAS [DY19]	2.93	3.4	0.3	gradient
BayesNAS [ZYW19]	2.81 ± 0.04	3.4	0.2	gradient
ProxylessNAS [CZH19]†	2.08	5.7	4.0	gradient
NASP [YXT20]	2.83 ± 0.09	3.3	0.1	gradient
P-DARTS [CXW19]	2.50	3.4	0.3	gradient
PC-DARTS [XXZ20]	2.57 ± 0.07	3.6	0.1	gradient
R-DARTS (L2) [ZES20]	2.95 ± 0.21	-	1.6	gradient
DARTS [LSY19]	3.00 ± 0.14	3.3	0.4	gradient
SDARTS-RS [CH20]	2.67 ± 0.03	3.4	0.4	gradient
SGAS (Cri 1. avg) [LQD20]	2.66 ± 0.24	3.7	0.25	gradient
DARTS+PT (avg)*	2.61 ± 0.08	3.0	0.8‡	gradient
DARTS+PT (best)	2.48	3.3	0.8‡	gradient
SDARTS-RS+PT (avg)*	2.54 ± 0.10	3.3	0.8‡	gradient
SDARTS-RS+PT (best)	2.44	3.2	0.8‡	gradient
SGAS+PT (Crit.1 avg)*	2.56 ± 0.10	3.9	0.29‡	gradient
SGAS+PT (Crit.1 best)	2.46	3.9	0.29‡	gradient

† Obtained on a different space with PyramidNet [HKK17] as the backbone.

‡ Recorded on a single GTX 1080Ti GPU.

* Obtained by running the search and retrain phase under four different seeds and computing the average test error of the derived architectures.

lection, we conduct experiments on NAS-Bench-201. NAS-Bench-201 provides a unified cell-based search space similar to DARTS. Every architecture in the search space is trained

under the same protocol on three datasets (cifar10, cifar100, and imagenet16-120), and their performance can be obtained by querying the database. As in section 3.1.3.1, we take the pretrained supernet from DARTS and apply our method on top of it. All other settings are kept unmodified. Figure 3.1 shows the performance trajectory of DARTS+PT compared with DARTS. While the architectures found by magnitude-based selection degenerates over time, the perturbation-based method is able to extract better architectures from the same underlying supernets stably. The result implies that the DARTS' degenerated performance comes from the failure of magnitude based architecture selection.

3.2 Aligning architecture parameter with operation strength via distribution learning

Recall that in Section 2.3, we show that the misalignment between α and operation strength is caused by the existence of the Hessian term $\nabla^2_{\alpha^2}\mathcal{L}_{val}(\alpha, w)$ (eq. 2.2). When this hessian is ill-conditioned, the magnitude of *alpha* deviates from the operation strength. Therefore, another way to improve the robustness of DARTS is to regularize this Hessian. In this section, we propose a principled method by formulating differentiable NAS as a distribution learning problem.

3.2.0.1 Neural Architecture Search as distribution learning

Bilevel-Optimization with Simplex Constraints In DARTS, the uncontrained architecture parameters are mapped to the operation mixing weight via softmax function, allowing it to lie in the probability simplex. To motivate our method, we first generalize the bilevel formulation of DARTS by using θ to simplex-constrained represent the operation mixing weight:

$$\min_{\theta} \quad \mathcal{L}_{val}(w^*, \theta) \quad \text{s.t.} \quad w^* = \arg\min_{w} \ \mathcal{L}_{train}(w, \theta), \qquad \sum_{o=1}^{|\mathcal{O}|} \theta_o^{(i,j)} = 1, \ \forall\ (i,j),\ i < j, \tag{3.1}$$

where the simplex constraint $\sum_{o=1}^{|\mathcal{O}|} \theta_o^{(i,j)} = 1$ can be either solved explicitly via Lagrangian function [LKB20], or eliminated by substitution method (e.g., $\theta = Softmax(\alpha), \alpha \in \mathcal{R}^{|\mathcal{O}|\times|E|}$) [LSY19].

Learning a Distribution over Operation Mixing Weight Previous differentiable architecture search methods view the operation mixing weight θ as learnable parameters that can be directly optimized [LSY19, XXZ20, LKB20]. This has been shown to cause θ to overfit the validation set and thus induce large generalization error [BKZ18, ZSH20, CH20]. We recognize that this treatment is equivalent to performing point estimation (e.g., MLE/MAP) of θ in probabilistic view, which is inherently prone to overfitting [Bis16, GCS04]. Based on this insight, we formulate the differentiable architecture search as a distribution learning problem. The operation mixing weight θ is treated as random variables sampled from a learnable distribution. Formally, let $q(\theta|\beta)$ denote the distribution of θ parameterized by β. The bi-level objective is then given by:

$$\min_{\beta} E_{q(\theta|\beta)}\big[\mathcal{L}_{val}(w^*, \theta)\big] + \lambda d(\beta, \hat{\beta}) \;\; \text{s.t.} \;\; w^* = \arg\min_{w} \; \mathcal{L}_{train}(w, \theta). \tag{3.2}$$

where $d(\cdot, \cdot)$ is a distance function. Since θ lies on the probability simplex, we select Dirichlet distribution to model its behavior, i.e., $q(\theta|\beta) \sim Dir(\beta)$, where β represents the Dirichlet concentration parameter. Dirichlet distribution is a widely used distribution over the probability simplex [JLP19, Dav03, LHZ20, KNZ19], and it enjoys nice properties that enables gradient-based training [Mar18].

3.2.1 The implicit Regularization on Hessian

The objective in (3.2) can be viewed as a Lagrangian function of the following constraint objective:

$$\min_{\beta} E_{q(\theta|\beta)}\big[\mathcal{L}_{val}(w^*, \theta)\big] \;\; \text{s.t.} \;\; w^* = \arg\min_{w} \; \mathcal{L}_{train}(w, \theta) \;,\; d(\beta, \hat{\beta}) \leq \delta, \tag{3.3}$$

Here we derive an approximated lower bound based on (3.3), which demonstrates that our method implicitly controls the conditional number of the Hessian matrix $\nabla^2_\theta \tilde{\mathcal{L}}_{val}(w, \theta)$.

Proposition 2. *Let $d(\beta, \hat{\beta}) = \|\beta - \hat{\beta}\|_2 \leq \delta$ and $\hat{\beta} = 1$ in the bi-level formulation (3.3). Let μ denote the mean under the Laplacian approximation of Dirichlet. If $\nabla^2_\mu \tilde{\mathcal{L}}_{val}(w^*, \mu)$ is Positive Semi-definite, the upper-level objective can be approximated bounded by:*

$$E_{q(\theta|\beta)}(\mathcal{L}_{val}(w, \theta)) \gtrsim \tilde{\mathcal{L}}_{val}(w^*, \mu) + \frac{1}{2}(\frac{1}{1+\delta}(1 - \frac{2}{|\mathcal{O}|}) + \frac{1}{|\mathcal{O}|}\frac{1}{1+\delta})tr\big(\nabla^2_\mu \tilde{\mathcal{L}}_{val}(w^*, \mu)\big) \quad (3.4)$$

with:

$$\tilde{\mathcal{L}}_{val}(w^*, \mu) = \mathcal{L}_{val}(w^*, Softmax(\mu)), \quad \mu_o = \log \beta_o - \frac{1}{|\mathcal{O}|}\sum_{o'} \log \beta_{o'}, \quad o = 1, \ldots, |\mathcal{O}|.$$

This proposition is driven by the Laplacian approximation to the Dirichlet distribution [Mac98, Aka17]. The lower bound (3.4) indicates that minimizing the expected validation loss controls the trace norm of the Hessian matrix. We refer the reader to the DrNAS paper [CWC21] for the detailed proof.

3.2.1.1 Experimental Results

Empirically, DrNAS significantly outperforms DARTS and comparable methods on CIFAR-10 (Table 3.2) and ImageNet (Table 3.3).

Table 3.2: Test error of the architectures discovered by DrNAS on DARTS Space and CIFAR-10.

Architecture	**Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC [HLM17]⋆	3.46	25.6	-	manual
NASNet-A [ZVS18]	2.65	3.3	2000	RL
AmoebaNet-A [RAH19]	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-B [RAH19]	2.55 ± 0.05	2.8	3150	evolution
PNAS [LZN18]⋆	3.41 ± 0.09	3.2	225	SMBO
ENAS [PGZ18]	2.89	4.6	0.5	RL
DARTS (1st) [LSY19]	3.00 ± 0.14	3.3	0.4	gradient
DARTS (2nd) [LSY19]	2.76 ± 0.09	3.3	1.0	gradient
SNAS (moderate) [XZL19]	2.85 ± 0.02	2.8	1.5	gradient
GDAS [DY19]	2.93	3.4	0.3	gradient
BayesNAS [ZYW19]	2.81 ± 0.04	3.4	0.2	gradient
ProxylessNAS [CZH19]†	2.08	5.7	4.0	gradient
PARSEC [CGF19]	2.81 ± 0.03	3.7	1	gradient
P-DARTS [CXW19]	2.50	3.4	0.3	gradient
PC-DARTS [XXZ20]	2.57 ± 0.07	3.6	0.1	gradient
SDARTS-ADV [CH20]	2.61 ± 0.02	3.3	1.3	gradient
GAEA + PC-DARTS [LKB20]	2.50 ± 0.06	3.7	0.1	gradient
DrNAS	2.54 ± 0.03	4.0	0.4‡	gradient
DrNAS + progressive learning	2.46 ± 0.03	4.1	0.6‡	gradient

⋆ Obtained without cutout augmentation.

† Obtained on a different space with PyramidNet [HKK17] as the backbone.

‡ Recorded on a single GTX 1080Ti GPU.

Table 3.3: Test error of architectures discovered by DrNAS on DARTS Space and ImageNet under mobile setting.

Architecture	**Test Error(%)**		**Params**	**Search Cost**	**Search**
	top-1	top-5	**(M)**	**(GPU days)**	**Method**
Inception-v1 [SLJ15]	30.1	10.1	6.6	-	manual
MobileNet [HZC17]	29.4	10.5	4.2	-	manual
ShuffleNet 2× (v1) [ZZL18]	26.4	10.2	~ 5	-	manual
ShuffleNet 2× (v2) [MZZ18]	25.1	-	~ 5	-	manual
NASNet-A [ZVS18]	26.0	8.4	5.3	2000	RL
AmoebaNet-C [RAH19]	24.3	7.6	6.4	3150	evolution
PNAS [LZN18]	25.8	8.1	5.1	225	SMBO
MnasNet-92 [TCP19]	25.2	8.0	4.4	-	RL
DARTS (2nd) [LSY19]	26.7	8.7	4.7	1.0	gradient
SNAS (mild) [XZL19]	27.3	9.2	4.3	1.5	gradient
GDAS [DY19]	26.0	8.5	5.3	0.3	gradient
BayesNAS [ZYW19]	26.5	8.9	3.9	0.2	gradient
DSNAS [HXZ20]†	25.7	8.1	-	-	gradient
ProxylessNAS (GPU) [CZH19]†	24.9	7.5	7.1	8.3	gradient
PARSEC [CGF19]	26.0	8.4	5.6	1	gradient
P-DARTS (CIFAR-10) [CXW19]	24.4	7.4	4.9	0.3	gradient
P-DARTS (CIFAR-100) [CXW19]	24.7	7.5	5.1	0.3	gradient
PC-DARTS (CIFAR-10) [XXZ20]	25.1	7.8	5.3	0.1	gradient
PC-DARTS (ImageNet) [XXZ20]†	24.2	7.3	5.3	3.8	gradient
GAEA + PC-DARTS [LKB20]†	24.0	7.3	5.6	3.8	gradient
DrNAS†	24.2	7.3	5.2	3.9	gradient
DrNAS + progressive learning†	23.7	7.1	5.7	4.6	gradient

† The architecture is searched on ImageNet, otherwise it is searched on CIFAR-10 or CIFAR-100.

CHAPTER 4

Beyond Differentiable NAS - Predictor-based Architecture Search

While previous sections mainly focus on analyzing and improving differentiable NAS methods, in this section we introduce an alternative paradigm of NAS called predictor-based architecture search. Unlike Differentiable NAS, Predictor-based methods do not rely on building weight-sharing supernets to estimate the performance of child architectures. Instead, they build a surrogate predictor to infer the performance of child architectures; The predictor can be trained using a tiny selected subset of all architectures (usually in hundreds of architectures), and subsequently deployed for architecture evaluation the efficiently. This way, predictor-based methods are free from the inductive biases of weight-sharing differentiable NAS. However, search efficiency becomes a major bottleneck for predictor-based NAS: To label the training pool for the surrogate predictor, we now need to train and evaluate hundreds of architecture fully from scratch. To improve the search efficiency, previous works mainly focus on developing more sample-efficient predictors. We tackle this challenge from different perspective: pause and resume the training of poor architectures to save budget. The resulting framework, RANK-NOSH, reduces the search budget of the best predictor-based algorithm by 5 folds, while achieving competitive performance.

4.1 Predictor-based NAS

4.1.1 Framework

Starting from a pool of randomly selected architectures, previous methods iteratively conduct the following steps: 1) train and evaluate all the architectures in the pool fully; 2) fit a surrogate performance predictor; 3) use the predictor to propose new architectures and add them to the pool for the next round [DCA20, WNS19, YZA20]. Compared with previous RL and evolution-based NAS methods, using a performance predictor can reduce the number of networks evaluated from scratch. However, training all the architectures in the candidate pool fully is still extremely computationally expensive. Most complementary advances alone this line focus on developing better predictors that require a smaller training pool [DCA20, WNS19, YZA20], but the potential to further cut down the search cost by reducing the training length of individual architectures in the pool has not drawn much attention.

4.1.2 Improving the efficiency of predictor-based NAS with early termination and learning to rank

Inspired by successive halving [JT16], our key idea is that the learning process of poor architectures can be terminated early to avoid wasting budgets. However, it is non-trivial to integrate successive halving to predictor-based NAS formulations. Firstly, predictor-based algorithms iteratively add new architectures to the candidate pool [DCA20, WNS19, YZA20], whereas regular successive halving only removes underperforming candidates from the initial pool. Secondly, with successive halving, architectures in the pool will be trained for different number of epochs, so their validation accuracy are not directly comparable in a semantically meaningful way. Standard regression-based predictor fitting, which requires the exact validation accuracy for each architecture when fully trained, will be problematic in this setting.

To tackle those challenges in a unified way, we introduce RANK-NOSH, an efficient predictor-based framework with significantly improved search efficiency. RANK-NOSH consists of two parts. The first part is NOn-Uniform Successive Halving (NOSH), which describes a multi-level scheduling algorithm that allows adding new candidates and resuming terminated training process. It is non-uniform in the sense that NOSH maintains a pyramid-like candidate pool of architectures trained for various epochs without discarding any candidates. For the second part, we construct architecture pairs and use a pairwise ranking loss to train the performance predictor. The predictor is essentially a ranking network and can efficiently distill useful information from our candidate pool consisting of architectures trained for different epochs. Moreover, the proposed framework naturally integrates recently developed proxies that measure architecture performance without training [AMD21, CGW21, MTS20], which allows more architectures to be included in the candidate pool at no cost.

4.1.2.1 Experimental Results

On DARTS Space, RANK-NOSH achieves an average test error of 2.53% on CIFAR-10 with over 5x search budget reduction than previous SOTA predictor-based NAS method arch2vec, which obtains an average test error of 2.56%.

CHAPTER 5

Conclusion

This paper provides both theoretical and empirically analysis of differentiable architecture search, which leads to advancement in both algorithmic and paradigmatic design. To push the limit of Neural Architecture Search, it requires not only understanding and advancing of existing algorithms, but also out-of-box thinking about new settings. For example, all existing NAS paradigm assumes and traverses a predefined search space, which is essential to the search outcome but requires a lot of prior knowledge to construct. Automated search space design remains a challenging future direction with huge potential in fully automatizing NAS pipeline.

REFERENCES

[Aka17] Charles Sutton Akash Srivastava. "Autoencoding Variational Inference For Topic Models." In *International Conference on Learning Representations*, 2017.

[AMD21] Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. "Zero-cost proxies for lightweight NAS." In *ICLR*, 2021.

[BHX19] Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. "Stabilizing DARTS with Amended Gradient Estimation on Architectural Parameters.", 2019.

[Bis16] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2016.

[BKZ18] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. "Understanding and Simplifying One-Shot Architecture Search." In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 550–559, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[BLR18] Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. "SMASH: One-Shot Model Architecture Search through HyperNetworks." In *International Conference on Learning Representations*, 2018.

[CGF19] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. "Probabilistic Neural Architecture Search." *arXiv: 1902.05116*, 2019.

[CGW21] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. "Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective." In *International Conference on Learning Representations*, 2021.

[CH20] Xiangning Chen and Cho-Jui Hsieh. "Stabilizing Differentiable Architecture Search via Perturbation-based Regularization." In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1554–1565. PMLR, 13–18 Jul 2020.

[CWC21] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. "Dr{NAS}: Dirichlet Neural Architecture Search." In *International Conference on Learning Representations*, 2021.

[CXW19] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1294–1303, 2019.

[CZH19] Han Cai, Ligeng Zhu, and Song Han. "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware." In *International Conference on Learning Representations*, 2019.

[Dav03] Michael I. Jordan David M. Blei, Andrew Y. Ng. "Latent Dirichlet Allocation." *The Journal of Machine Learning Research*, Mar 2003.

[DCA20] Lukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. "BRP-NAS: Prediction-based NAS using GCNs." In *NeurIPS*, 2020.

[DY19] Xuanyi Dong and Yi Yang. "Searching for A Robust Neural Architecture in Four GPU Hours." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

[DY20] Xuanyi Dong and Yi Yang. "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search." In *International Conference on Learning Representations (ICLR)*, 2020.

[GCS04] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

[GSS17] Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. "Highway and Residual Networks learn Unrolled Iterative Estimation." In *International Conference on Learning Representations (ICLR)*, 2017.

[GZM19] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. "Single Path One-Shot Neural Architecture Search with Uniform Sampling.", 2019.

[HKK17] Dongyoon Han, Jiwhan Kim, and Junmo Kim. "Deep Pyramidal Residual Networks." In *CVPR*, 2017.

[HLM17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[HXZ20] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. "DSNAS: Direct Neural Architecture Search without Parameter Retraining." In *CVPR*, 2020.

[HZC17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." *arXiv: 1704.04861*, 2017.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 06 2016.

[JLP19] Weonyoung Joo, Wonsung Lee, Sungrae Park, , and Il-Chul Moon. "Dirichlet Variational Autoencoder.", 2019.

[JT16] Kevin Jamieson and Ameet Talwalkar. "Non-stochastic best arm identification and hyperparameter optimization." In *AISTATS*, 2016.

[KNZ19] Samuel Kessler, Vu Nguyen, Stefan Zohren, and Stephen Roberts. "Hierarchical Indian Buffet Neural Networks for Bayesian Continual Learning.", 2019.

[LHZ20] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. "A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning," in International Conference on Learning Representations." In *ICLR*, 2020.

[LKB20] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. "Geometry-Aware Gradient Algorithms for Neural Architecture Search." *arXiv: 2004.07802*, 2020.

[LQD20] Guohao Li, Guocheng Qian, Itzel C. Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. "SGAS: Sequential Greedy Architecture Search." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1620–1630, 2020.

[LSV17] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. "Hierarchical Representations for Efficient Architecture Search.", 2017.

[LSY19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "DARTS: Differentiable Architecture Search." In *International Conference on Learning Representations*, 2019.

[LT19] Liam Li and Ameet Talwalkar. "Random Search and Reproducibility for Neural Architecture Search.", 2019.

[LTQ18] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. "Neural Architecture Optimization." In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pp. 7816–7827. Curran Associates, Inc., 2018.

[LZN18] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. "Progressive Neural Architecture Search." *Lecture Notes in Computer Science*, p. 19–35, 2018.

[LZS19] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. "DARTS+: Improved Differentiable Architecture Search with Early Stopping.", 2019.

[Mac98] David J. C. MacKay. "Choice of Basis for Laplace Approximation." *Machine Language*, October 1998.

[Mar18] Fritz Obermeyer Martin Jankowiak. "Pathwise Derivatives Beyond the Reparameterization Trick." In *International Conference on Machine Learning*, 2018.

[MTS20] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. "Neural Architecture Search without Training.", 2020.

[MZZ18] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design." In *ECCV*, 2018.

[PGZ18] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. "Efficient Neural Architecture Search via Parameters Sharing." In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[RAH19] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. "Regularized Evolution for Image Classifier Architecture Search." *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**:4780–4789, Jul 2019.

[RMS17] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. "Large-Scale Evolution of Image Classifiers." In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 2902–2911. JMLR.org, 2017.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." In *CVPR*, 2015.

[SM02] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies." *Evolutionary Computation*, **10**(2):99–127, 2002.

[TCP19] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. "MnasNet: Platform-Aware Neural Architecture Search for Mobile." In *CVPR*, 2019.

[VWB16] Andreas Veit, Michael Wilber, and Serge Belongie. "Residual Networks Behave Like Ensembles of Relatively Shallow Networks." In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pp. 550—-558. Curran Associates, Inc., 2016.

[WCC21] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. "Rethinking Architecture Selection in Differentiable NAS." In *International Conference on Learning Representations (ICLR)*, 2021.

[WNS19] Colin White, Willie Neiswanger, and Yash Savani. "BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search." *arXiv preprint arXiv:1910.11858*, 2019.

[XXZ20] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. "PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search." In *International Conference on Learning Representations*, 2020.

[XZL19] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. "SNAS: stochastic neural architecture search." In *International Conference on Learning Representations*, 2019.

[YXT20] Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. "Efficient Neural Architecture Search via Proximal Iterations." In *AAAI*, 2020.

[YZA20] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. "Does Unsupervised Architecture Representation Learning Help Neural Architecture Search?" In *NeurIPS*, 2020.

[ZES20] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. "Understanding and Robustifying Differentiable Architecture Search." In *International Conference on Learning Representations*, 2020.

[ZL17] Barret Zoph and Quoc V. Le. "Neural Architecture Search with Reinforcement Learning." In *International Conference on Learning Representations (ICLR)*, 2017.

[ZSH20] Arber Zela, Julien Siems, and Frank Hutter. "NAS-BENCH-1SHOT1: BENCHMARKING AND DISSECTING ONE-SHOT NEURAL ARCHITECTURE SEARCH." In *International Conference on Learning Representations*, 2020.

[ZVS18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. "Learning Transferable Architectures for Scalable Image Recognition." *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[ZYW18] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. "Practical Block-Wise Neural Network Architecture Generation." In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.

[ZYW19] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. "BayesNAS: A Bayesian Approach for Neural Architecture Search." In *ICML*, pp. 7603–7613, 2019.

[ZZL18] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices." In *CVPR*, 2018.