PertNAS: Architectural Perturbations for Memory-Efficient Neural Architecture Search

Afzal Ahmad, Zhiyao Xie, Wei Zhang
The Hong Kong University of Science and Technology
afzal.ahmad@connect.ust.hk, {eezhiyao, eeweiz}@ust.hk

***Abstract*—Differentiable Neural Architecture Search (NAS) relies on aggressive weight-sharing to reduce its search cost. This leads to GPU-memory bottlenecks that hamper the algorithm's scalability. To resolve these bottlenecks, we propose a perturbations-based evolutionary approach that significantly reduces the memory cost while largely maintaining the efficiency benefits of weight-sharing. Our approach makes minute changes to compact neural architectures and measures their impact on performance. In this way, it extracts high-quality motifs from the search space. We utilize these perturbations to perform NAS in compact models evolving over time to traverse the search space. Our method disentangles GPU-memory consumption from search space size, offering exceptional scalability to large search spaces. Results show competitive accuracy on multiple benchmarks, including CIFAR10, ImageNet2012, and NASBench-301. Specifically, our approach improves accuracy on ImageNet and NASBench-301 by 0.3% and 0.87%, respectively. Furthermore, the memory consumption of search is reduced by roughly 80% against state-of-the-art weight-shared differentiable NAS works while achieving a search time of only 6 GPU hours.**

I. Introduction

Neural Architecture Search (NAS) is paving the way for efficient and automated architecture designs that offer better accuracy/FLOPs tradeoffs compared to handcrafted models [1]. Complex convolutional models consist of a number of control knobs – including architecture depth, operation choices, kernel sizes, and topologies – that govern their accuracy and computational/memory complexity. Tuning these control knobs by hand is a tedious process. NAS aims to automate this tuning process to achieve parameter-efficient models. However, the applicability of many NAS methods is limited by computation resources for two major reasons: 1) Model evaluation is costly [1]–[3]. Training all individual models in the search space is prohibitively time-consuming and resource-demanding. 2) Search spaces are limited by hardware resources. To reduce the exorbitant computational cost of NAS methods, experts often have to severely restrict the search space [4], [5], leading to serious inductive biases and sub-optimal solutions. These two reasons make NAS research fairly expensive and seriously limit the scalability of existing solutions.

Some NAS methods perform an exhaustive evaluation using search schemes like reinforcement learning (RL). They first train a large number of models sampled from the search space. Then the performance and architectures of these models are used to train a surrogate RL controller; usually an RNN, to further sample better architectures over time. Given the insurmountable number of model evaluations required to train the RL controller, RL-based methods require a huge amount of computation resources [2]. Take the MobileNetV3 [1] as an example. Although these models remain unbeatable in terms of their parameter efficiency (accuracy/FLOPs), their RL-based searching process requires thousands of GPUs/TPUs over several days to weeks to be searched. Hence, these methods cannot be replicated within reasonable cost when applied to new datasets or tasks.

To mitigate such high demand on computation resources, differentiable architecture search approaches [5]–[7] adopt aggressive weight-sharing [8] between models in the search space. It allows evaluation of the search space components in a single-shot fashion, offering search costs as low as only a few GPU hours. However, the search space scalability of these methods is seriously limited due to the single-shot evaluation, which requires training models with sizes proportional to the search space size. It leads to prohibitive GPU memory consumption, which becomes one major bottleneck in the hardware constraints [9], [10]. To ensure a good result from a limited search space, experts carefully design these search spaces using knowledge obtained from high-quality manual neural architecture designs [11], [12]. This inevitably adds significant inductive bias to the search process [4]. Furthermore, the effectiveness of the differentiable search methods has been questioned repeatedly [7], [13], whereby random search has outperformed differentiable search approaches [13]. Their weight-sharing technique [8] has been shown to suffer from a rank miscorrelation problem whereby weight-shared models are known to serve as poor surrogates for full model training [11], [14], [15].

In this work, we propose an evolutionary search scheme that utilizes architectural 'perturbations' as an evaluation metric to identify high-quality components of the search space. This search scheme evaluates and evolves compact models over time to explore the search space. It circumvents the severe memory limit faced by one-shot NAS methods. During searching, we apply small changes to neural architectures, called perturbations, and measure their impact on the model performance. This helps to identify high-quality motifs from search spaces. We demonstrate that perturbations offer more efficient, scalable, and verifiable operation strength evaluation over black-box differentiable approaches.

The contributions of this work are as follows:

1) We explore architectural perturbations as an operation strength evaluation metric for NAS. We are the first to explore perturbations as the basis to perform NAS without using over-parametrized models, addressing their memory bottlenecks.
2) We propose a perturbations-based evolutionary approach that relies only partially on weight-sharing and serves as a fast, accurate, and scalable search method. Our method achieves a search cost of only 6 GPU-hours with peak memory consumption of only 20% that of differentiable approaches. Furthermore, experimentation on ImageNet and NASBench-301 show an improvement of 0.3% and 0.87% in accuracy, respectively, against state-of-the-art in differentiable NAS.
3) The memory consumption of PertNAS does not depend on search space size. It is similar to training a single compact model sampled from the search space. In comparison, existing differentiable search works [5], [6] all require GPU memory that scales linearly with search space size. Disentangling the size from memory-cost paves the way for scaling up the search space, which proves to produce better models [1].

II. Preliminaries

A. Background

Search Spaces. Search spaces represent the control knobs that require tuning. Most one-shot NAS approaches, such as DARTS [5] and its variants [6], [7] employ a repeating cell-based search space. This alleviates the search cost to that of searching for cells instead

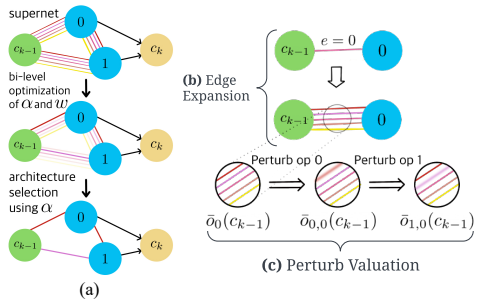

Fig. 1. (a) Differentiable architecture search [5] performs a bi-level optimization on a weight-shared supernet to jointly optimize architecture α and weights w, (b) Expansion of an edge involves placing all operations in $\mathcal{O}$ on that edge, (c) Perturb valuation of an operation strength is performed by masking that operation when measuring the output of mixed operation $\bar{o}_{m,e}$ (masked operations represented by blurred lines in inset).

of an entire neural network. A cell is represented by a directed acyclic graph (DAG) where nodes represent feature maps while edges represent operations. The DARTS cells utilize *two input* nodes, *four intermediate* nodes, and *one output* node. The two input nodes are the outputs of the two previous cells, while each successive intermediate node takes its input from all previous nodes. The feature map representing an intermediate node is calculated by applying the operations on each of its incoming edges to their respective input nodes. The output of the k-th cell, c_k, is obtained by concatenating the feature maps represented by the intermediate nodes along the channel dimension. Fig. 1 (a) top shows a cell with one input, one output, and two intermediate nodes – in green, yellow, and blue colors, respectively. DARTS and its variants employ two types of cells: normal cells with stride one operations and reduction cells with stride two operations to reduce the spatial resolution. Cells resulting from the search phase are replicated as part of a larger model for evaluation. For example, an evaluation model for CIFAR10 utilizes 18 normal and two reduction cells. Since reduction cells serve only to reduce spatial resolution, we utilize the same architecture for normal and reduction cells but with stride two operations in reduction cells.

The DARTS search space consists of 8 operations which include a skip connect, two pooling, two separable and two dilated convolutions. This work considers the baseline DARTS search space $\mathcal{O}_D$ along with two compact variants listed in Table I. The variants allow exhaustive evaluation and ablation studies to be performed, given that the baseline DARTS search space is too large to be evaluated exhaustively with $\sim 10^{25}$ models.

Search Progression. The goal of the search is to find high-quality *operations* and *connections* within the cells. Specifically, DARTS-like works aim to find top-2 incoming edges per intermediate node (*topology search*) and the best operation per edge (*operation search*).

We follow a 3-step approach in our search progression: 1) Coarse-grained operation search for eliminating low-quality operations from search space in an approximate fashion with little search cost, 2) Fine-grained operation search for selecting the best operation on each edge (*discretization* of each edge) with high accuracy, and 3) Topology search for finding top-2 incoming edges for each intermediate node (*discretization* of each node) as per the goal of DARTS [5]. Dividing the search into these three steps reduces search cost significantly by performing the search at different scales; approximate and low-cost in the early stages, accurate and high-cost in later stages.

B. Related Works

Differentiable NAS. In recent years, differentiable one-shot approaches have gained traction due to their ability to jointly optimize a weight-shared supernet architecture and its network weights for efficient search [5], [10], [16]. DARTS [5] represents the search space by relaxing the categorical choice of a candidate operation to a softmax over all possible operations in the form of an over-parametrized 'supernet' (Fig. 1 (a), Top, Different colors on edges represent different operations in the candidate operation set). A supernet is a DAG representation of the search space that consists of all possible operations and topology connections that can be constructed from it. Each edge of the DAG consists of all candidate operations, represented by $\mathcal{O}$. The output of an edge, called a mixed operation, is the weighted sum of operations on that edge applied to the input. The weights associated with the candidate operations are parametrized by a vector α which represents the relative strength of their corresponding operations. The output of a mixed operation on edge $e \in \mathcal{E}$, where $\mathcal{E}$ is the list of all edges, is given as

$$\bar{o}^e(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^e)}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^e)} o(x) \tag{1}$$

Where $\mathcal{O}$ represents the candidate operations in the search space and $o(x)$ represents an operation o to be applied to the input feature map x. A bi-level optimization is utilized to jointly learn both architecture α as well as the network weights w (Fig. 1 (a), Center, Opacity represents operation value/strength, α).

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha) \end{aligned} \tag{2}$$

The final architecture is obtained by choosing the operations corresponding to the highest α (Fig. 1 (a), Bottom, Only branches with highest opacity on each edge remain).

Memory-Efficient NAS. DARTS [5] is known to suffer from memory bottleneck issues [9], [10]. The DARTS supernet consumes roughly $|\mathcal{O}|\times$ the GPU memory of a single model in the search space. Hence, DARTS-like approaches impose search space size limits due to the memory limits of the hardware. ProxylessNAS [9] proposed a solution in the form of binarized architecture paths. During the training of the supernet, only one of the $|\mathcal{O}|$ paths is activated using stochastically sampled binary gates, leading to a memory cost roughly the same as that of training a single compact model. However, search cost of over 8 GPU-days and model sizes $\sim$ 70% greater than DARTS, combined with only a modest improvement in accuracy, makes the approach less appealing. PC-DARTS [10] proposed another solution to the memory bottlenecks by sampling partial channel connections of the supernet during the search. This approach relies on performing the operation search using only a subset of feature map channels. However, the reproduced results of PC-DARTS [10] show unremarkable improvements over existing works in accuracy.

Perturbation methods. The success of deep learning comes from exploiting infinitesimal perturbations at an enormous scale. Stochastic gradient descent applies minuscule perturbations to the weights of a model so that, over time, the model weights satisfy the objective function better. Hence, it is natural to ask whether perturbations can also be exploited at an architectural level. DARTS+PT [6] showed that perturbing DARTS-optimized supernets can help evaluate and remove weak operations and connections from the search space. Our work is closely related to DARTS+PT but explores architectural

TABLE I
SEARCH SPACES CONSIDERED IN THIS WORK.

Search Space	Candidate Operations, $\mathcal{O}$	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{E}\rvert$	Topology
$\mathcal{O}_D$	DARTS [5]	8	14	Flexible
$\mathcal{O}_1$	{skip_connect, max_pool_3x3, sep_conv_3x3, sep_conv_5x5}	4	8	Shallow (d=1)
$\mathcal{O}_2$	Best ops for each edge in $\mathcal{O}_1$	1	14	Flexible

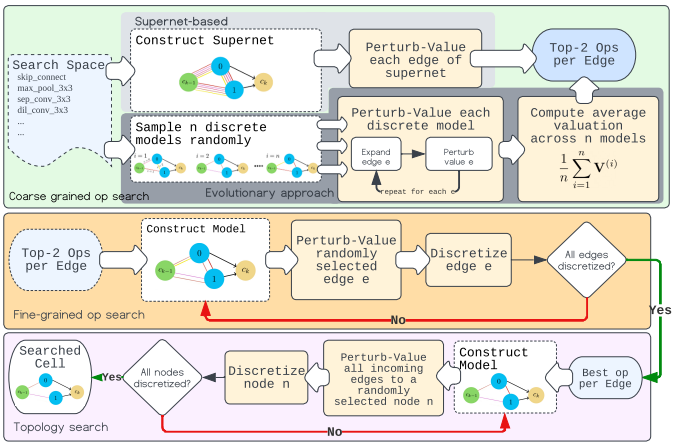

Fig. 2. Flowchart of our perturbations-based search. Supernet-based search relies on training a large supernet (light gray) while evolutionary approach relies on performing operation valuation from within n discrete models (dark gray) using expansion followed by perturb valuation.

perturbations as the basis to perform NAS without using gradient-based optimization utilized by DARTS-like works.

III. Search Method

A. Perturbations-based Evaluation

Approach. We explore a perturbations-based approach to identify high-quality components of the search space. Perturbations to an operation o on edge e refer to the removal of the operation from that edge. This is implemented by masking the corresponding operation as shown in Fig. 1 (c). The output of the masked mixed operation $\bar{o}_{m,e}$, which denotes operation m masked on edge e is given by

$$\bar{o}_{m,e}(x) = \sum_{o \in \mathcal{O}} \mathbb{1}_{o \neq m} \cdot o(x) \tag{3}$$

Where the mask vector $\mathbb{1}$ is the indicator function. Comparing the output of the mixed operation of our approach in Eq. 3 with that of differentiable approaches in Eq. 1, we note that we do not utilize α to represent operation strength. Instead, in our approach, the value/strength of the operation o on edge e, denoted by $\Delta V_{o,e}$, is defined as the change in accuracy resulting from the perturbation of o on edge e.

$$\Delta V_{o,e} = V_{o,e} - V^* \tag{4}$$

Where V^* is the validation accuracy of the unperturbed model, and $V_{o,e}$ is the accuracy when a perturbation is applied to o on e. The meaning of $\Delta V_{o,e}$ is reasonably intuitive to interpret: it is the degradation in accuracy of the model when operation o is removed from edge e, signifying the contribution of $\{o, e\}$ towards the accuracy of the model.

Approximation of Accuracy of Perturbed Models. For measuring operation strength, or *valuation* (Eq. 4), V^* can be obtained trivially by training the unperturbed model to convergence. However, since perturbed and unperturbed models have different architectures, obtaining $V_{o,e}$ is non-trivial since it needs to be obtained from the perturbed model with $\{o, e\}$ pair removed/masked. Let $s^*_\theta(x)$ and $s^{(o,e)}_{\theta'}(x)$ be the unperturbed model with parameters θ and perturbed model with parameters θ', respectively. It follows that $\theta' \neq \theta$ given $s^{(o,e)}(x) \neq s^*(x)$. Training $s^{(o,e)}_{\theta'}(x)$ from scratch to convergence is computationally impractical, given that this needs to be performed for each o, e pair. For a sufficiently small perturbation to the model such that the base and perturbed model architectures are fairly similar (i.e., $s^*(x) \sim s^{(o,e)}(x)$), θ' can be approximated from θ by sharing the weights of the base model with those of perturbed model and finetuning the perturbed model. This allows $V_{o,e}$ to be estimated from $s^{(o,e)}_{\theta'}(x)$ using $\theta' \sim \theta$ as the parameter initialization for the perturbed model in a computationally tractable manner.

B. Search Flow using Supernets

Figure 2 shows a flowchart of our approach detailing the three-stage search progression highlighted in Sec. II-A. Please note that cells depicted in Fig. 2 are simplified variants of the real cells used in experiments for ease of visualization. The actual cells utilize two input and four intermediate nodes as per the cell template utilized by DARTS [5] and its variants [5], [10], [16] for fair comparison.

Operation search is divided into two stages of different granularities; coarse- and fine-grained search. Coarse-grained search eliminates weak operations in a low-cost fashion. In contrast, fine-grained search caters to the operation search goal by discretizing edges of the DAG in succession and with high confidence. In coarse-grained operation search, we utilize perturbations to obtain valuations of all operations in the search space. We utilize two approaches to perform this valuation. The first approach, supernet-based search, shown in light gray in Fig. 2 top, utilizes perturbations on a *supernet* to obtain the valuation of all operations on each edge of the DAG. Specifically, we construct a supernet cell by placing all search space operations $\mathcal{O}$ onto each edge of the DAG. Eight cells are connected sequentially to form the search network, whereby cell k takes its input from the two previous cells, $k-1$ and $k-2$. After training the search network $s^*_\theta(x)$ to convergence and obtaining V^*, we perform the valuation of search space using the perturbations-based approach. To measure strength of operation o on an edge e, we mask o on e to construct $s^{(o,e)}_{\theta'}(x)$. We then initialize θ' with θ, i.e., $s^{(o,e)}_{\theta'=\theta}(x)$, and finetune this perturbed model so $s^{(o,e)}(x)$ reaches convergence. The validation accuracy of this perturbed model $V_{o,e}$ is used to calculate $\Delta V_{o,e}$. The process of o, e valuation is repeated for all $o \in \mathcal{O}$ and $e \in \mathcal{E}$ to obtain the valuation matrix $\mathbf{V} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{O}|}$, which represents the strength of each operation on each edge. For the DARTS search space, $\mathcal{O}_D$, this corresponds to $|\mathcal{O}| = 8$ and $|\mathcal{E}| = 14$. Valuation matrix $\mathbf{V}$ is utilized to select top-2 operations for each edge, i.e., the operations that cause the most degradation of accuracy when perturbed.

Fine-grained op search (Fig. 2 center) starts by training a model constructed by sequentially connecting eight cells containing only the top-2 operations per edge. We then select and perturb-value operations on a random edge e. We discretize e by selecting the best operation for e using the valuation, and construct and train the new model with e discretized. We repeat this process until all edges in $\mathcal{E}$ have been discretized. The generated model is sparse and consists only of the best operation for each edge. Topology search (Fig. 2 bottom) follows the same process as fine-grained operation search but discretizes nodes instead of edges. This entails selecting top-2 incoming edges for each of the intermediate nodes.

Existing works have highlighted the memory bottlenecks presented by supernets [9], [10]. Specifically, a supernet consumes roughly $|\mathcal{O}|\times$ the memory of a single model in the search space. This places restrictions on search space sizes owing to hardware limitations. In the following subsection, we demonstrate how GPU memory required for search can be disentangled from the search space size.

C. Disentangling Mem Consumption from Search Space Size

Supernets have enjoyed much of their success owing to weight-sharing; each sub-network can inherit its weights from the supernet, bypassing the need for exhaustively training each sub-network independently. However, numerous works have exposed the rank miscorrelation problem in weight sharing, whereby weight-shared discrete

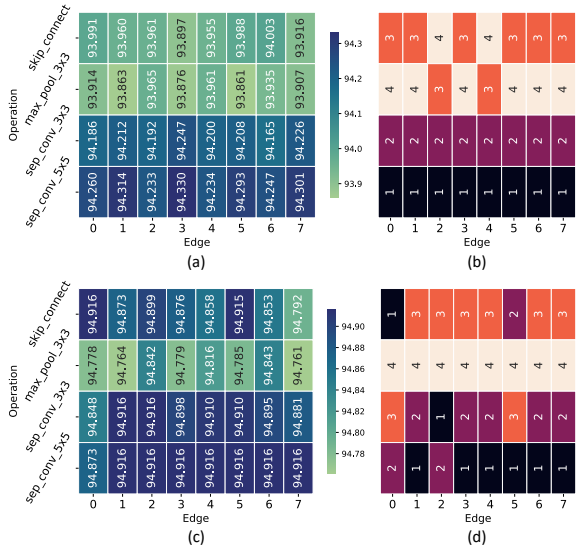

Fig. 3. Heatmaps of valuations and rankings obtained using (a)-(b) averaging architecture accuracies, and (c)-(d) obtaining accuracy of best model. They indicate that the rankings obtained by averaging model accuracies are fairly similar to rankings when best accuracy is selected for each operation choice.

architectures serve as poor surrogates for full model training [11], [14], [15]. Furthermore, high memory consumption of supernets imposes restrictions on search space size [9], [10]. We show that the performance of operations is not necessarily tied to the edge they are located in relative to the other edges. This would mean that operations on each edge can be valued independently without sharing weights between operations on the remaining edges.

We obtain the accuracies of all $|\mathcal{O}|^{|\mathcal{E}|} = 4^8 \sim 65k$ models in the $\mathcal{O}_1$ search space on NASBench-301 [12]. We compute the average accuracy of all possible *discrete* models (i.e., a model with one op per edge) that can be constructed from making each op selection on each edge (e.g., if we discretize edge 0 to the skip_connect operation, we can construct a total of $4^7 \sim 16.4k$ different architectures). Similarly, we also obtain the accuracies of the best discrete model that can be constructed by discretizing each operation on each edge. Fig. 3 shows heatmaps and operation importance rankings of the average and best accuracies of the remaining discrete architectures in the search space when each operation is discretized on each edge. We make two observations from these rankings: 1) Average rankings of operations are fairly consistent across all edges, i.e., some operations are almost always better than others, and 2) Top operations based on average and best rankings are fairly similar, i.e., the fact that sep_conv_5x5 and 3x3 are the best operations for almost all edges is consistent between the two rankings. Since averaging the performance of models gives a good indicator of operation importance, we conclude that operations on edges can be valued independently.

D. From Supernets to Evolutionary Search

Based on the idea that each edge can be valued in isolation while the remaining edges need not be overparametrized, we propose an evolutionary approach that does not rely on supernets. Figure 2 (dark gray) shows the coarse-grained operation search phase using this evolutionary technique. We randomly sample a population of n discrete models from the search space and train each model to convergence. A discrete model refers to a model with only one operation per edge. For each model, we then expand an edge, finetune the model with the expanded edge, and apply perturb-valuation on this edge according to the approach detailed in Sec. III-A. Expanding an edge refers to loading all candidate operations to that edge, as shown in Fig. 1 (b). After obtaining the valuation of operations on this edge, we discretize this edge according to the valuation, expand the next edge and repeat the valuation. Starting from the random selection of operations on all edges, the model evolves over time since each discretization step improves the operation selected for an edge. After every edge is valued using this successive expansion followed by the valuation technique, each discrete model i gives a valuation matrix $\mathbf{V}^{(i)}$, which corresponds to the values of search space operations within that discrete model. The valuation matrices from the n models are averaged to obtain the average valuation of the operations. We only utilize this evolutionary approach in the coarse-grained op search phase since this stage forms the memory bottleneck due to the expensive supernet training.

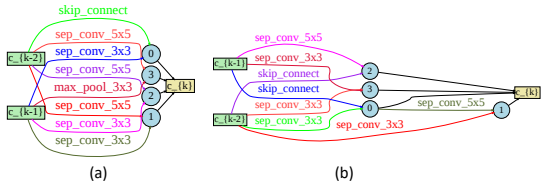

Fig. 4. Best cells obtained using (a) supernet and (b) evolutionary approach with CIFAR10 error rates of 2.65% and 2.49%, respectively.

IV. Experiments and Ablation Studies

This section studies the experiments and results of our approach on CIFAR10, NASBench-301, and ImageNet2012. We highlight the importance of search space scalability and, correspondingly, the memory consumption of search. We also perform ablation studies to validate perturbations as an accurate operation evaluation technique.

A. CIFAR10 – Full Search

This section details experiments for cell search on CIFAR10. We perform search using both the supernet-based and the evolutionary approach (Fig. 2) to show the difference in accuracy and memory consumption of the two approaches.

Experimental Setup. Most of our hyperparameter settings are similar to those utilized by [5] and [6] to allow fair comparison of results. We perform our experiments on the DARTS search space $\mathcal{O}_D$. We found the search results of the evolutionary approach to exceed that of the supernet-based approach using only $n = 4$ models in the initial population. The search follows the progression detailed in Sec. II-A and Fig. 2, whereby a *coarse-grained operation search* phase selects top-2 operations for each edge, followed by a *fine grained operation search*, and then the *topology search*. We repeat the search process with different random seeds four times to ensure consistency in evaluating results compared to previous works.

Evaluation. Evaluation follows the same settings as most existing DARTS-like works for consistency of comparison. We utilize 20 cells, with reduction cells at 1/3 and 2/3 of the depths of the model, where reduction cells have the same architecture as normal cells but with stride-2 operations. We train the models for 600 epochs with stochastic gradient descent, 36 initial channels, auxiliary towers with weight 0.4, and cutout with length 16. We report and compare the average and best accuracy of the four runs in Table. II. All works below the double horizontal line in Table. II utilize the DARTS search space. Architectures of the best searched cells of the four runs from the supernet-based and evolutionary approach are shown in Fig. 4. ProxylessNAS [9] achieves 0.41% and 0.03% better accuracy on CIFAR10 and ImageNet, respectively than the best cell found using our approach. However, the search cost of 8.3 GPU days, combined with model sizes roughly $1.4\times$ that of our results, makes the approach less appealing. Furthermore, ProxylessNAS has not been applied to DARTS-like search spaces, which makes the comparison inequitable.

TABLE II
PERFORMANCE COMPARISON AGAINST THE STATE-OF-THE-ART ON CIFAR10 AND IMAGENET2012.

Work	Error CIFAR10 (%)	Params CIFAR (M)	Error ImageNet Top 1	Error ImageNet Top 5	Params ImgNet (M)	Search Time CIFAR (GPU days)	Peak GPU Mem (GB) ‡
NASNet-A [4]	2.65	3.3	26.0	8.4	5.3	1800	-
AmoebaNet-B [17]	2.55 ± 0.05	2.8	26.0	8.5	5.3	3150	-
ENAS [8]	2.89	4.6	-	-	-	0.5	-
PNAS [18]	3.41 ± 0.09	3.2	25.8	8.1	5.1	225	-
ProxylessNAS [9]	2.08	5.7	24.9	7.5	-	8.3	-
SNAS [16]	2.85 ± 0.02	2.8	27.3	9.2	4.3	1.5	33.1
PC-DARTS [10] (Reproduced)	2.84† ± 0.09	3.6	25.1	7.8	5.3	0.1	11.8
R-DARTS [7]	2.95 ± 0.21	-	-	-	-	1.6	32.6
SGAS [19]	2.66 ± 0.24	3.7	26.0	8.5	5.3	0.3	32.6
DARTS [5]	3.00 ± 0.14	3.3	26.7	8.7	4.7	0.4	32.6
DARTS+PT (avg) [6]	2.61 ± 0.08	3.0	-	-	-	0.8	32.6
DARTS+PT (best) [6]	2.48	3.3	25.5	8.0	4.6	0.8	32.6
PertNAS (Supernet) (avg)	2.69 ± 0.04	3.9	-	-	-	0.2	17.8
PertNAS (Supernet) (best)	2.65	4.1	-	-	-	0.2	17.8
PertNAS (Evolutionary) (avg)	2.58 ± 0.08	4.1	-	-	-	0.2	6.2
PertNAS (Evolutionary) (best)	2.49	4.0	25.2	7.8	5.6	0.3	6.2

† Authors' reported accuracy: 2.57%, Reproduced accuracy using authors' official implementation: 2.84%
‡ Measured using an A100 GPU, PyTorch 1.12 + CUDA 11.3 @ batch size 256

TABLE III
PERFORMANCE COMPARISON ON NASBENCH-301, AND THE GPU-MEMORY CONSUMPTION FOR SEARCH ON CIFAR10.

Work	Accuracy (%)	Params (M)	Peak GPU Mem (GB)
SNAS [16]	94.22	2.8	33.1
PC-DARTS [10]	94.03	3.6	11.8
SGAS [19]	94.31	3.8	32.6
DARTS [5]	93.68	3.3	32.6
DARTS+PT [6]	93.84	3.3	32.6
PertNAS (Supernet)	94.58	4.1	17.8
PertNAS (Evolutionary)	94.71	4.0	6.2

Our evolutionary scheme exceeds the average accuracy achieved by PC-DARTS [10] by 0.26% and requires only around $0.5\times$ the memory. Against DARTS+PT, we achieve comparable accuracy with almost $1/4$ the search time while utilizing less than $1/5$ the memory.

B. NASBench-301

We compare the performance of the best-generated cells under both search schemes against SoTA DARTS-like works on NASBench-301 [12]. NASBench-301 is a surrogate benchmark that allows comparing the performance of NAS works without expensive model training. We utilize the two best cells obtained as a result of search on CIFAR10 (Sec. IV-A, Fig. 4) to perform the evaluation on NASBench. Table. III shows that the best cells generated using the evolutionary approach exceed DARTS [5] by 1.03%, PC-DARTS [10] by 0.68%, and DARTS+PT [6] by 0.87%. The corresponding cells, shown in Fig. 4, are rich in learnable parameters as they contain many convolutions, leading to large model sizes. Existing works have shown that searched model accuracies are proportional to their sizes [11], [12] as will also be shown in Sec. IV-E and Fig. 6 (a). Furthermore, high quality models utilize separable as opposed to dilated convolutions [12], and shallow as opposed to deep cells, as exhibited by our searched cells in Fig. 4.

C. Transferability Evaluation on ImageNet

We test the transferability of cells found on CIFAR10 to the ImageNet dataset. This test follows the mobile setting for evaluation whereby the number of multiply-add operations in the model is restricted to be less than 600M, with an input size of 224x224. The remaining hyperparameters also follow the configuration utilized by previous works for fair comparison [5], [6]. Comparison of top-1 and top-5 accuracy achieved on ImageNet under transferability evaluation of cells found on CIFAR10 against some state-of-the-art works is shown in Table. II. We achieve 0.3% better top-1 accuracy compared to DARTS+PT owing to the parameter-rich cells resultant from our approach. Against ProxylessNAS, we achieve 0.3% worse top-1 accuracy but with a search time of only 0.3 GPU-days. Against PC-DARTS, we achieve 0.1% worse top-1 accuracy. However, PC-DARTS performs search directly on each dataset, with a search on ImageNet costing 3.8 GPU-days, roughly $13\times$ our search time. Hence, our approach produces results comparable to the SoTA with minimal search cost in both time and memory.

D. Importance of Search-Space and Memory Scalability

The evolutionary approach allows the evaluation of the search space from within discrete models. Hence, the peak memory consumption of our evolutionary approach is similar to that of training a compact model sampled from the search space. We evaluate the peak memory allocated to search at different search space sizes using supernet-based and evolutionary techniques. We also showcase the importance of scaling up the search space by analyzing its impact on the quality of searchable architectures. Starting from $\mathcal{O}$={none, skip_connect, max_pool_3x3, avg_pool_3x3}, i.e., the subset of the DARTS search space with $|\mathcal{O}| = 4$, we scale up the search space by adding convolution operations with different kernel sizes into the search space, e.g., to make $|\mathcal{O}| = 5$, we add a sep_conv_3x3 into $\mathcal{O}$.

At each $|\mathcal{O}|$, we obtain the average accuracy of up to $10k$ randomly sampled models from the search space on NASBench-301, whereby fixed shallow search space topology, similar to $\mathcal{O}_1$ in Table. I, is utilized. This average accuracy denotes the quality of searchable models in the search space; higher average accuracy implies that search methods can potentially generate high-quality models from the search space. However, we can only obtain average accuracies for search space sizes up to $|\mathcal{O}| = 7$ due to limitations on the operations available in NASBench-301. We also measure the peak reserved memory for search, composed of memory allocated to tensors, cached memory, and CUDA context. The results in Fig. 5 show that while the GPU memory requirement for supernet-based search (e.g., DARTS-like approaches that rely on supernets) increases almost linearly with search space size, the memory requirement of the evolutionary approach stays almost constant. Hence, our approach offers possibilities for scaling the search space significantly, sidestepping the hardware limits imposed by supernet-based approaches like DARTS. This scalability is essential since larger search spaces hold more high-quality models, as depicted by the increasing average accuracy of models with increasing search space sizes $|\mathcal{O}|$.

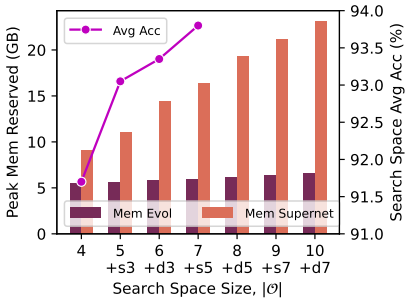

Fig. 5. Bar chart of GPU-memory consumption for search on search spaces of different sizes. Also shown average accuracy of $10k$ randomly chosen models in the search spaces (s3=sep_conv_3x3, d3=dil_conv_3x3). +s3 denotes search space size is increased by adding a sep_conv_3x3 op to the preceding $\mathcal{O}$.

E. Ablation 1 – Fidelity of Perturbations for Operation Search

This ablation study explores the validity of perturbations-based *operation search*. We utilize search space $\mathcal{O}_1$ with NASBench-301 for this study due to its fixed topology and the availability of ground truth valuation of operations in the search space. To perform search using the perturbations-based approach, we randomly sample a total of $n = 8$ discrete models from the search space. We train each of the discrete models to convergence, expand and value each edge using perturbations, average the valuation matrices obtained from each of the 8 models, and select the top o, e pair from the cell and discretize e to o. We repeat this process for the remaining edges until the best operation for each edge is selected. To compare the results of operation search on $\mathcal{O}_1$ against previous works, we perform similar searches using DARTS [5] and DARTS+PT [6]. Figure 6 (a) shows the accuracy of all $\sim 65k$ models in the search space and compares the accuracy of models obtained using the three different approaches: DARTS, DARTS+PT, and PertNAS. The cells resultant from our approach consist of 7 sep_conv operations, leading to higher accuracy and model size compared to DARTS+PT and DARTS.

F. Ablation 2 – Fidelity of Perturbations for Topology Search

For evaluation of perturbations for *topology search*, we utilize the search space of $\mathcal{O}_2$ whereby we enumerate all possible topologies that can be constructed from the best operation choices in $\mathcal{O}_1$. This corresponds to a total of only 180 topologies, with their accuracies on NASBench-301 [12], and cell depths shown in Fig. 6 (b). Shallower DARTS cells perform better, as also validated by [12]. We construct a supernet consisting of all 14 possible edges in each cell, with each edge consisting only of top operation from $\mathcal{O}_1$. We apply topology search on this supernet using perturb-valuation of all edges and selecting top-2 incoming edges for each intermediate node. Results in Fig. 6 (b) show that the topology obtained using this valuation scheme is among the top 5 topologies in this search space and achieves 0.16% better accuracy at the same model size as DARTS+PT [6].

V. Conclusion

This paper proposed a perturbations-based approach to evaluating search space components in NAS. We showed how perturbations can be applied to discrete models in an evolutionary search scheme to allow the valuation of search space in a memory-efficient fashion. The evolutionary approach disentangles memory consumption from search space size, overcoming the constraints imposed by hardware limits on search spaces. Experiments using CIFAR10, NASBench-301, and ImageNet show a competitive performance of perturbations-based search relative to differentiable approaches, at roughly only 20% of their memory costs. Our approach is fast, accurate, and scalable to larger search spaces. It generates architectures that conform to our understanding of high-quality architectures in the search space, such as shallow and parameter-rich cells.

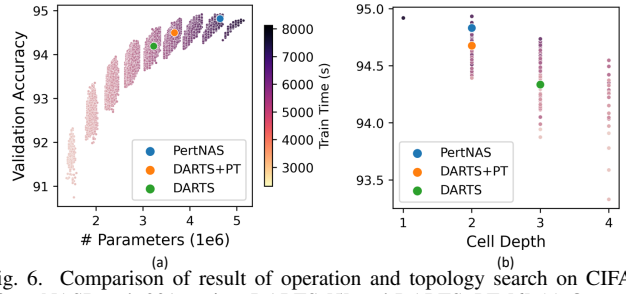

Fig. 6. Comparison of result of operation and topology search on CIFAR-10 on NASBench-301 against DARTS [5] and DARTS+PT [6] (a) Operation Search Space $\mathcal{O}_1$ with $65.5k$ models, and (b) Topology Search Space $\mathcal{O}_2$ with only 180 models plotted against cell depth. Hue represents train time.

Acknowledgements

This research was partially supported by ACCESS - AI Chip Center for Emerging Smart Systems, Hong Kong SAR. The authors also acknowledge compute support from Cloudlab [20] and Turing AI Compute Cluster (TACC).

References

[1] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *ICCV*, 2019.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2017.

[3] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, 2019.

[4] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[5] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[6] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable nas," *arXiv preprint arXiv:2108.04392*, 2021.

[7] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," *arXiv preprint arXiv:1909.09656*, 2019.

[8] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*. PMLR, 2018.

[9] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv:1812.00332*, 2018.

[10] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," *arXiv preprint arXiv:1907.05737*, 2019.

[11] A. Zela, J. Siems, and F. Hutter, "Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search," *arXiv preprint arXiv:2001.10422*, 2020.

[12] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter, "Nas-bench-301 and the case for surrogate benchmarks for neural architecture search," *arXiv preprint arXiv:2008.09777*, 2020.

[13] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search." PMLR, 2020.

[14] Y. Zhang, Z. Lin, J. Jiang, Q. Zhang, Y. Wang, H. Xue, C. Zhang, and Y. Yang, "Deeper insights into weight sharing in neural architecture search," *arXiv preprint arXiv:2001.01431*, 2020.

[15] A. Pourchot, A. Ducarouge, and O. Sigaud, "To share or not to share: A comprehensive appraisal of weight-sharing," *arXiv preprint arXiv:2002.04289*, 2020.

[16] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[17] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019.

[18] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.

[19] G. Li, G. Qian, I. C. Delgadillo, M. Muller, A. Thabet, and B. Ghanem, "Sgas: Sequential greedy architecture search," in *CVPR*, 2020.

[20] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb *et al.*, "The design and operation of cloudlab." in *USENIX Annual Technical Conference*, 2019.