CSCO: Connectivity Search of Convolutional Operators

Tunhou Zhang[1], Shiyu Li[1], Hsin-Pai Cheng[3], Feng Yan[2], Hai Li[1], Yiran Chen[1]
[1]ECE Department, Duke University, Durham, NC 27708
[2]Department of Computer Science, University of Houston, Houston, TX 77204
[3] Qualcomm AI Research, San Diego, CA 92121
[1] {tunhou.zhang,shiyu.li,hai.li,yiran.chen}@duke.edu,
[2] fyan5@central.uh.edu,
[3] hsinpaic@qti.qualcomm.com

Abstract

Exploring dense connectivity of convolutional operators establishes critical "synapses" to communicate feature vectors from different levels and enriches the set of transformations on Computer Vision applications. Yet, even with heavy-machinery approaches such as Neural Architecture Search (NAS), discovering effective connectivity patterns requires tremendous efforts due to either constrained connectivity design space or a sub-optimal exploration process induced by an unconstrained search space. In this paper, we propose CSCO, a novel paradigm that fabricates effective connectivity of convolutional operators with minimal utilization of existing design motifs and further utilizes the discovered wiring to construct high-performing ConvNets. CSCO guides the exploration via a neural predictor as a surrogate of the ground-truth performance. We introduce Graph Isomorphism as data augmentation to improve sample efficiency and propose a Metropolis-Hastings Evolutionary Search (MH-ES) to evade locally optimal architectures and advance search quality. Results on ImageNet show $\sim 0.6\%$ performance improvement over hand-crafted and NAS-crafted dense connectivity. Our code is publicly available here.

1. Introduction

The fundamental success of Convolutional Neural Network (CNN) on Computer Vision lies in the effective wiring pattern, represented by dense connectivity within convolutional layers [20, 21] and atomic-level neurons [2]. Throughout neural synapses, a convolution operator, as an elementary atomic building operator, establishes receptive fields to extract spatial-local information in 2D images.

However, from classic CNNs [16, 25, 38] to modernized CNNs driven by Neural Architecture Search (NAS) [40, 41], the construction of CNNs mainly innovates an effective building block composed of a combination of building operators and directly stacks a few copies of these operators to construct the overall architecture. On images, most CNN designs are constrained to a chain-like structure without delicate consideration of building block connectivity. On the one hand, hardware is designed to handle better chain-like architectures such as MobileNets [18, 19, 37]. On the other hand, chain-like CNN architectures are easier to study, requiring less extensive efforts to fully explore, thus yielding a better rate of improvement (ROI) in research and development on vision benchmarks. The limitations in the aforementioned chain-like designs may prevent the discovery of effective inter-block synapses that enhance feature interaction in different positions of CNN architectures.

As a result, more recent works start to scratch the surface of dense connectivity by constructing a graph representation of the network design space [29, 33, 45, 52]. These works explicitly seek a building cell with searchable wiring of building operators via Directed Acyclic Graphs (DAGs) as the design motif for CNN architectures. Various search strategies are implemented to achieve a good architecture outcome, such as differentiable-based search [27, 29], Bayesian Optimization [24, 44], and local search [6, 43]. However, these methods employ brutal-force optimization algorithms such as differentiable-based search [22, 29], reinforcement-learning [33, 52] without any topology consideration when seeking the optimal wiring of building operators. Despite the remarkable success, existing methods may have the following challenges: (1) The constrained design space does not support the dense connectivity of versatile building operators that integrate feature vectors from all building cell levels without limitations. (2) With unconstrained dense connectivity, exploring such design space is difficult without topological information in the graph representation of building cells, such as isomorphism in adjacency matrices and locality contained with similar graphs.

In this paper, we tackle the above challenges by proposing a new paradigm, CSCO (**C**onnectivity **Search** of **C**onvolutional **O**perators), that enables the delicate exploration of the structural wiring within building cells for CNN architectures. CSCO establishes a hierarchical structure of CNN architectures via a meta-graph comprising several Directed Acyclic Graphs (DAGs). Each DAG represents a building cell in each hierarchy. Within each building cell, CSCO integrates a structural design space with versatile building operators (e.g., convolution, depthwise convolution) with varying transformation capacities of input features, allowing dense connectivity searches. The combination of dense edge connectivity and versatile heterogeneous operators we employ can craft various design motifs. For example, depthwise separable convolution [19], Inverted Bottleneck [37], Inception-like [39], and Condensenets [21], etc. This covers most design motifs from hand-crafted CNN design principles and more recent block-based search spaces in NAS (e.g., ProxylessNAS [1], MobileNetV3 [18] etc.), providing more opportunities to obtain top-performing CNN architectures with minimal design space constraints and priors.

Intuited by predictor-based NAS [10, 42] that accurately models the design space via a surrogate model of the ground-truth performance, we follow this principle and address two key factors to demystify search on dense connectivity. First, we propose Graph Isomorphism to enrich architecture-performance pairs during the sampling phase of predictor-based NAS, enhancing the quality of performance prediction with improved sample efficiency. As a result, Graph Isomorphism advances prediction reliability in dense connectivity design space with large cardinality. Second, we propose Metropolis-Hastings Evolutionary Search (MH-ES), which evades local optimal solutions during search space exploration. This allows us to approach a better region of the dense connectivity design space and discover better building cells for CNNs.

CSCO improves both the evaluation strategy (i.e., the quality and reliability of prediction) and the search strategy (i.e., the quality of top-performing architectures discovered) in predictor-based NAS. As a result, CSCO discovers good CNN architectures that achieve impressive empirical results over existing hand-crafted and NAS-crafted connectivity on ImageNet. We summarize our contributions as follows:

- We propose a new paradigm, CSCO, to automatically explore dense connectivity within building cells to fabricate CNN architectures. CSCO supports dense connectivity search on structural wiring of versatile convolutional building operators to seek the optimal CNN architecture.
- We pioneer using predictor-based NAS in dense connectivity search and demonstrate two essential techniques that advance search quality and efficiency. Specifically, we propose Graph Isomorphism to improve sample efficiency and introduce Metropolis-Hastings Evolutionary Search (MH-ES) to improve search quality.
- We thoroughly evaluate each component of CSCO and demonstrate 0.6% accuracy gain over existing NAS-crafted dense connectivity CNN architectures under mobile computation regimes.

2. Related Work

Dense Connectivity of Convolutional Neural Networks. Existing NAS methods emphasize cell design on a graph design space [29, 33]; these methods are primarily topology-agnostic. For example, DARTS implies addressing the bi-level optimization problem without considering any graph information (e.g., graph adjacency and graph isomorphism). More recent works [3, 11, 45, 51] manage to incorporate topological information into search space design and improve the flexibility of crafted CNN architectures. Yet, they still focus on a constrained search space emphasizing either a single-cell design (i.e., normal cell and./or reduction cell) or macro-level connections between building cells with constraints and limitations. Another line of research takes advantage of network generators [36, 47, 48] to obtain the best cluster of CNN architectures. Yet, these methods emphasize discovering the top-performing local regions of the search space and may miss the opportunity to discover an individual architecture with a globally optimal solution.
Predictor-based NAS. Neural predictors [42] are potent tools to map candidate architectures to their performance in a search space. Predictor-based NAS has two significant phases: (1) train an accuracy predictor based on exploitable architecture-performance samples collected from a search space, and (2) utilize the accuracy predictor to probe the whole search space and obtain the top-performing architectures. Existing works enhance predictor-based NAS in sample-efficiency [10, 30], quality of prediction [4], and better regions of the search space [46]. Yet, these works focus on a constrained block-based search space without considering topology, making them unsuitable for exploring cell structures for better sample efficiency. In addition, existing approaches mostly employ Evolutionary Search [5, 34, 35] as the backbone methodology to obtain top-performing architectures on neural predictors, gradually approaching a narrower local region of the dense connectivity design space and ending up with locally optimal architectures.

3. Dense Connectivity Design Space

CNN architectures comprise a series of building cells that transform input features into learned representations. For input features with varying properties (e.g., different spatial sizes), CNN architectures employ stage to represent the building cells with identical input sizes and repeat building

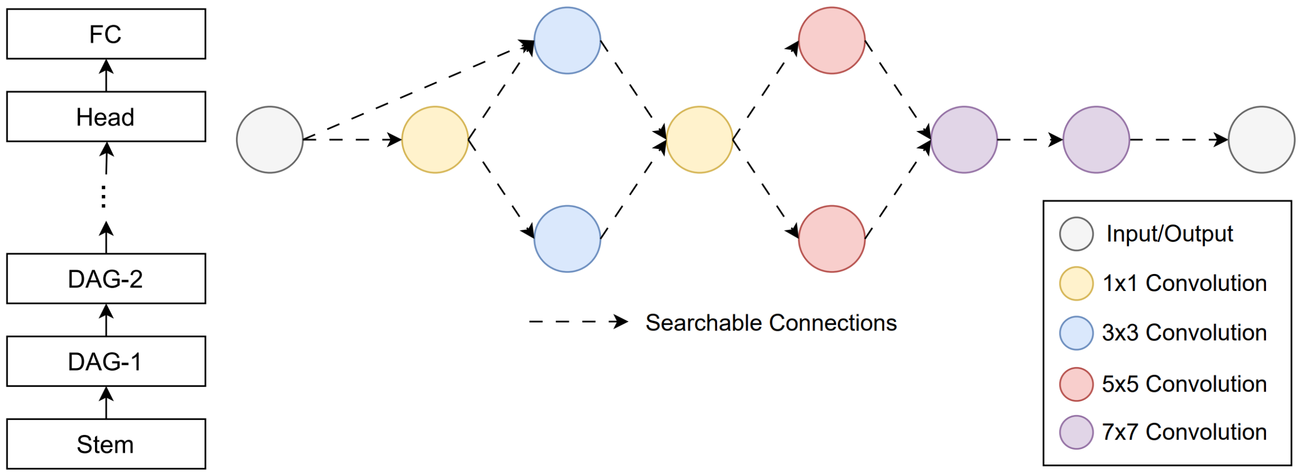

Figure 1. Overview of dense connectivity design space.

cells to build deeper architectures. Our dense connectivity design space delicately seeks the wiring of versatile convolutional operators in a building cell. Figure 1 demonstrates an overview of our dense connectivity design space. We utilize existing positional settings (i.e., $\times D$, C denotes D copies of DAG with C base channels) and delicately seek the edge connections that wire versatile building operators independently for all building cells. We first discuss the graph representation of CNN architectures in dense connectivity design space and then discuss versatile building operators.

3.1. Graph Representation of CNN

Given a CNN architecture with K stages, we specify independent Directed Acyclic Graphs (DAG) for each hierarchical level of input features to represent the dense connectivity of building versatile operators. A DAG is a building cell with multiple CNN layers and identical spatial feature sizes (i.e., image height and width). Each DAG $\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$ contains N vertices and $[N \cdot (N-1)/2]$ edges with total capacity for dense connectivity search. Among N vertices, vertex 0 is defined as the input vertex that receives an output from the previous building cell, and vertex N is the output vertex that sends an output to a succeeding building cell. We define vertex $1 \sim N-1$ as intermediate vertices. Each intermediate vertex takes input features X from an arbitrary number of preceding vertices and produces an output Y via an assigned building operator *op*. More specifically, each intermediate vertex v can make a connection to any preceding vertex $u_0, ..., u_M$, concatenates all these inputs, and use the assigned building operator *op* to produce output representations as follows:

$$Y_v = op_v[Concat(X_{u_0}, X_{u_2}, ..., X_{u_M})], \qquad (1)$$

$Concat$ denotes the feature concatenation in the channel dimension, and M denotes the number of connections that vertex v makes to preceding vertices.

The output vertex collects the output representations from each leaf vertex in DAG and concatenates them as an output to the succeeding building cells as follows:

$$Y_N = Concat(\{Y_{u_i} | d_{out}(i) = 0\}), \qquad (2)$$

where $d_{out}(\cdot)$ denotes the out degree of a vertex. A meta-graph $\mathbf{G} = (\mathcal{G}^{(1)}, ..., \mathcal{G}^{(K)}) = (\mathbf{V}, \mathbf{E})$ composes K DAGs to build the overall CNN architecture that processes input features of different hierarchies. We construct each candidate architecture $\mathcal{A}$ by a function of vertices and edge connections on the meta-graph (i.e., the union of all independent DAGs): $\mathcal{A} = f_{arch}(\hat{\mathcal{V}}^{(1)}, ..., \hat{\mathcal{V}}^{(K)}, \hat{\mathcal{E}}^{(1)}, ..., \hat{\mathcal{E}}^{(K)}; \hat{op}^{(1)}, ..., \hat{op}^{(K)})$. In dense connectivity design space, we assign each vertex an atomic convolutional operator from a set of versatile building operators and seek the best connectivity by optimizing edge connectivity $\hat{\mathcal{E}}^*$ as follows:

$$\arg\max_{\mathcal{E}^* \subset \mathbf{E}} Perf(f_{arch}(\hat{\mathcal{V}}^{(1)}, ..., \hat{\mathcal{V}}^{(K)}, \mathcal{E}^{*(1)}, ..., \mathcal{E}^{*(K)}; \hat{op}^{(1)}, ..., \hat{op}^{(K)})), \qquad (3)$$

where $Perf(\cdot)$ denotes the performance metric, and f_{arch} transforms a meta-graph representation to a concrete CNN architecture. Given a meta-graph with K independent DAGs (e.g., K stages) and N vertices each, a dense connectivity design space optimizes $O(K \cdot N^2)$ dense edge connections to seek the optimal architecture and contains a much richer source of architecture fabrications.

3.2. Convolution Building Operators

The dense connectivity design space is built upon versatile atomic building operators to remove the constraints in CNN design and enhance flexibility during the search. An atomic

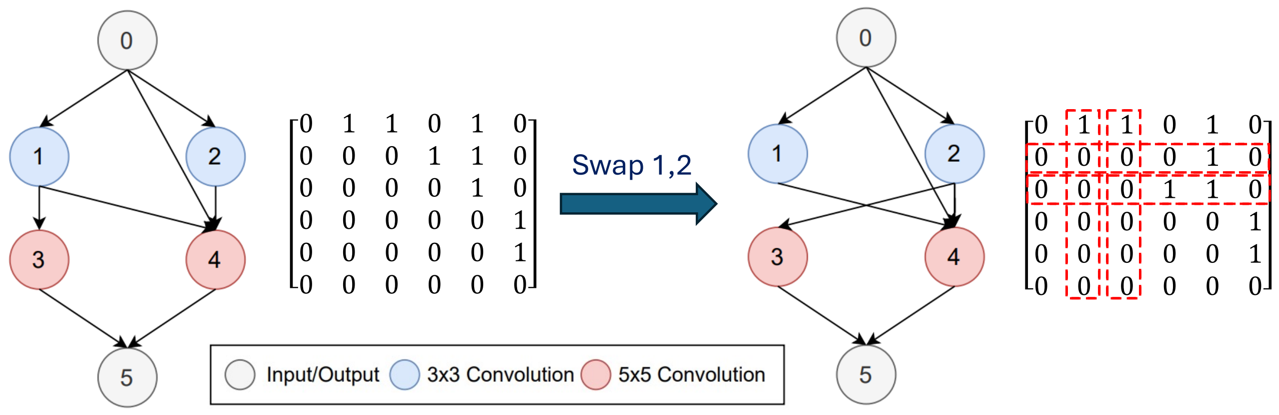

Figure 2. Graph Isomorphism creates extra training examples without extra cost.

convolution operator should contain precisely one convolution operation, followed by batch normalization [23] and ReLU activation [38]. We collect the popular design motifs from the existing literature and use the following set of building operators to craft the dense connectivity space:

- Convolution 1×1.
- Depthwise convolution 3×3, 5×5, or 7×7.

Convolution 1×1 and depthwise convolution are heterogeneous building operators with distinct functionality on input features: convolution 1×1 learns a transformation of local input features, and depthwise convolution learns a transformation of spatial input features. Next, we instantiate two dense connectivity spaces for ImageNet and CIFAR-10 as follows:

- *ImageNet Dense Connectivity Space.* Each meta-graph contains 4 stages which corresponds to the 4×4, 8×8, 16×16, 32×32 down-sampling region of an input image. In each DAG, we employ one input vertex, one output vertex, and 16 intermediate vertices assigned with one of the aforementioned convolutional building operators. We follow MobileNetV2 [37] for the design of stem architecture (i.e., first three blocks) and head architecture (i.e., last two blocks).
- *CIFAR-10 Dense Connectivity Space.* Each meta-graph contains 4 stages which corresponds to the 1×1, 2×2, 4×4 down-sampling region of an input image. In each DAG, we employ one input vertex, one output vertex, and 16 intermediate vertices assigned with one of the aforementioned convolutional building operators. We follow ResNet [16] to design stem architecture (i.e., first block) and employ no head architecture.

Notably, we set the number of vertices to far exceed that of versatile building operators in dense connectivity design space to ensure scalability. The dense connectivity design space is prohibitively large. Even with $N=8$ vertices, a single DAG contains 4.5×10^6 architectures. Consequently, an *ImageNet Dense Connectivity Space* contains up to 4×10^{26} architectures, calling for an effective and efficient algorithm to demystify dense connectivity optimization thoroughly.

4. Demystifying Dense Connectivity Search

A dense connectivity design space contains a rich source of candidates to ensure flexibility. However, versatile building operators and the dense connectivity design space challenge the efficiency and quality of search. CSCO incorporates two key techniques that facilitate architecture exploration in the dense connectivity design space. First, CSCO adopts Graph Isomorphism to augment architecture-performance pairs in the dense connectivity design space to boost the accuracy predictor's prediction quality without additional cost. Second, CSCO proposes a novel search strategy, Metropolis-Hastings Evolutionary Search (MH-ES), as an in-place improvement over evolutionary search during full search space exploration via a trained predictor. Inspired by the definition of Markov Chains, MH-ES rejects weaker samples with a lower probability and effectively evades local optimal solutions in discovery.

4.1. Graph Isomorphism as Data Augmentation

As is discussed in Section 3.2, a DAG within a dense connectivity design space contains more building operators than the number of vertices. This provides a chance to find isomorphic structures in the dense connectivity design space and further exploit these architectures to enhance the performance of predictor-based NAS. We formally define the isomorphism of meta-graphs as follows:

Definition 1. *Two meta-graphs* $\mathbf{G}, \hat{\mathbf{G}}$ *are isomorphic if every pair of DAG:* $(\mathcal{G}^{(\cdot)}, \hat{\mathcal{G}}^{(\cdot)})$ *is isomorphic.*

Isomorphic meta-graphs represent the same architecture

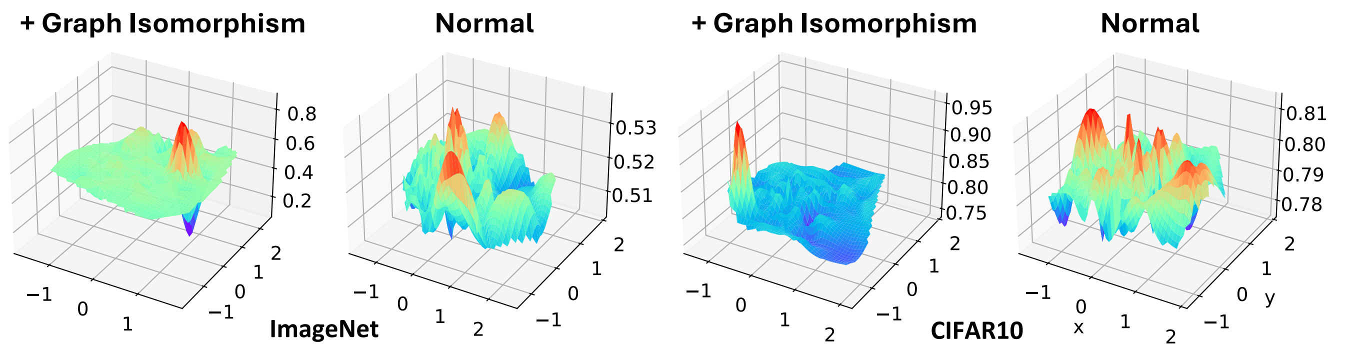

Figure 3. Accuracy surface of a performance predictor with/without Graph Isomorphism.

in the dense connectivity search space. This is because isomorphic meta-graphs have an identical set of building operators and identical dense connectivity of these building operators, see Figure 2. As a result, isomorphic meta-graphs represent the same neural architecture, leading to the same level of performance during evaluation.

Thus, we propose Graph Isomorphism to augment the architecture samples. Graph Isomorphism conducts a valid vertex permutation to one of the DAGs within each sampled meta-graph to construct a new isomorphic meta-graph and incorporate it as a new architecture sample with no extra search cost. These isomorphic samples can augment the architecture-performance pairs to brew a more accurate performance predictor without additional search costs.

Prediction Surface. We visualize the prediction surface of performance predictors with/without Graph Isomorphism in Figure 3. Here, a higher z-axis value denotes better predictive performance on the target dataset. Notably, Graph Isomorphism not only enhances the prediction quality of a performance predictor but also provides a smooth performance surface that eases the following search process in an ample, dense connectivity design space.

4.2. Metropolis-Hastings Evolutionary Search

Evolutionary Search is a popular method that efficiently explores the best architecture in predictor-based NAS. Yet, these methods may not efficiently explore our dense connectivity design space, thus suffering from the sub-optimal quality of discovered CNN architectures. We follow the same intuition of evolutionary search and first define the mutation space as follows:

- Re-sample a random edge connection for one DAG.
- Randomly add an edge connection for one DAG.
- Randomly remove an edge connection for one DAG.

Given an intermediate meta-graph with N vertices and K stages, the mutation space covers up to $O(N^{2K})$ possible candidate architectures, thus being prohibitively large for existing evolutionary search algorithms to explore fully. For example, (1) tremendous samples are needed to cover the good regions of the dense connectivity space and obtain the best child architecture, and (2) the complexity of prediction surface in dense connectivity design space may lead to the discovery of locally optimal solutions. This is because evolutionary search judiciously accepts the strongest child architectures during the evolutionary process and, thus, obtains locally optimal solutions with high concentration on a specific region of the dense connectivity design space.

We are inspired by Markov Chain Monte Carlo (MCMC) optimization, especially Metropolis-Hastings Algorithm [32], extensively addressing such issues by adopting an acceptance-rejection mechanism. Such mechanism maintains a current best solution and admits weaker solutions with an acceptance-rejection probability AC, defined as follows:

$$AC = \min(1, \exp\left((score' - score)/T\right)), \tag{4}$$

Where $score$ denotes the score (e.g., predicted performance of architectures) for a weaker solution, $score'$ denotes the score of the current best solution, and T denotes the temperature. The acceptance-rejection probability is proportional to the gap between the weaker solution and the current best solution. As a result, the optimization process may not greedily stick to the current best solution and thus have better potential to avoid locally optimal solutions.

Thus, we propose Metropolis-Hastings Evolutionary Search (MH-ES) as an alternative to existing evolutionary search algorithms on dense connectivity design space. MH-ES allows the discovery of better architectures within the large dense connectivity design space. MH-ES maintains the best parent architecture, which is initialized among P_0 randomly sampled candidate architectures in the initial population. Then, child architectures are obtained by randomly mutating one of the stages (i.e., DAGs) in the parent architecture (i.e., meta-graph). Each evolution round selects the best child architecture in the current population as a weaker

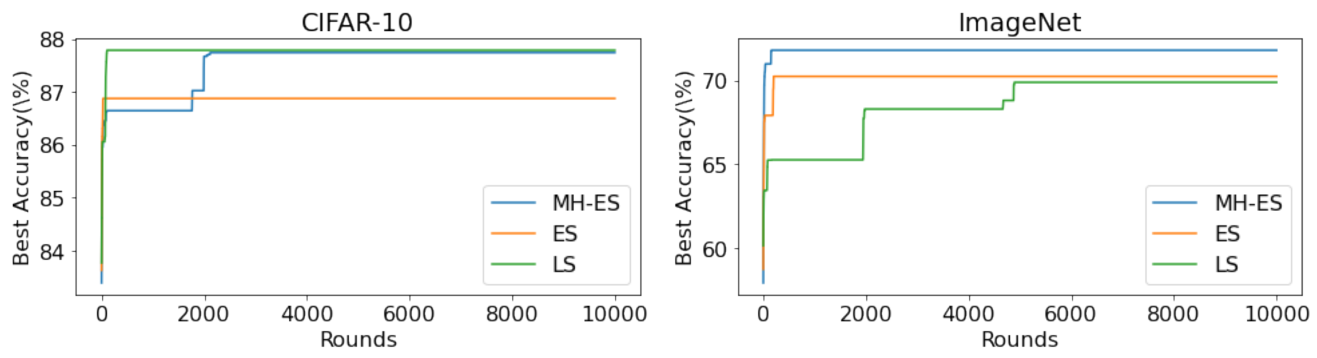

Figure 4. The search progress and predicted accuracy of discovered architectures via MH-ES compared to ES and RS baselines.

solution. The aforementioned MH acceptance-rejection ratio AC is applied to update the current best architecture. The proposed MH-ES generalizes to local search when $T \to 0$ and evolutionary search when $T \to \infty$. MH-ES also adopts a cosine simulated annealing [34] of the temperature to eliminate locally optimal solutions at early evolutionary rounds.

Optimization Curve of MH-ES. We compare the optimization curve of MH-ES versus Evolutionary Search (ES) [14] and Local Search (LS) [12] to demonstrate its effectiveness and efficiency. Figure 4 depicts the optimization curve of accuracy on top-performing CNN architectures for both CIFAR-10 and ImageNet. On a small-scale CIFAR-10 dense connectivity design space, MH-ES performs on-par as local search yet significantly outperforms evolutionary search by ~ 0.01. This is greatly attributed to the capability of MH-ES to evade locally optimal solutions during architecture exploration over the dense connectivity design space. On large-scale ImageNet dense connectivity design space, MH-ES significantly edges other search algorithms, highlighting its efficiency and effectiveness in exploring dense connectivity design space.

5. Experiments

Following all NAS methods focusing on connectivity, we apply CSCO to obtain promising CNN architectures within mobile computation regimes on CIFAR-10/ImageNet-1K [7] classification over *CIFAR-10 Dense Connectivity Space* / *ImageNet Dense Connectivity Space*.

5.1. CSCO Setup

We first elaborate on the search settings on CSCO, including search space configuration and MH-ES guided by a trained neural predictor. Then, we discuss the evaluation settings of CSCO over a dense connectivity design space.

Search Space Settings. We set a search budget of 4 GPU days and employ a fixed assignment of the building operators in all DAGs of the meta-graph. This enhances the reproducibility of our methods. We employ a large-scale dense connectivity design space with $N = 18$ vertices, where vertex $1, 5, 9, 13$ are assigned with convolution 1×1, vertices $2, 6, 10, 14$ are assigned with depthwise convolution 3×3, vertex $3, 7, 11, 15$ are assigned with depthwise convolution 5×5, and vertex $4, 8, 12, 16$ are assigned with depthwise convolution 7×7. We define vertex 0/17 as the input/output vertex.

Predictor Training and MH-ES. We train the above MLP performance predictor on the sampled architecture-performance pairs for 300 epochs with batch size 128, initial learning rate 0.1, and an L2 weight decay of 1e-4 for ImageNet. During MH-ES, we employ an initial population of 4096 to ensure the discovery of a good parent architecture. We proceed with 10K rounds of MCMC optimization with 96 child architectures sampled and evaluated in each round. We set the sensitivity parameter to 0.001 for the best solution in the dense connectivity design space.

Evaluation Settings. The outcome of CSCO leads to a pool of candidate CNN architectures for both CIFAR-10 and ImageNet, respectively. We simply evaluate the top-5 models on CIFAR-10/ImageNet proxy dataset for 20/10 epochs and scale the best model to 600M Multiply-Accumulates (MACs) mobile computation budget.

Table 1. CIFAR-10 evaluation of best models.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU Days)
WRN-28-10 [50]	4.17	36.5	-
DenseNet-BC [20]	3.46	25.6	-
PNAS [28]	$3.41_{\pm 0.09}$	3.2	-
AmoebaNet-A [35]	$3.34_{\pm 0.06}$	3.2	3150
DARTS (1st-order) [29]	$3.0_{\pm 0.14}$	3.3	4
GDAS [9]	2.93	3.4	0.3
CSCO (Ours)	**2.82**	**3.1**	4

Table 2. ImageNet-1K results. All models use 224×224 input under mobile settings.

Architecture	**Test Err.(%)**		**MACs**	**Params**	**Search Cost**
	top-1	**top-5**	**(M)**	**(M)**	**(GPU days)**
CondenseNet [21]	26.2	8.3	529	4.8	N/A
MobileNetV2 1.4 [37]	25.3	7.5	585	6.9	N/A
ProxylessNAS-G [1]	25.4	7.8	320	4.1	8.33
MnasNet-A1 [41]	24.8	7.5	312	3.9	1.7K
DARTS [29]	26.7	8.7	574	4.7	4
NASNet-A [52]	26.0	8.4	564	5.3	2K
RandWire-WS [47]	$25.3_{\pm 0.25}$	$7.8_{\pm 0.15}$	$583_{\pm 6.2}$	$5.6_{\pm 0.1}$	N/A
MiLeNAS [15]	24.7	7.6	584	5.3	0.3
PC-DARTS [49]	24.2	7.3	597	5.3	3.8
TopoNAS [22]	24.1	7.2	598	5.3	6.2
GAEA + PC-DARTS [26]	24.0	7.3	N/A	5.6	3.8
DenseNAS [11]	23.9	-	479	-	2.67
CSCO (Ours)	**23.3**	**6.7**	598	5.7	8

5.2. CIFAR-10 Experiments

We first evaluate each component of the CSCO paradigm and then proceed to evaluate the top-performing architecture discovered on CIFAR-10.

Evaluating Best CIFAR-10 Model. In CIFAR-10, we follow DARTS-series architectures [29, 49]) and stack 6 building cells in each stage to construct the final CNN architecture. To match the number of parameters reported in the DARTS-series paper, we apply a width multiplier to scale up the candidate networks to ensure a fair comparison with existing state-of-the-art.

We follow the DARTS protocol to train the best network. Specifically, we train the best network discovered by CSCO on 50K CIFAR-10 training data from scratch for 600 epochs with batch size 96. We employ an initial learning rate of 0.025 with a cosine learning rate schedule [31]. Following DARTS series works, we employ Dropout [17], DropPath [13] and Cutout [8] with a L-2 weight-decay of 3e-4 to combat overfitting.

The key results of CSCO on the CIFAR-10 dataset are summarized in Table 1. CSCO outperforms SMBO-based PNAS and EA-based AmoebaNet by up to 0.42%. Compared with DARTS and GDAS, CSCO achieves up to 0.2% better accuracy within a reasonable 4 GPU day search cost.

5.3. ImageNet Classification

We evaluate the best architectures crafted by CSCO on ImageNet by training them from scratch using the same training protocol as previous works. We compare the accuracy versus various metrics such as Multiply-Accumulates (MACs) with hand-crafted and NAS-crafted models. We train the best-discovered model on 1.28M ImageNet-1K training data from scratch for 450 epochs with batch size 768. We employ an initial learning rate of 0.6 with cosine learning rate schedule [31]. Following DARTS series works, we employ Inception pre-processing [39], Dropout [17], Drop Path [13], an L2 weight-decay of 1e-5.

Table 2 demonstrates the critical results of CSCO on the ImageNet-1K validation set within the mobile computation regime (i.e., $\leq$ 600M MACs). CSCO outperforms Condensenet [21] by 3% higher accuracy, demonstrating its superiority over prior art with dense connectivity of building operators in CNN architectures. CSCO outperforms SOTA hand-crafted models MobileNetV2 1.4 by 2.0% higher top-1 accuracy with similar MACs, where the latter one is crafted via manual architecture engineering. Compared to existing NAS works that emphasize dense connectivity [11, 15, 22, 26, 49], CSCO achieves $\sim$ 0.6% accuracy gain under the mobile computation regime with comparable parameter consumption. Despite having a sizeable dense connectivity design space, CSCO maintains a reasonable search cost of 8 GPU days thanks to Graph Isomorphism, which boosts sample efficiency. Finally, CSCO also achieves competitive performance compared to existing block-based NAS methods [1, 41] under a well-designed MobileNetV2-like search space, demonstrating the potential of optimizing dense connectivity to seek high-performing CNN architectures in the future.

6. Discussion

CSCO employs two key components in dense connectivity search space: Graph Isomorphism and Metropolis-Hastings Evolutionary Search (MH-ES). In this section, we discuss the individual contribution of Graph Isomorphism and MH-ES towards better dense connectivity in discovery.

Table 3. Evaluation of Graph Isomorphism and MH-ES on CIFAR-10.

Graph Isomorphism?	Search Strategy	Best Acc.	Mean Acc.	Std. Acc.
-	Random Search (RS)	91.52	-	-
	MH-ES	92.26	92.11	0.10
✓	MH-ES	92.55	92.39	0.11
✓	Local Search (LS)	92.33	92.25	0.05
✓	Evolutionary Search (ES)	92.49	92.07	0.23

6.1. Ablation Studies

Due to the time-consuming process of complete architecture evaluation from scratch, we adopt a simple training protocol to evaluate the top models discovered by different Graph Isomorphism and search strategies. Table 3 demonstrates the detailed evaluation result, including the accuracy of best-performing architectures and the statistics on top-5 models to reflect the stability of the proposed method. Here, we compute all accuracies using top models selected by the trained predictor. Notably, each model contains only 3 building cells in each stage, and the building operator adopts 16, 32, and 64 filters, respectively, within each stage. We can see that the combination of MH-ES and Graph Isomorphism yields up to a 0.3% accuracy gain on the mean accuracy of top-performing models while achieving the best top-performing architecture among other baseline methods.

6.2. Ranking with Graph Isomorphism

We analyze the performance predictor trained with/without Graph Isomorphism. Before Graph Isomorphism, we sample $\sim$ 800 samples from the dense connectivity design space for both CIFAR-10 and ImageNet benchmark and augment $10\times \sim 12\times$ more samples via Graph Isomorphism without extra computation cost. We split all architecture-performance pairs into 85% training pairs and 15% testing pairs. We utilize a Multi-Layer Perceptron (MLP) performance predictor to map architectures (i.e., the union of adjacency matrices in each DAG within a meta-graph) to their predicted performance (i.e., evaluated accuracy). We delicately train the performance predictor on the training split with/without Graph Isomorphism.

Prediction Quality. We measure the prediction quality of neural predictors on the testing architecture-performance pairs via two famous ranking metrics: Pearson's ρ and Kendall's τ in Table 4. Here, all ranking coefficients are computed on the testing pairs, which are not utilized to train the performance predictor.

Using Graph Isomorphism, the prediction ranking quality (i.e., Kendall's τ) significantly increases from 0.215 to 0.904 on ImageNet and from 0.428 to 0.874 on CIFAR-10. In addition, the ranking evaluation reveals that more DAGs in the meta-graph lead to poorer sample efficiency in predictor-based NAS and thus lead to a more challenging search process on top-performing models. In contrast, Graph Isomorphism judiciously incorporates isomorphic graph transformation into each independent DAG in the meta-graph, yielding an even more significant prediction quality improvement on large-scale *ImageNet Dense Connectivity Space* over small-scale *CIFAR-10 Dense Connectivity Space*.

Table 4. Evaluation of prediction quality with/without Graph Isomorphism (GI).

Dataset	GI?	Pearson's ρ	Kendall's τ	Mean-Squared Error
ImageNet		0.342	0.215	5×10^{-4}
	✓	0.972	0.904	4×10^{-5}
CIFAR-10		0.581	0.428	4×10^{-4}
	✓	0.946	0.874	6×10^{-5}

7. Conclusion

We propose CSCO, a novel paradigm that allows flexible exploration of the dense connectivity of building operators and innovates building cells in CNN architectures. CSCO aims to seek the optimal building cells of CNN architectures represented by Directed Acyclic Graphs (DAGs), containing rich sources of dense connectivity of versatile building operators to cover CNN architecture designs flexibly. CSCO crafts a dense connectivity space to fabricate the building cells of the CNN architectures and further leverages a performance predictor to obtain the best dense connectivity. To enhance the reliability and quality of prediction, we propose Graph Isomorphism as data augmentation to boost sample efficiency and Metropolis-Hastings Evolutionary Search (MH-ES) to efficiently explore dense connectivity space and evade locally optimal solutions in CSCO. Experimental on ImageNet demonstrates $\sim$ 0.6% accuracy gain over other NAS-crafted dense connectivity designs under mobile computation regime.

Acknowledgement. This project is partly supported by NSF 2112562, ARO W911NF-23-2-0224, NSF CAREER-2305491, and NSF 2148253.

References

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware.

In *Proceedings of the International Conference on Learning Representations*, 2019. 2, 7

[2] Alain Chédotal and Linda J Richards. Wiring the brain: the biology of neuronal guidance. *Cold Spring Harbor perspectives in biology*, 2(6):a001917, 2010. 1

[3] Hsin-Pai Cheng, Tunhou Zhang, Yixing Zhang, Shiyu Li, Feng Liang, Feng Yan, Meng Li, Vikas Chandra, Hai Li, and Yiran Chen. Nasgem: Neural architecture search via graph embedding method. In *Proceedings of the Thirty-Fifth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, 2021. 2

[4] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16276–16285, 2021. 2

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. 2

[6] Tom Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter AN Bosman. Local search is a remarkably strong baseline for neural architecture search. In *Evolutionary Multi-Criterion Optimization: 11th International Conference, EMO 2021, Shenzhen, China, March 28–31, 2021, Proceedings 11*, pages 465–479. Springer, 2021. 1

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[8] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 7

[9] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 6

[10] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems*, 33:10480–10490, 2020. 2

[11] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10628–10637, 2020. 2, 7

[12] Zachary Friggstad, Mohsen Rezapour, and Mohammad R Salavatipour. Local search yields a ptas for k-means in doubling metrics. *SIAM Journal on Computing*, 48(2):452–480, 2019. 6

[13] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984. 7

[14] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, pages 69–93. Elsevier, 1991. 6

[15] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 4

[17] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. In *IEEE Signal processing magazine*, 2012. 7

[18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 1, 2

[19] 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. In *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 1, 6

[21] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2752–2761, 2018. 1, 2, 7

[22] Tao Huang, Shan You, Yibo Yang, Zhuozhuo Tu, Fei Wang, Chen Qian, and Changshui Zhang. Explicitly learning topology for differentiable neural architecture search. *arXiv preprint arXiv:2011.09300*, 2020. 1, 7

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[24] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018. 1

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. 1

[26] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. *arXiv e-prints*, pages arXiv–2004, 2020. 7

[27] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+:

Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 1

[28] 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. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 6

[29] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations*, 2019. 1, 2, 6, 7

[30] Yuqiao Liu, Yehui Tang, and Yanan Sun. Homogeneous architecture augmentation for neural predictor. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12249–12258, 2021. 2

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2017. 7

[32] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. 5

[33] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018. 1, 2

[34] Martin Pincus. Letter to the editor—a monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18(6):1225–1228, 1970. 2, 6

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4780–4789, 2019. 2, 6

[36] Robin Ru, Pedro Esperanca, and Fabio Maria Carlucci. Neural architecture generator optimization. *Advances in Neural Information Processing Systems*, 33:12057–12069, 2020. 2

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 2, 4, 7

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4278–4284. AAAI Press, 2017. 2, 7

[40] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019. 1

[41] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1, 7

[42] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer, 2020. 2

[43] Colin White, Sam Nolen, and Yash Savani. Local search is state of the art for nas benchmarks. *arXiv preprint arXiv:2005.02960*, page 76, 2020. 1

[44] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, pages 10293–10301, 2021. 1

[45] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *arXiv preprint arXiv:1906.00586*, 2019. 1, 2

[46] Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Stronger nas with weaker predictors. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[47] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 7

[48] Sirui Xie, Shoukang Hu, Xinjiang Wang, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Understanding the wiring evolution in differentiable neural architecture search. In *International Conference on Artificial Intelligence and Statistics*, pages 874–882. PMLR, 2021. 2

[49] 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 (ICLR)*, 2020. 7

[50] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 6

[51] Tunhou Zhang, Hsin-Pai Cheng, Zhenwen Li, Feng Yan, Chengyu Huang, Hai Li, and Yiran Chen. Autoshrink: A topology-aware nas for discovering efficient neural architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2019. 2

[52] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 7