The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

Neural Architecture Search as Sparse Supernet

Yan Wu[1*], Aoming Liu[1*], Zhiwu Huang[1], Siwei Zhang[1], Luc Van Gool[1,2]
[1]Computer Vision Lab, ETH Zürich, Switzerland
[2]VISICS, KU Leuven, Belgium
{wuyan, aoliu}@student.ethz.ch, siwei.zhang@inf.ethz.ch, {zhiwu.huang, vangool}@vision.ee.ethz.ch

Abstract

This paper aims at enlarging the problem of Neural Architecture Search (NAS) from Single-Path and Multi-Path Search to automated Mixed-Path Search. In particular, we model the NAS problem as a sparse supernet using a new continuous architecture representation with a mixture of sparsity constraints. The sparse supernet enables us to automatically achieve sparsely-mixed paths upon a compact set of nodes. To optimize the proposed sparse supernet, we exploit a hierarchical accelerated proximal gradient algorithm within a bi-level optimization framework. Extensive experiments on Convolutional Neural Network and Recurrent Neural Network search demonstrate that the proposed method is capable of searching for compact, general and powerful neural architectures.

Introduction

While deep learning has proven its superiority over manual feature engineering, most of the conventional neural network architectures are still handcrafted by experts in a tedious and ad hoc fashion. Neural Architecture Search (NAS) has been suggested as the path forward for alleviating the network engineering pain by automatically optimizing architectures. The automatically searched architectures perform competitively in computer vision tasks such as image classification (Zoph and Le 2018; Liu et al. 2018b; Zoph et al. 2018; Liu, Simonyan, and Yang 2018; Real et al. 2019; Chen et al. 2019c; Luo et al. 2018; Cai, Zhu, and Han 2019; Wu et al. 2019; Zheng et al. 2019; You et al. 2020), object detection (Zoph et al. 2018), semantic segmentation (Liu et al. 2019; Chen et al. 2019a) and image generation (Gong et al. 2019; Tian et al. 2020).

As one of the most popular NAS families, one-shot NAS generally models the architecture search problem as a one-shot training process of an over-parameterized supernet that comprises candidate architectures (paths). From the supernet, either Single-Path or Multi-Path architecture can be derived. However, both the existing Single-Path and Multi-Path Search works typically require a predefined structure on the searched architectures. For the Single-Path Search, some works like (Liu, Simonyan, and Yang 2018; Liu et al. 2018a) search for a computation cell as the backbone block of the final architecture. Based on the modeling of directed acyclic graph, the cell comprises a set of nodes, each of which corresponds to a feature map, as well as their associated edges that represent single operations such as convolution and max-pooling. The resulting neural networks are limited to Single-Path architectures where each intermediate feature map is processed by a single operation. On the other hand, while Multi-Path architecture search methods like (Chu et al. 2020a) search for a more flexible architecture with multiple paths between nodes, they generally require to fix the number of paths in advance. Moreover, both the existing Single-Path and Multi-Path Search methods have to manually fix the node number, which is another strong constraint for architecture search.

In this paper, we target for a more automated NAS which can automatically optimize the mixture of paths as well as a changeable set of nodes. In other words, the target of automated Mixed-Path Architecture Search is to reduce unnecessary constraints on the structure of the searched architecture so as to explore in a more broad and general search space. For this purpose, we model the automated NAS problem as a one-shot searching process of a sparse supernet, which consists of sparsely-mixed paths and nodes without loss of network power. In particular, we exploit a new continuous architecture representation using Sparse Group Lasso to achieve the sparse supernet. As a result, the supernet is not only able to produce diverse mixed-paths between different pairs of nodes, but also automatically removes some useless nodes. The more general modeling however makes the optimization much more challenging due to the complex bi-level optimization with the non-differentiable sparsity constraint, which cannot be optimized by traditional network optimization algorithms well. To address this challenging issue, we propose a hierarchical accelerated proximal gradient algorithm that is capable of addressing the mixed sparsity under the bi-level optimization framework. In summary, this paper brings several innovations to the domain of NAS as follows:

- We suggest the new problem of Mixed-Path Neural Architecture Search where the node and path structure of the cell are automatically derived.
- We model the problem as a sparse supernet using a new continuous architecture representation with a mixture of sparsity constraints.

[*] Equal contribution

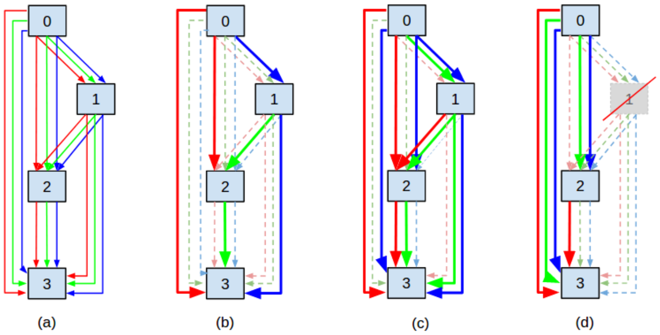


Figure 1: Overview of various architecture search tasks over (a) Supernet that represents the continuous relaxation of the search space: (b) Single-Path Architecture Search that optimizes architectures with one single operation between each node pair, (c) Multi-Path Architecture Search that searches for multiple paths with a fixed amount between two nodes, and (d) the proposed Mixed-Path Architecture Search that has no rigid constraint on the path and node structure.

- We propose a hierarchical accelerated proximal gradient algorithm to optimize the supernet search with the mixed sparsity constraints.
- We study that the searched Mixed-Path architectures are compact, general and powerful for some standard NAS benchmarks.

Problem Statement

Neural Architecture Search aims at searching for computation cells as the building block of the final architecture. In general, each achitecture cell can be formulated as a directed acyclic graph (DAG) as shown in Fig.1 (a), where each node represents a feature map in neural networks, and each directed edge is a mixture of operations that transform the tail node to the head node. As a consequence, the output of each intermediate node is a sum of incoming feature maps from predecessors. The DAG modeling enables the training of an over-parameterized supernet that stacks multiple basic cells. The optimal architecture is then derived from the supernet with the following three strategies.

Single-Path Architecture Search methods generally search for one single operation between each node pair with fixed edge amounts in each cell as shown in Fig.1 (b). To select a single path from a given supernet, they commonly first optimize the mixture of all associated operations, and finally choose those with the highest contribution to the supernet. As one of the most representative Single-Path Search methods, Differentiable Architecture Search (DARTS) (Liu, Simonyan, and Yang 2018) optimizes the mixture of operations within supernet using softmax combination. For each edge, the operation with largest softmax weight was selected. For each node, two input edges was selected by comparing each edge's largest operation weight. The rigid requirement on final architecture highly reduces the search space, such that the optimization process has a high potential to be stuck at a local minimum.

Multi-Path Architecture Search searches for multiple paths between any pair of nodes (Fig.1 (c)), which is inspired by Multi-Path feature aggregations such as Inception networks (Szegedy et al. 2015) and ResNeXt (Xie et al. 2017). Nevertheless, the Multi-Path Search approach, such as (Chu et al. 2020b,a) typically still requires a strong prior knowledge about the aggregation intensity (i.e., path number) in advance of search. Furthermore, enforcing the same number of operations for each pair of nodes is very likely to reach a locally optimal architecture.

Mixed-Path Architecture Search is hence proposed for the exploration of a more general search space to avoid the human intervene as much as possible. For this purpose, we suggest to enlarge the search space of NAS by relaxing the constraints on the network structure. In particular, the supernet is merely required to learn a complete and compact neural architecture, without any more rigid constraints on the node and path structure. In other words, it should be trained to automatically derive an optimal node and path structure as compact as possible without loss of classification or regression ability. The suggested new problem is conceptually illustrated in Fig.1 (d). In comparison to Single-Path Search and Multi-Path Search, Mixed-Path Search dramatically increases the architecture search space, leading to a much more challenging NAS problem.

Related Work

For **Neural Architecture Search**, early one-shot models generally aim for **Single-Path architecture search** (Liu, Simonyan, and Yang 2018; Cai, Zhu, and Han 2019; Wu et al. 2019; Liang et al. 2019; Xie et al. 2019; Chen et al. 2019b; Li et al. 2020a). For instance, DARTS (Liu, Simonyan, and Yang 2018) introduces a continuous relaxation of the discrete search space by aggregating candidate paths with softmax weights, so that a differentiable Single-Path Architecture Search can be performed. Based on DARTS, improvements including progressive search (Chen et al. 2019b) and early stopping (Liang et al. 2019) are proposed. In addition, ProxylessNAS (Cai, Zhu, and Han 2019) and FBNet (Wu et al. 2019) perform Single-Path Architecture Search with single-path sampling.

Multi-Path Architecture Search problem is proposed and addressed by some recent works. For example, MixPath (Chu et al. 2020a) activates m paths each time and a Shadow Batch Normalization is proposed to stabilize the training. GreedyNAS (You et al. 2020) is also Multi-Path architecture search with activating multiple paths. CoNAS (Cho, Soltani, and Hegde 2019) achieves Multi-Path Architecture Search by sampling sub-graph from a pre-trained one-shot model and doing Fourier analysis based on sub-graphs' performances. Multiple paths are selected based on the coefficients. Path amount of Fourier basis is controlled by its degree d. FairDARTS (Chu et al. 2020b) uses sigmoid instead of softmax to eliminate the unfair optimization and allows multi paths, but it limits the maximum 2 paths between nodes. Generally, Multi-Path architecture search is still limited to search for a fixed number of paths.

Few works approach the variants of our defined **Mixed-Path Architecture Search**. For instance, DSO-NAS (Zhang et al. 2020) enforces the $\ell 1$-norm sparsity constraint (i.e., Lasso) to individual architecture parameters, which can achieve sparsely-mixed paths but overlooks the quest for sparse node structures especially when nodes are redundant initially (e.g., DARTS' cell structure). BayseNAS (Zhou et al. 2019) exploits either $\ell 1$-norm sparsity (with the same drawback with (Zhang et al. 2020)) or group-level sparsity with a weighted Group Lasso constraint in the classic Bayesian leaning manner, leading to sparse node structures. However, their Group Lasso constraint focuses on the sparsity on groups (i.e., nodes), and theoretically it may reach unsatisfactory sparsity on elements (i.e., paths). As a concurrent work, Gold-NAS (Bi et al. 2020) suggests to gradually prune individual paths using one-level optimization to approach a mixed-path structure. By comparison, our Mixed-Path Architecture Search problem aims at both node and path structures. To this end, we model the Mixed-Path Architecture Search as a supernet with the Sparse Group Lasso constraint, which enables us to go for a more compact structure of nodes and paths. This enables our work to serve as a valuable pioneer study for such a more general Mixed-Path Architecture Search problem.

Sparsity Constraints, including Lasso (Tibshirani 1996), Group Lasso (Yuan and Lin 2006), Sparse Group Lasso (Simon et al. 2013), *etc.* have been widely applied to areas such as statistics, machine learning and deep learning. A close application of sparsity constraints to NAS is the network pruning task which targets for reducing the model complexity by removing redundant network weights, neurons, layers, *etc.*. Some network pruning works (Huang and Wang 2017; Scardapane et al. 2017; Li et al. 2020b; Ye et al. 2018; Liu et al. 2017; Wen et al. 2016; Alvarez and Salzmann 2016) proposed to impose sparsity constraints on network weights or auxiliary scale factors so as to sparsify the networks. In particular, (Scardapane et al. 2017) applies the same Sparse Group Lasso constraint to remove network neurons and weights. There are two major differences from our work: 1) The search space of (Scardapane et al. 2017) is fundamentally different from NAS. (Scardapane et al. 2017) imposes SGL constraint on the filter weights and focuses on pruning the neurons and weights, while ours focuses on pruning the connections between different layers, namely structural connections, and the sparsity constraint is imposed on architecture weights which yields a more challenging sparsity constrained problem under the bi-level optimization framework. 2) (Scardapane et al. 2017) merely adopts the traditional Adam to optimize the one-level Sparse Group Lasso constrained optimization problem. It is known that stochastic gradient descent algorithms, such as SGD and Adam are not proper to address non-differentiable optimization problem. Some works like (Simon et al. 2013; Ida, Fujiwara, and Kashima 2019) have learned blockwise descent algorithms to optimize Sparse Group Lasso, but they cannot be trivially applied to the stochastic optimization framework. To address this, we propose a hierarchical proximal bi-level optimization algorithm.

Sparse Supernet

Our Mixed-Path Architecture Search starts from an over-parameterized supernet, and aims at deriving an compact and optimal neural architecture. With the target of automatically selecting useful operations and nodes within the supernet, we are inspired by the prevailing sparsity regularization which can act as an automated feature selection mechanism. We thereby consider to introduce a sparse constraint to our supernet to select meaningful intermediate feature maps automatically. With the imposed sparsity constraint, we enable an automated sparse Mixed-Path Architecture Search.

The supernet is designed as a stack of repetitive cells, and each cell is formulated as a DAG cell as shown in Fig.1 (a). In particular, the mixture of operations on each edge is formulated in a "regression-like" way. Instead of employing the widely-used softmax combination and its variants, such as Gumbel softmax (Chang et al. 2019), we formulate the edge e_{ij} between node x_i and x_j as a linear combination of operations, and the feature map derived from each operation $o \in O$ is scaled by a weight factor A^o_{ij}. The output feature map of intermediate node is now a scaled linear combination of various feature maps from different predecessors with their associated operations, i.e.,

$$x_j = \sum_{i<j} \sum_{o \in O} A^o_{ij} o(x_i), A^o_{ij} \in \mathbb{R}^1 \tag{1}$$

To relax the structure constraints on both the number of nodes and paths per edge, we aim at achieving the operation sparsity as well as the node sparsity. Sparse Group Lasso (SGL) regularization (Simon et al. 2013) meets our expectation exactly, which allows for both element (operation) sparsity and group (node) sparsity. In a DAG with N intermediate nodes, for each node x_j, we group weight factors for all incoming feature maps A^o_{ij}, where $i < j, o \in O$ as $A_{(j)}$. Mathematically, the full objective function is derived as:

$$L(w, A) = l(w, A) + \Omega_{SGL}(A) \tag{2}$$

$$= l(w, A) + \lambda\alpha ||A||_1 + \lambda(1-\alpha) \sum_{n=1}^{N} \sqrt{|A_{(n)}|} \cdot ||A_{(n)}||_2 \tag{3}$$

Algorithm 1: Bi-level Optimization with the Proposed Hierarchical Accelerated Proximal Gradient (HAPG) Algorithm

```
Require: Supernet parameterized by $w$ and $A$ with $A^o_{i,j}$ being applied for each operation $o$ between nodes $i, j$;
while not converged do
    Step1: Update architecture weights A with HAPG given in Eq.6-10. Note that the gradient are computed with the
      second order approximation given in Eq.13.
    Step2: Update $w$ by descending $\nabla_w l_{train}(w, A)$;
end
Ensure: Sparse supernet based on sparse architecture weights A.
```

where Ω_{SGL} corresponds to the mixed sparsity regularization, λ is the sparsity strength factor, and α controls the balance between operation sparsity and node sparsity. By optimizing the network parameters w and the architecture weights A with the target function in Eq.3, we can achieve a sparse supernet structure. Ideally, the final architecture will be derived by removing the operations with zero weights and the nodes with all zero incoming weights.

To jointly optimize the supernet and learn a sparse network structure, we target for solving the following bi-level optimization problem:

$$\begin{aligned} \min_A \quad & l_{val}(w^*(A), A) + \Omega_{SGL}(A) \\ s.t. \quad & w^*(A) = \text{argmin}_w \ l_{train}(w, A) \end{aligned} \tag{4}$$

where the network weights w and the architecture weights A are optimized on two separate training and validation sets to avoid architecture from overfitting to data.

Optimization

As ℓ_1-norm term is convex but non-differentiable, the SGL regularization term yields a challenging optimization problem. Conventional stochastic gradient descent algorithms, such as SGD and Adam generally cannot work well. While some exiting works like (Simon et al. 2013; Ida, Fujiwara, and Kashima 2019) have exploited blockwise descent algorithms to fit SGL, it is non-trivial to apply their algorithms to the stochastic optimization setting. We thereby turn to the proximal methods (Bach et al. 2012) which is capable of solving the optimization problem with the non-differentiable term and enables us to learn some exact zero weights via soft-threshold. We propose a Hierarchical Accelerated Proximal Gradient algorithm (**HAPG**) and its improved version (**AdamHAPG**), both of which are suitable for stochastic optimization. Finally, we further appropriately incorporate these two methods into the bi-level optimization framework.

Hierarchical Proximal Optimization

Computing the proximal operator $\text{Prox}_\Omega(\cdot)$ of the regularization term Ω is a key part of proximal optimization algorithms. The joint combination of ℓ_1 and ℓ_1/ℓ_2 norm in SGL brings much higher complexity to the direct proximal operator computing. Inspired by (Bach et al. 2012) that the SGL norm is a special case of hierarchical norm (Zhao et al. 2009), with ℓ_1-norm of each individual weight factor being a child group of the ℓ_1/ℓ_2-norm, we derive the hierarchical proximal operator as a composition of ℓ_1-norm and ℓ_1/ℓ_2-norm proximal operators:

$$\text{Prox}_\Omega(\cdot) = \text{Prox}_{\lambda(1-\alpha)||\cdot||_2} \circ \text{Prox}_{\lambda\alpha||\cdot||_1}(\cdot) \tag{5}$$

As for proximal algorithms, widely-used methods include ISTA and FISTA (Beck and Teboulle 2009), and here we employ an efficiently reformulated Accelerated Proximal Gradient (APG) optimization scheme (Huang and Wang 2017) which allows for the stochastic optimization setting. Accordingly, we propose a Hierarchical Accelerated Proximal Gradient (**HAPG**) algorithm tailored for the Sparse Group Lasso regularization:

$$z_t = A_{t-1} - \eta_t g_{t-1} \tag{6}$$

$$\begin{aligned} v_t = & \text{Prox}_{\eta_t\lambda(1-\alpha)||\cdot||_2} \circ \text{Prox}_{\eta_t\lambda\alpha||\cdot||_1}(z_t) \\ & - A_{t-1} + u_{t-1}v_{t-1} \end{aligned} \tag{7}$$

$$A_t = \text{Prox}_{\eta_t\lambda(1-\alpha)||\cdot||_2} \circ \text{Prox}_{\eta_t\lambda\alpha||\cdot||_1}(z_t) + u_t v_t \tag{8}$$

where g_{t-1} represents the gradient, η_t is the gradient step size and $u_t = \frac{t-2}{t+1}$. And the proximal operators can be derived as:

$$[\text{Prox}_{\eta_t\lambda\alpha||\cdot||_1}(\mathbf{z})]_i = \text{sgn}(z_i)(|z_i| - \eta_t\lambda\alpha)_+ \tag{9}$$

$$[\text{Prox}_{\eta_t\lambda(1-\alpha)||\cdot||_2}(\mathbf{z})]_n = \left(1 - \frac{\sqrt{|\mathbf{z}_{(\mathbf{n})}|}\eta_t\lambda(1-\alpha)}{||\mathbf{z}_{(\mathbf{n})}||_2}\right)_+ \mathbf{z}_{(\mathbf{n})} \tag{10}$$

To further facilitate the optimization, we introduce the powerful Adam into the proposed HAPG (**AdamHAPG**) and replace the gradient descent in Eq.6 with an Adam gradient update (Kingma and Ba 2015). We should note that each weight factor gets an individual gradient step size in AdamHAPG, and we thereby make small adaptations when computing proximal operators. As for the ℓ_1-norm proximal operator, we implement the proximal update using the corresponding step size for each weight, while for the ℓ_1/ℓ_2-norm proximal operator, we heuristically take the median value of step sizes for each group as an approximation and we experimentally show that it works properly for our problem.

Bi-level Optimization with Hierarchical Proximal Optimization

We incorporate our proposed hierarchical proximal algorithms into the bi-level optimization framework (Liu, Simonyan, and Yang 2018) to alternatively optimize the network parameter ω and the architecture weight A. In particular, we follow (Liu, Simonyan, and Yang 2018) to compute

Architecture	Test Error (%)		Params	Search Cost	Architecture
	C10	C100	(M)	(GPU days)	Type
DenseNet-BC (Huang, Liu, and Weinberger 2016)	3.46	17.18	25.6	–	manual
DARTS (first order) (Liu, Simonyan, and Yang 2018)	3.00 ± 0.14	17.76	3.3	1.5	Single-Path
DARTS (second order) (Liu, Simonyan, and Yang 2018)	2.76 ± 0.09	17.54	3.3	4	Single-Path
P-DARTS (Chen et al. 2019b)	2.50	16.55	3.4	0.3	Single-Path
PC-DARTS (Xu et al. 2020)	2.57 ± 0.07	–	3.6	0.1	Single-Path
FairDARTS (Chu et al. 2020b)	2.54 ± 0.05	–	3.32 ± 0.46	0.5	Multi-Path
CoNAS (Cho, Soltani, and Hegde 2019)	2.62 ± 0.06	–	4.8	0.7	Multi-Path
DSO-NAS (Zhang et al. 2020)	2.84 ± 0.07	–	3.0	1	Mixed-Path
BayesNAS (Zhou et al. 2019)	2.81 ± 0.04	–	3.4	0.2	Mixed-Path
SparseNAS + HAPG	2.73 ± 0.05	16.83	3.8	1	Mixed-Path
SparseNAS + AdamHAPG	2.69 ± 0.03	17.04	4.2	1	Mixed-Path
SparseNAS + AdamHAPG*	2.50	16.79	3.5	0.27	Mixed-Path

* Obtained by searching in a cherry-picked search space.

Table 1: Performance Comparison on CIFAR-10 and the Transferability to CIFAR-100 (lower error rate is better).

Architecture	Test Error (%)		Params	Architecture
	top-1	top-5	(M)	Type
MobileNet (Howard et al. 2017)	29.40	10.5	4.2	manual
DARTS (second order) (Liu, Simonyan, and Yang 2018)	26.70	8.7	4.7	Single-Path
FairDARTS-B (Chu et al. 2020b)	24.90	7.5	4.8	Multi-Path
DSO-NAS (Zhang et al. 2020)	26.20	8.6	4.7	Mixed-Path
BayesNAS (Zhou et al. 2019)	26.50	8.9	3.9	Mixed-Path
SparseNAS + HAPG	25.48	8.1	5.3	Mixed-Path
SparseNAS + AdamHAPG	24.67	7.6	5.7	Mixed-Path

Table 2: Transferability Comparison on ImageNet in the Mobile Setting (lower error rate is better)

the gradient of the architecture weights (i.e., g_{t-1} in Eq.6):

$$\begin{aligned} g_t &= \nabla_A l_{val}(w^*(A_t), A_t) \qquad (11) \\ &\approx \nabla_A l_{val}(w'_t, A_t) \\ &\quad - \gamma \nabla^2_{A,w} l_{train}(w_t, A_t) \nabla_{w'} l_{val}(w'_t, A_t) \qquad (12) \\ &\approx \nabla_A l_{val}(w'_t, A_t) \\ &\quad - \frac{\nabla_A l_{train}(w_t^+, A_t) - \nabla_A l_{train}(w_t^-, A_t)}{2\epsilon} \qquad (13) \end{aligned}$$

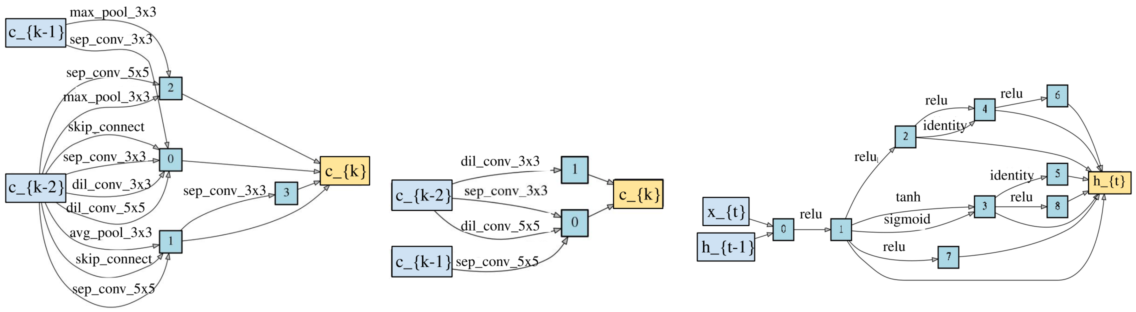
where $w'_t = w_t - \gamma \nabla_w l_{train}(w_t, A_t)$, $w_t^{\pm} = w_t \pm \epsilon \nabla_w l_{train}(w'_t, A_t)$ and ϵ and γ are set to be small scalars as done in (Liu, Simonyan, and Yang 2018). Eq.12 is derived by a one-step forward approximation, i.e., $w^*(A_t) \approx w'_t = w_t - \gamma \nabla_w l_{train}(w_t, A_t)$, and Eq.13 follows the second-order approximation in (Liu, Simonyan, and Yang 2018). Especially, when introducing the HAPG and AdamHAPG to the bi-level optimization framework, to stabilize the training, we follow (Simon et al. 2013) to adopt a similar pathwise solution for an incremental increase of regularization factor λ, and we experimentally show the effectiveness of this progressive sparsifying solution. The complete optimization algorithm is presented in Alg.1.

Evaluation

We evaluate the proposed SparseNAS for Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) architecture search on CIFAR-10 and Penn Treebank (PTB) respectively, and further investigate the transferability of searched architectures on CIFAR-10 to CIFAR-100 and ImageNet. In both CNN and RNN cell search experiments, we follow the setup of DARTS (Liu, Simonyan, and Yang 2018) to implement SparseNAS, where we use the same search space, cell setup and we stack the same number of cells for fair comparison. While there exist many direct improvements over DARTS like progressive search (Chen et al. 2019b) and early stopping (Liang et al. 2019), most of them make orthogonal contributions to our work and thus can be used to improve our method as well. However, further applying them to our method is beyond the scope of our paper. Hence, we mainly take the most related Single-Path (DARTS (Liu, Simonyan, and Yang 2018)), Multi-Path (FairDARTS (Chu et al. 2020b), CoNAS (Cho, Soltani, and Hegde 2019)) and Mixed-Path (DSO-NAS (Zhang et al. 2020), BayesNAS (Zhou et al. 2019)) methods as our real competitors. Note that the reported results of all the competitors are from their original papers. For more detailed ex-

Architecture	Perplexity		Params	Search Cost	Architecture
	valid	test	(M)	(GPU days)	Type
LSTM (Merity, Keskar, and Socher 2018)	60.70	58.80	24	–	manual
DARTS (first order) (Liu, Simonyan, and Yang 2018)	60.20	57.60	23	0.5	Single-Path
DARTS (second order) (Liu, Simonyan, and Yang 2018)	58.10	55.70	23	1	Single-Path
CoNAS (Cho, Soltani, and Hegde 2019)	59.10	56.80	23	0.25	Multi-Path
SparseNAS + AdamHAPG	57.73	55.37	23	0.25	Mixed-Path

Table 3: Performance Comparison on PTB (lower error rate is better)



(a) Normal Convolutional Cell (b) Reduction Convolutional Cell (c) Recurrent Cell

Figure 2: Architectures searched on CIFAR-10 and PTB

periments setup, please refer to the supplementary material[*].

Convolutional Neural Architecture Search

The convolutional cell is directly searched on CIFAR-10 and transferred to CIFAR-100 and ImageNet.

On CIFAR-10, HAPG method obtains architecture with error 2.73 ± 0.05. Due to the use of the adaptive learning rate, AdamHAPG (2.69 ± 0.03) performs better. Both the HAPG and AdamHAPG based SparseNAS outperform the second-order DARTS (2.76 ± 0.09) and all Mixed-Path competitors. In particular, a search performed on a cherry-picked search space achieves the best performance (2.50) among all competitors. The results are shown in Table 1.

The transferability results from CIFAR-10 to CIFAR-100 and ImageNet are shown in Table 1 and Table 2 respectively. Note that some competitors like CoNAS did not report their transferability results from CIFAR-10 to CIFAR-100 and ImageNet. The reported results show that our AdamHAPG performs the best (even better than the recent FairDARTS on ImageNet), and both the HAPG and AdamHAPG based SparseNAS outperform the other competitors.

Note that we search for a sparse model structure, but not necessarily a model with small model size, and our method show a clear advantage in structural sparsity. We present the normal and reduction cell searched on CIFAR-10 in Figure 2(a) and (b). Notably, we get compact reduction cells with only 2 remained nodes, which proves our advantage in group-level sparsity compared with DOS-NAS (Zhang et al. 2020). Compared with another Mixed-Path work BayesNAS (Zhou et al. 2019), we have fewer paths in normal cells and reduction cells and thus we achieve better element-level sparsity and even higher performance. Compared with the competitors, our searched architectures show more general properties with more diverse path and node structures. In addition, there is no obvious ”collapse” problem (e.g. excessive skip-connection selected) problem observed.

Reccurent Neural Architecture Search

Recurrent cell search is performed on the PTB dataset, and we follow the experiment setup of (Liu, Simonyan, and Yang 2018). When optimizing with HAPG, we observe that the search phase does not converge with an exploded gradient of architecture weights which is a typical RNN training problem. The AdamHAPG with adaptive learning rate is shown to be helpful to stabilize the training for RNN model search. The results are summarized in Table 3. Except for DARTS and CoNAS, the other competitors did not report their results on PTB. To our best knowledge, the derived architecture by our method obtains a new state-of-the-art NAS on PTB with valid perplexity 57.73 and test perplexity 55.37.

In Figure 2(c), we present the derived recurrent cell on PTB. Compared to DARTS and CoNAS where each node only receives a fixed number of incoming edges, the cell derived by SparseNAS is more general that intermediate nodes can have different numbers of incoming edges. And this general architecture has shown its superior performance.

Ablation Study

HAPG/AdamHAPG vs. SGD/Adam We study the advantage of our HAPG and AdamHAPG over conventional SGD and Adam in the sparsity constrained optimization problem.

[*] https://arxiv.org/pdf/2101.06658.pdf

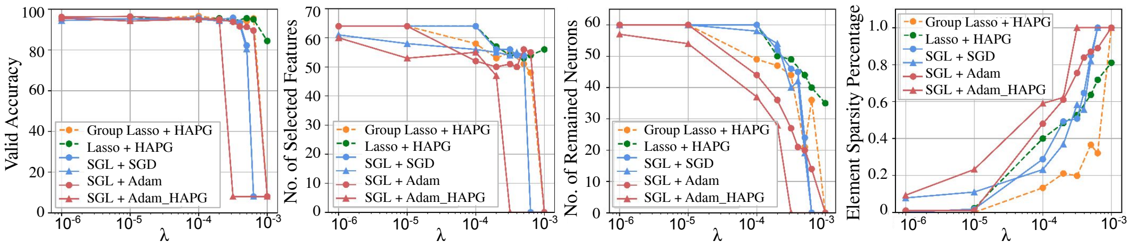


Figure 3: Comparison between Lasso, Group Lasso and Sparse Group Lasso, and comparison between different optimization methods, including SGD, Adam, HAPG and AdamHAPG. From left to right: the valid accuracy of standalone sparse network, the number of selected input features, the number of remained inner neurons, the total sparsity percentage of network weights. With comparable valid accuracy, the HAPG and AdamHAPG with Sparse Group Lasso give better element and group sparsity.

Following (Scardapane et al. 2017) that applies SGL constraint to network weights to select features and remove redundant weights and neurons, we conduct various experiments in this easier network pruning task, which is essentially a one-level sparsity constrained optimization problem, to purely evaluate the effectiveness and advantage of our proposed optimization algorithms. Starting with a fully connected network with two hidden layers (40 and 20 hidden neurons respectively), we implement classification task on DIGITS (Alimoglu and Alpaydin 1996). The 8×8 images are flatten into 64-dim vectors as the input features.

The performances of these four methods are presented in Fig.3. From left to right, Fig.3 shows the valid accuracy of stand-alone sparse network, number of selected features, number of remained inner neurons and the element sparsity percentage of network weights. The horizontal axis indicates the different sparsity constraint factor λ. As λ increasing, stronger sparsity regularization derives sparser network. With λ ranging from 10^{-5} to $10^{-3.7}$, stand-alone architectures are all well-performed with comparable performances. Whereas, in terms of sparsity, our proposed HAPG and AdamHAPG clearly outperform their counterparts Adam and SGD. In particular, AdamHAPG shows a clear superiority to have a more compact structure and more powerful feature selection without loss in the classification accuracy.

SGL vs. Lasso and Group Lasso (GL) With the same optimization method HAPG, 3 experiments with Lasso, GL, SGL constraints are conducted respectively. Fig.3 shows that enforcing SGL is clearly better than using Lasso and GL both in group (features and neurons) and element (network weights) sparsity, while having comparable valid accuracies.

Effect of $\boldsymbol{\alpha}$ In Table 4, we study the effect of different α on CNN task. Theoretically, α controls the balance between path sparsity and node sparsity. With α decreasing, the algorithm tend to obtain a more node-sparse architecture. Empirically, in CNN task, with AdamHAPG, the $\alpha = 0.5$ obtains the optimal architecture with highest accuracy.

Effect of $\boldsymbol{\lambda}$ λ is another hyperparameter to control the sparsity strength. Since we progressively increase the sparsity strength during search via a linearly increased λ, we study the effect of λ increasing step to the training stability of search process. We plot the evolution of valid accuracy during CNN cell search phase with increasing λ step 0.01, 0.02 and 0.03 respectively in Fig. 4. With larger λ, we expect a better sparsity, but it is likely that architecture weights fluctuate heavily at each updating step, and this can explain that with step size 0.03, we see a large fluctuation in valid accuracy after 15 epochs. And small step sizes 0.01, 0.02 lead to a more stable training. As a trade-off between architecture sparsity level and stability of searching process, we typically choose a value like 0.01 as the step size in our experiments.

	Test Error (%)
AdamHAPG(α=0.3)	3.30
AdamHAPG(α=0.5)	2.69
AdamHAPG(α=0.7)	2.79

Table 4: Study on the effect of α on CIFAR-10

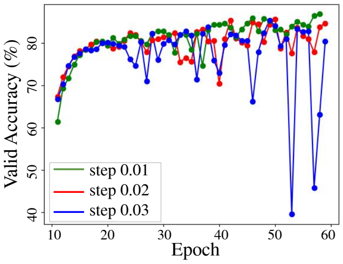


Figure 4: Valid accuracy during search with different λ steps

Conclusion

In this work, we launch Neural Architecture Search to explore in a more general and flexible Mixed-Path Search space using a sparse supernet. Starting from a supernet parameterized by architecture weight factors, we exploit the Sparse Group Lasso regularization on weight factors to automatically search for optimal structures of nodes and paths. To address the challenging optimization problem with non-differentiable sparsity constraint, we propose novel hierarchical proximal algorithms and incorporate them into a bi-level optimization framework. We experimentally show very competitive results and potentials of our derived Mixed-Path architectures on various datasets. We believe that our general Mixed-Path Search modeling will lead the future NAS research to a much broader search space and bring the possibility to derive more flexible and powerful architectures.

Acknowledgements

This work was supported by the ETH Zürich Fund (OK), an Amazon AWS grant, and an Nvidia GPU grant.

References

Alimoglu, F.; and Alpaydin, E. 1996. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN)*. Citeseer.

Alvarez, J. M.; and Salzmann, M. 2016. Learning the Number of Neurons in Deep Networks. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*, 2270–2278. Curran Associates, Inc. URL http://papers.nips.cc/paper/6372-learning-the-number-of-neurons-in-deep-networks.pdf.

Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning* 4(1): 44–49.

Beck, A.; and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2(1): 183–202.

Bi, K.; Xie, L.; Chen, X.; Wei, L.; and Tian, Q. 2020. Gold-nas: Gradual, one-level, differentiable. *arXiv preprint arXiv:2007.03331* .

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*. URL https://arxiv.org/pdf/1812.00332.pdf.

Chang, J.; Guo, Y.; MENG, G.; XIANG, S.; Pan, C.; et al. 2019. DATA: Differentiable ArchiTecture Approximation. In *Advances in Neural Information Processing Systems*, 874–884.

Chen, W.; Gong, X.; Liu, X.; Zhang, Q.; Yuan, L.; and Wang, Z. 2019a. FasterSeg: Searching for Faster Real-time Semantic Segmentation. In *International Conference on Learning Representations (ICLR)*.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019b. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, 1294–1303.

Chen, Y.; Meng, G.; Zhang, Q.; Xiang, S.; Huang, C.; Mu, L.; and Wang, X. 2019c. Renas: Reinforced evolutionary neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4787–4796.

Cho, M.; Soltani, M.; and Hegde, C. 2019. One-Shot Neural Architecture Search via Compressive Sensing. *CoRR* abs/1906.02869. URL http://arxiv.org/abs/1906.02869.

Chu, X.; Li, X.; Lu, Y.; Zhang, B.; and Li, J. 2020a. MixPath: A Unified Approach for One-shot Neural Architecture Search. *arXiv preprint arXiv:2001.05887* .

Chu, X.; Zhou, T.; Zhang, B.; and Li, J. 2020b. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *16th Europoean Conference On Computer Vision*. URL https://arxiv.org/abs/1911.12126.pdf.

Gong, X.; Chang, S.; Jiang, Y.; and Wang, Z. 2019. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3224–3234.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861. URL http://arxiv.org/abs/1704.04861.

Huang, G.; Liu, Z.; and Weinberger, K. Q. 2016. Densely Connected Convolutional Networks. *CoRR* abs/1608.06993. URL http://arxiv.org/abs/1608.06993.

Huang, Z.; and Wang, N. 2017. Data-Driven Sparse Structure Selection for Deep Neural Networks. *CoRR* abs/1707.01213. URL http://arxiv.org/abs/1707.01213.

Ida, Y.; Fujiwara, Y.; and Kashima, H. 2019. Fast Sparse Group Lasso. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 1702–1710. Curran Associates, Inc. URL http://papers.nips.cc/paper/8447-fast-sparse-group-lasso.pdf.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* .

Li, G.; Qian, G.; Delgadillo, I. C.; Muller, M.; Thabet, A.; and Ghanem, B. 2020a. Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1620–1630.

Li, Y.; Gu, S.; Mayer, C.; Van Gool, L.; and Timofte, R. 2020b. Group Sparsity: The Hinge Between Filter Pruning and Decomposition for Network Compression. In *Proceedings of the IEEE International Conference on Computer Vision*.

Liang, H.; Zhang, S.; Sun, J.; He, X.; Huang, W.; Zhuang, K.; and Li, Z. 2019. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035* .

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 82–92.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018a. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018b. Hierarchical Representations for Efficient Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, 2736–2744.

Luo, R.; Tian, F.; Qin, T.; Chen, E.; and Liu, T.-Y. 2018. Neural architecture optimization. In *Advances in neural information processing systems*, 7816–7827.

Merity, S.; Keskar, N. S.; and Socher, R. 2018. Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, 4780–4789.

Scardapane, S.; Comminiello, D.; Hussain, A.; and Uncini, A. 2017. Group sparse regularization for deep neural networks. *Neurocomputing* 241: 81–89.

Simon, N.; Friedman, J.; Hastie, T.; and Tibshirani, R. 2013. A Sparse-Group Lasso. *Journal of Computational and Graphical Statistics* 22(2): 231–245. doi:10.1080/10618600.2012.681250. URL https://doi.org/10.1080/10618600.2012.681250.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tian, Y.; Wang, Q.; Huang, Z.; Li, W.; Dai, D.; Yang, M.; Wang, J.; and Fink, O. 2020. Off-policy reinforcement learning for efficient and effective gan architecture search. In *Proceedings of the European Conference on Computer Vision*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58(1): 267–288.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, 2074–2082.

Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; and Keutzer, K. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10734–10742.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations (ICLR)*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Ye, J.; Lu, X.; Lin, Z.; and Wang, J. Z. 2018. Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers. In *International Conference on Learning Representations*.

You, S.; Huang, T.; Yang, M.; Wang, F.; Qian, C.; and Zhang, C. 2020. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1999–2008.

Yuan, M.; and Lin, Y. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68(1): 49–67.

Zhang, X.; Huang, Z.; Wang, N.; XIANG, S.; and Pan, C. 2020. You Only Search Once: Single Shot Neural Architecture Search via Direct Sparse Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1–1.

Zhao, P.; Rocha, G.; Yu, B.; et al. 2009. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics* 37(6A): 3468–3497.

Zheng, X.; Ji, R.; Tang, L.; Zhang, B.; Liu, J.; and Tian, Q. 2019. Multinomial Distribution Learning for Effective Neural Architecture Search. In *Proceedings of the IEEE International Conference on Computer Vision*, 1304–1313.

Zhou, H.; Yang, M.; Wang, J.; and Pan, W. 2019. BayesNAS: A Bayesian Approach for Neural Architecture Search. In *36th International Conference on Machine Learning, ICML 2019*, volume 97, 7603–7613. Proceedings of Machine Learning Research (PMLR).

Zoph, B.; and Le, Q. V. 2018. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.