Revisiting Parameter Sharing for Automatic Neural Channel Number Search

Jiaxing Wang[*†1,3], **Haoli Bai**[*†2], **Jiaxiang Wu**[4], **Xupeng Shi**[5],
Junzhou Huang[4,6], **Irwin King**[2], **Michael Lyu**[2], **Jian Cheng**[1,3]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences
[2] The Chinese University of Hong Kong
[3]School of Artificial Intelligence, University of Chinese Academy of Sciences
[4]Tencent AI Lab [5]Northeastern University [6]University of Texas at Arlington
{hlbai,king,lyu}@cse.cuhk.edu.hk, {jiaxing.wang,jcheng}@nlpr.ia.ac.cn,
jonathanwu@tencent.com, shi.xup@northeastern.edu, jzhuang@uta.edu

Abstract

Recent advances in neural architecture search inspire many channel number search algorithms (CNS) for convolutional neural networks. To improve searching efficiency, parameter sharing is widely applied, which reuses parameters among different channel configurations. Nevertheless, it is unclear how parameter sharing affects the searching process. In this paper, we aim at providing a better understanding and exploitation of parameter sharing for CNS. Specifically, we propose affine parameter sharing (APS) as a general formulation to unify and quantitatively analyze existing channel search algorithms. It is found that with parameter sharing, weight updates of one architecture can simultaneously benefit other candidates. However, it also results in less confidence in choosing good architectures. We thus propose a new strategy of parameter sharing towards a better balance between training efficiency and architecture discrimination. Extensive analysis and experiments demonstrate the superiority of the proposed strategy in channel configuration against many state-of-the-art counterparts on benchmark datasets.

1 Introduction

Convolutional neural networks (CNNs) have achieved great success in various areas, but substantial computational overhead limits their applications on resource-constrained platforms, *e.g.* mobile devices. To design light-weighted CNNs, neural architecture search (NAS) has been broadly adopted for channel number search (CNS) in CNNs [21, 6]. As NAS generally consumes extensive computation resources [39, 12], parameter sharing [25] is widely applied to improve the searching efficiency.

In the context of CNS, parameter sharing refers to reusing convolutional kernels of multiple network architectures. While this is intuitively believed to accelerate network training during searching [37, 6], no analysis is conducted to study its underlying mechanism. Existing parameter sharing methods largely rely on hand-crafted heuristics. For instance, as shown in Figure 1, ordinal selection takes channels ordinally from a shared super kernel to construct different candidates of that convolutional layer [6, 37, 31, 28, 8, 29]. Independent selection, as another common practice, instantiates different candidates of a layer as distinct trainable variables [7, 18, 24]. Though these heuristics are widely applied, it is still not well understood *how parameter sharing benefits the searching process*, and *what the potential drawback is*.

[*]Both authors contribute equally and are listed in the random order.
[†]This work is mainly done during internship at Tencent AI Lab.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

In this paper, we formally investigate these questions. We first establish affine parameter sharing (APS), which unifies previous heuristics as applying different affine transformations on network parameters. The unified formulation facilitates quantitative analysis of the effect of parameter sharing. We thus define a metric to measure how much parameters are shared (*a.k.a. sharing level*), which is based on the cross-covariance matrix between different candidate kernels in APS. It is theoretically found that previous heuristics of ordinal sharing and independent sharing attain the maximum and minimum of the defined metric respectively. We show that a higher level of sharing accelerates the searching process (i.e. faster accuracy rise) by better aligning the gradients of different candidates. However, this also results in coupled optimization among different candidates, making architectures less discriminative. On the contrary, a lower level of sharing can better distinguish architectures but requires more iterations for searching. Therefore it remains a trade-off between the searching efficiency and architecture discrimination. Towards a better balance between the two aspects, we propose a transitionary strategy for APS. Specifically, the sharing level is initialized at maximum such that network parameters can be rapidly optimized in the early stage. Then it is gradually annealed during the searching process such that good architectures can be better distinguished.

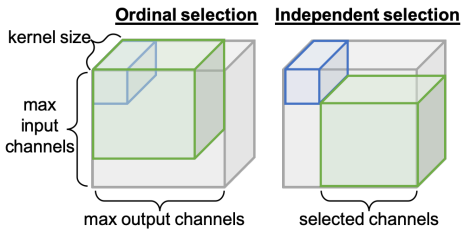


Figure 1: Previous parameter sharing heuristics.

We conduct extensive experiments to study the effects of parameter sharing on channel number search. Besides, the transitionary sharing strategy is shown to achieve a better balance between efficient searching and architecture discrimination. Experimental results on both CIFAR-10 and ImageNet datasets show that our approach outperforms a number of competitive counterparts.

In summary, the contributions of this paper are threefolds:

- We establish affine parameter sharing (APS) as a versatile framework to unify previous hand-crafted parameter sharing heuristics in channel number search (CNS) problems.
- We define a metric to quantitatively measure the level of parameter sharing in the proposed framework, and analyze its effect on CNS algorithms.
- Based on the analysis, we propose transitionary APS, a new parameter sharing strategy that balances searching efficiency and architecture discrimination. Extensive experiments demonstrate the superiority of transitionary APS against previous methods.

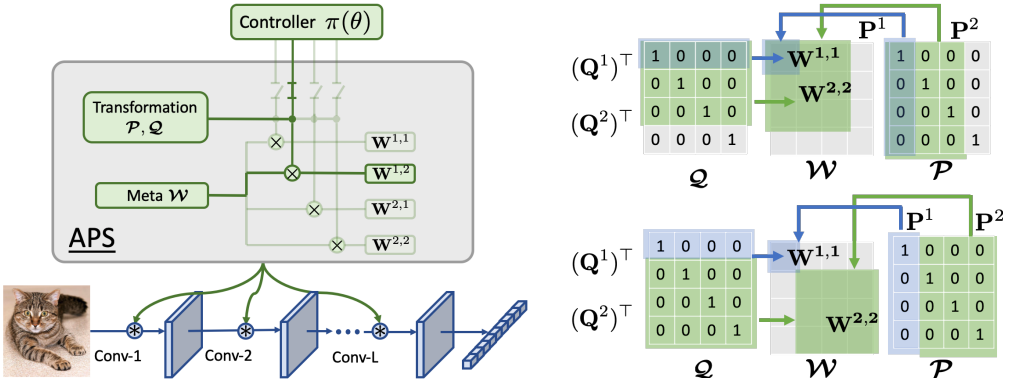
2 Preliminaries

2.1 Problem Setup

Channel number search (CNS) refers to the problem of finding the optimal channel numbers of convolutional neural networks within the computational overhead constraint. Prevalent CNS algorithms adopt a controller $\pi(\theta)$ parameterized by θ for architecture selection, as well as a super-net containing all possible architectures parameterized by w. For a L-layer neural network, layerwise channel number decisions are sampled from $\pi(\theta)$, i.e. $a = [a_1, ..., a_L] \sim \pi(\theta)$, where $a_l \in \mathcal{A} = \{1, 2, ..., A\}$, and $\mathcal{A}$ represents the index set of channel number choices $\mathcal{C} = \{c_1, c_2, ..., c_A\}$. Here we formulate CNS based on reinforcement learning (RL) [25, 30] and take $\pi(\theta)$ as an LSTM controller, while other NAS approaches such as gradient-based formulation [20, 6] can be similarly established. Given channel decision a and its associated parameter $w(a)$, we seek to maximize the expectation of reward function $\mathcal{R}\big(a, w^*(\alpha)\big)$ (e.g. the accuracy on the validation set) as follows:

$$\max_{\theta} \mathbb{E}_{a\sim\pi_\theta} \mathcal{R}\big(a, w^*(a)\big), \;\; \text{s.t. } w^*(a) = \arg\min_{w(a)} \mathcal{L}\big(a, w(a)\big) \;\text{ and }\; \mathcal{B}\big(w(a)\big) \leq B, \tag{1}$$

where $\mathcal{L}\big(a, w(a)\big)$ is the training objective such as cross-entropy, $\mathcal{B}\big(w(a)\big)$ is the network budget function (e.g. FLOPs) and B is the budget constraint. The controller $\pi(\theta)$ is updated with policy gradient [34]. As searching with explicit constraint is infeasible, one can adopt the objective developed in [30] and penalize computation-intensive models softly in the reward function.



(a) The overall framework of APS. (b) Parameter sharing heuristics under APS.

Figure 2: (a) The overall framework of the proposed affine parameter sharing (APS) for CNS. We take $\mathcal{A} = \{1, 2\}$ for illustration; (b) A 2-dimensional illustration of affine parameter sharing with ordinal selection (up) and independent selection (down). The candidate kernel is constructed by transforming the meta-weights into proper shapes with two transformation matrices.

2.2 Parameter Sharing for CNS

Equation 1 forms a typical bi-level optimization problem. To avoid training the associated parameter $w(a)$ to exact convergence before evaluating the architecture, parameter sharing [25] is widely applied in various efficient CNS algorithms. Below we summarize two commonly used sharing heuristics in CNS, which is outlined in Figure 1.

Ordinal Selection [6, 37, 31, 28, 8] maintains a super kernel with a sufficiently large width for each layer. For layer l, parameters of different decisions a_l are obtained by ordinally selecting the top c_{a_l} channels from that kernel. Thus channels with lower indices are multiplexed across different width decisions.

Independent Selection [7, 18, 24] instantiates independent convolutional kernels for each candidate $a_l \in \mathcal{A}$ in layer l. It is assumed that different channel configurations should be treated individually. Channels of different candidates are non-multiplexed in independent selection scheme.

3 Methodology

To investigate the role of parameter sharing for CNS, we first establish affine parameter sharing (APS), a general framework that unifies previous heuristics. Within APS, we quantitatively evaluate the level of parameter sharing, and demonstrate how it affects the searching dynamics. Based on our findings, we propose transitionary APS, a new strategy that dynamically adjusts the trade-off between efficient training and architecture discrimination. In the following discussion, we follow the standard notations: For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{a}_x \in \mathbb{R}^m$ is the x-th column; and $a_{x,y}$ is the (x, y)-th element of $\mathbf{A}$. $\|\cdot\|_F$ refers to the Frobenius norm, and $\|\cdot\|_2$ is the l_2-norm. We denote the range of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ as $\mathcal{R}(\mathbf{A}) = \{\mathbf{Ax} \mid \mathbf{x} \in \mathbb{R}^n\}$.

3.1 Affine Parameter Sharing

To facilitate the analysis of parameter sharing, a first step is to unify previous parameter sharing heuristics. Towards that end, we propose affine parameter sharing (APS), a general framework that allows flexible parameter sharing. Specifically, for each convolutional layer we maintain a meta-weight $\mathcal{W} \in \mathbb{R}^{c\times c\times k\times k}$ as the shared parameter pool for all candidates, where c, k are the number of filters and kernel size respectively. To transform $\mathcal{W}$ to different sizes, we keep two sets of transformation matrices $\mathcal{P} = \{\mathbf{P}^1, ..., \mathbf{P}^A\}$ and $\mathcal{Q} = \{\mathbf{Q}^1, ..., \mathbf{Q}^A\}$, where $\mathbf{P}^a, \mathbf{Q}^a \in \mathbb{R}^{c\times c_a}$ $(c \geq c_a)$ are designed to be semi-orthogonal [3] such that the distinctiveness in c_a-dimensional space is

[3] $\mathbf{A} \in \mathbb{R}^{m\times n}$ is semi-orthogonal if $\mathbf{A}^\top\mathbf{A} = \mathbf{I}_n$ for $m > n$.

maximally preserved. Given decisions on input and output width $i, o \in \mathcal{A}$, the candidate parameter $\mathbf{W}^{i,o} \in \mathbb{R}^{c_i \times c_o \times k \times k}$ can be obtained by affine transformation as:

$$\mathbf{W}^{i,o} = (\mathbf{Q}^i)^\top \times_2 \boldsymbol{\mathcal{W}} \times_1 \mathbf{P}^o, \tag{2}$$

where $\times_d$ denotes mode d multiplication [16], i.e, the matrix multiplication along the d-th dimension. The scheme of affine parameter sharing is visualized in Figure 2(a).

Remark APS can be easily reduced to previous parameter sharing heuristics with different $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{Q}}$. Suppose $\{\mathbf{e}_j\}_{j=1}^c$ are standard basis in $\mathbb{R}^c$. For $\forall o, i \in \mathcal{A}$, APS is reduced to **ordinal selection** [6, 21, 37, 31, 28, 8] by choosing $\mathbf{P}^o = [\mathbf{e}_1, ..., \mathbf{e}_{c_o}]$ and $\mathbf{Q}^i = [\mathbf{e}_1, ..., \mathbf{e}_{c_i}]$. On the other hand, by taking disjoint sets of $\{\mathbf{e}_j\}_{j=1}^c$ in $\mathbf{P}^o$ or $\mathbf{Q}^i$, APS is equivalent to **independent selection** [7, 18, 24]. A 2-dimensional illustration is presented in Figure 2(b).

3.2 Quantitative Measurement

Given the formulation of APS, we are able to quantitatively measure the level of parameter sharing. For notation simplicity, we treat meta weight $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{C \times C}$ as 2-D matrix and perform matrix multiplication.

Definition 3.1. *Assuming each element of meta weight $\boldsymbol{\mathcal{W}}$ follows the standard normal distribution, the **level of affine parameter sharing** of two candidate decisions (i, o) and $(\tilde{i}, \tilde{o})$ is defined as the Frobenius norm of cross-covariance matrix*[4] *between candidate parameters $\mathbf{W}^{i,o}$ and $\mathbf{W}^{\tilde{i},\tilde{o}}$, i.e. $\phi(i, o; \tilde{i}, \tilde{o}) = \left\| \operatorname{Cov}\left(\mathbf{W}^{i,o}, \mathbf{W}^{\tilde{i},\tilde{o}}\right) \right\|_F^2$.*

In other words, the sharing level can be quantitatively reflected by the squared sum of pairwise correlations of two candidate parameters. A large sharing level indicates high coupling between two candidates and vice versa. Taking the entire search space $\mathcal{C}$ into consideration, we are interested in the overall level of parameter sharing $\Phi = \sum_{i \le \tilde{i}} \sum_{o \le \tilde{o}} \phi(i, o; \tilde{i}, \tilde{o})$, as well as its maximal and minimal conditions. Without loss of generality, assuming $c_i \le c_j$ for $i < j$, we have the following theorem:

Theorem 3.1. *For $\forall i \le \tilde{i}$ and $\forall o \le \tilde{o}$, the overall level Φ of APS is maximized if $\mathcal{R}\left(\mathbf{Q}^i\right) \subseteq \mathcal{R}\left(\mathbf{Q}^{\tilde{i}}\right)$ and $\mathcal{R}\left(\mathbf{P}^o\right) \subseteq \mathcal{R}\left(\mathbf{P}^{\tilde{o}}\right)$. Φ is minimized if $\mathcal{R}\left(\mathbf{Q}^i\right) \subseteq \mathcal{R}^\perp\left(\mathbf{Q}^{\tilde{i}}\right)$ and $\mathcal{R}\left(\mathbf{P}^o\right) \subseteq \mathcal{R}^\perp\left(\mathbf{P}^{\tilde{o}}\right)$.*

The proof is left in Appendix B. The theorem connects the level of parameter sharing with the range of transformation matrices $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{Q}}$. Notably, previous heuristics of **ordinal selection** and **independent selection** attain maximum and minimum Φ respectively. With various designs of $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{Q}}$, APS allows more flexible patterns of parameter sharing.

3.3 Parameter Sharing and the Searching Dynamics

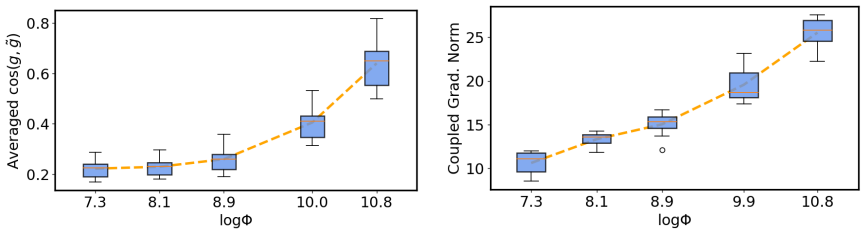
The effects of parameter sharing can be reflected by the impact of different parameter sharing level Φ in the searching process. Specifically, we have the following observations:

Parameter Sharing Benefits Efficient Searching We first investigate the impact of sharing level Φ on searching efficiency. Given two candidates (i, o) and $(\tilde{i}, \tilde{o})$, we check the relationship between Φ and their gradients alignment on meta-weights $\boldsymbol{\mathcal{W}}$, which is computed by the cosine similarity:

$$\cos(\mathbf{g}, \tilde{\mathbf{g}}) = \frac{\mathbf{g}^\top \tilde{\mathbf{g}}}{\|\mathbf{g}\|_2 \cdot \|\tilde{\mathbf{g}}\|_2}, \quad \text{where } \mathbf{g} = \nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}\left(\mathbf{W}^{i,o}\right), \ \tilde{\mathbf{g}} = \nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}\left(\mathbf{W}^{\tilde{i},\tilde{o}}\right). \tag{3}$$

A positive cosine value indicates that $\nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}\left(\mathbf{W}^{i,o}\right)$ stands in the same side with $\nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}\left(\mathbf{W}^{\tilde{i},\tilde{o}}\right)$, thus the gradient update on candidate (i, o) is also a descent direction for the other configuration $(\tilde{i}, \tilde{o})$. We plot the sharing level Φ against the averaged cosine similarity on a 20-layer residual network, shown in Figure 3(a), It can be observed that a larger Φ gives more alignment of gradients. In other words, each gradient update simultaneously benefits multiple architectures with parameter sharing, and thus accelerates the searching process in the sense of accuracy raise.

[4] The cross-covariance matrix between $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{\tilde{m} \times \tilde{n}}$ is defined as $\operatorname{Cov}(\mathbf{X}, \mathbf{Y}) = \mathbb{E}[(\mathbf{X} - \mathbb{E}(\mathbf{X})) \otimes (\mathbf{Y} - \mathbb{E}(\mathbf{Y}))^\top] \in \mathbb{R}^{m \times n \times \tilde{m} \times \tilde{n}}$, where $\otimes$ is the Kronecker product.



(a) Φ against gradient cosines similarity. (b) Φ against norm of coupled gradients.

Figure 3: The variation of Φ against the cosine similarity (left) and the norm of coupled gradients (right). We take the training of a 20-layer residual network for demonstration.

Parameter Sharing Couples Architecture Optimization As a side effect of efficient searching, sharing parameters inevitably couples the update of multiple architectures. We temporarily clean the notation by abbreviating the forwarding kernel $\mathbf{W}^t = \mathbf{W}^{i_t,o_t}$, transformation matrices $\mathbf{P}^t = \mathbf{P}^{o_t}$ and $\mathbf{Q}^t = \mathbf{Q}^{i_t}$ at time step t. Given decisions (i_t, o_t), the forwarding kernel can be updated as $\mathbf{W}^t = (\mathbf{Q}^t)^\top \mathcal{W}^{t-1}\mathbf{P}^t - \eta(\mathbf{Q}^t)^\top \mathbf{Q}^{t-1}\big(\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}^{t-1})\big)(\mathbf{P}^{t-1})^\top \mathbf{P}^t$. For $i_t = i_{t-1}$ and $o_t = o_{t-1}$, the semi-orthogonality constraints on $\mathbf{P}_t$ and $\mathbf{Q}_t$ reduce the update to the gradient descent on the same architecture. However, when $i_t \neq i_{t-1}$ or $o_t \neq o_{t-1}$, the update from other candidates interfere the current candidate. By expanding all historical updates and re-arranging them properly, we have:

$$\mathbf{W}^t = (\mathbf{Q}^t)^\top \mathcal{W}^0 \mathbf{P}^t - \eta \underbrace{\sum_{\substack{i_{\tilde{t}}=i_t \\ o_{\tilde{t}=o_t}}} (\mathbf{Q}^t)^\top \mathbf{Q}^{\tilde{t}}\big(\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}^{\tilde{t}})\big)(\mathbf{P}^{\tilde{t}})^\top \mathbf{P}^t}_{\text{Normal updates on the current candidate}} - \eta \underbrace{\sum_{\substack{i_{\tilde{t}}\neq i_t, \text{or} \\ o_{\tilde{t}}\neq o_t}} (\mathbf{Q}^t)^\top \mathbf{Q}^{\tilde{t}}\big(\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}^{\tilde{t}})\big)(\mathbf{P}^{\tilde{t}})^\top \mathbf{P}^t}_{\text{Coupled updates from other candidates}}.$$

In Figure 3(b), we compare the sharing level Φ against the Frobenius norm of coupled gradient (i.e, the third term) on the same 20-layer residual network. A larger Φ gives rise to more coupling among candidates, which could make the controller less discriminative to distinguish different architectures.

3.4 Transitionary Strategy

With the above analysis, we see that a higher level of sharing accelerates the searching process but couples the optimization of different candidates, making them less discriminative to the controller. It is thus critical to balance these two aspects. Towards that end, we propose a transitionary strategy for APS. We initialize $\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}$ with ordinal selection, where Φ attains its maximum and gradually anneal it. A large Φ in early stages quickly warms up the network, while the annealed Φ later on decouples the optimization and thus gives higher confidence to good architectures. The transition can be realized by minimizing the APS sharing level Φ with regard to $\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}$ as follows:

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{Q}}} \ \Phi \triangleq \sum_{i\leq\tilde{i}}\sum_{o\leq\tilde{o}} \Big\| \operatorname{Cov}\Big(\mathbf{W}^{i,o}, \mathbf{W}^{\tilde{i},\tilde{o}}\Big) \Big\|_F^2 = \sum_{i\leq\tilde{i}}\sum_{o\leq\tilde{o}} \|{\mathbf{Q}^i}^\top \mathbf{Q}^{\tilde{i}}\|_F^2 \cdot \|{\mathbf{P}^o}^\top \mathbf{P}^{\tilde{o}}\|_F^2, \\ &\text{s.t.} \ \big\|\mathbf{p}_x^o\big\|_2^2 = 1, \ \ \big\|\mathbf{q}_y^i\big\|_2^2 = 1, \ \text{ for } x \in \{1,...,c_o\}, y \in \{1,...,c_i\} \text{ and } i, o \in \mathcal{A} \end{aligned} \tag{4}$$

where the unit length constraints prevent the trivial zero solution. Note that the original semi-orthogonality constraints on $\mathbf{P}^o, \mathbf{Q}^i$ lead to a Stiefel manifold optimization [33] problem, which is computationally expensive. Instead, Equation 4 provides an feasible reformulation to the problem. For more details on both the derivation and efficient implementation of Equation 4, please refer to Appendix C. We apply projected gradient descent to update $\mathbf{P}^o$ and $\mathbf{Q}^i$ as:

$$\mathbf{p}_x^o \leftarrow \Pi_{\mathcal{U}}(\mathbf{p}_x^o - \tau\nabla_{\mathbf{p}_x^o}\Phi), \ \ \mathbf{q}_y^i \leftarrow \Pi_{\mathcal{U}}(\mathbf{q}_y^i - \tau\nabla_{\mathbf{q}_y^i}\Phi), \quad \text{where } \mathcal{U} = \{\mathbf{u} \in \mathbb{R}^C | \ \|\mathbf{u}\|_2 = 1\}. \tag{5}$$

The learning rate τ controls the transition rate of Φ. In summary, $\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}$ control the transition of parameter sharing, and $\boldsymbol{\mathcal{W}}$ is updated for the task-specific loss, both of which are optimized in a decoupled way. An overall workflow is shown in Algorithm 1 of Appendix A.

4 Related Work

Channel configuration has long been a research focus in academia to design light-weighted and efficient deep learning models. Our work follows the recent trend in neural architecture search (NAS) and adopts a new kind of parameter sharing in the proposed algorithm. Thus we summarize the related literature from these two strands of research in the following.

NAS-based Approaches The widely developed area of neural architecture search [39, 25, 3, 32, 36, 14] has inspired a number of automatic algorithms for channel configuration [12, 21, 6, 37]. Different from earlier works that determines the pruning strategy in a hand-crafted way [13, 5, 10, 11, 1], NAS-inspired approaches typically search for the optimal configuration and saves much manual labor. Conventional NAS paradigm involves joint optimization of a controller (that outputs the strategy) and a supernet (that is updated for the task specific loss). Under this paradigm, common searching approaches include differentiable search [6, 22], evolutionary algorithms [21, 19] and reinforcement learning [12, 35]. Our work follows this paradigm and adopts reinforcement learning as a searching engine due to its superiority in run-time and memory efficiency. One-shot NAS is the other paradigm that updates the supernet and search the strategy in two disjoint stages [2, 8, 37]. Note that our work can be readily extended to the one-shot paradigm by first updating the meta parameters and then training the RL controller separately. Besides, we highlight that our analysis of parameter sharing is fundamental and agnostic to the paradigm of NAS.

Parameter Sharing NAS-based methods typically rely on parameter sharing [25] techniques to enable efficient searching. In the context of automatic channel configuration, existing approaches can be mainly categorized into two heuristic schemes: ordinal selection [6, 37, 29, 31, 28, 8] and independent selection [7, 18, 24], both of which adopt linear projection as discussed in previous sections. To better interprete the role of parameter sharing, there are some preliminary discussions [2, 38, 26, 4, 23] but grounded in general NAS. In this paper, we target an automatic channel number search and seek for a better understanding of parameter sharing specifically. We establish a unified view and formally investigate the effects of different parameter sharing schemes. Our analysis leads to a transition-based algorithm, which enjoys both efficient and discriminative searching.

Finally, we remark that parameter sharing can also be realized by using a meta-network to generate weights of different candidate models [21]. However, the meta-network could be memory consuming as its output dimension of fully connected layers matches the maximum number of model parameters. Moreover, while a meta-network allows more flexibility in parameter sharing, it is less interpretable and controllable comparing to our discussion.

5 Experiments

In this section, we first demonstrate the effect of parameter sharing and the advantage of the proposed transitionary strategy for CNS in Section 5.2. Then we compare to state-of-the-art algorithms in Section 5.3. Code is available at `https://github.com/haolibai/APS-channel-search`.

5.1 Experimental Setup

We conduct experiments on CIFAR-10 [17] and ImageNet 2012 [15], following standard data pre-processing techniques in [9, 27]. A brief summarization of experimental setup is introduced below, while complete hyper-parameter settings and implementation details can be found in Appendix C.

CIFAR-10 Experiments For CIFAR-10, we take ResNet [9] as base models similar to [6, 12]. To be consistent with [6], the total searching epoch is set to 600, which can be finished within 6.9 hours for ResNet-20 and 8.6 hours for ResNet-56 on a single NVIDIA Tesla-P40. The first 200 epochs are used for warm-up training with fixed $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{Q}}$, and candidate architectures are uniformly sampled from $\mathcal{C}$. The rest 400 epochs are left for transition and training of the RL controller. We set $\mathcal{C} = \{16, 32, 64, 96\}$ for the analysis of parameter sharing in Section 5.2 and 100% FLOPs search, and $\mathcal{C} = \{4, 8, 16, 32, 64\}$ when searching for more compact model to compare to other baselines in Section 5.3. Note that in CIFAR-10 experiments all convolutional layers share the same search space, which is free from domain expertise on the search space design.

ImageNet Experiments For ImageNet experiments, we choose ResNet-18 and MobileNet-v2 as base models. For memory efficiency, we increase candidate channels after each down-sampling layer according to default expansion rates of base models. The initial candidates $\mathcal{C}$ are set to $\{32, 48, 64, 80\}$

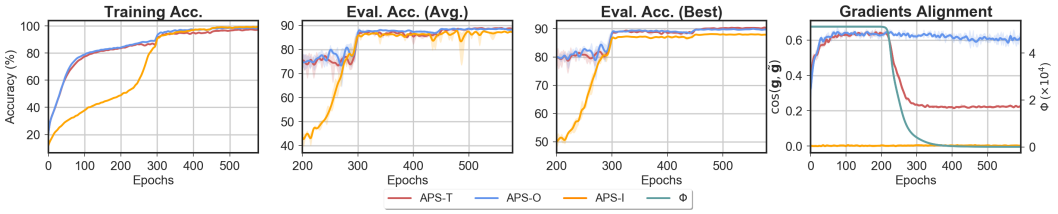


Figure 4: The left three figures show the Accuracy curvature with APS-O, APS-I and APS-T. For evaluation, we sample 20 architectures and report average and maximal accuracy. The rightmost figure shows the averaged alignment of gradients $(\cos(\mathbf{g}, \tilde{\mathbf{g}}))$ of APS-O, APS-I and APS-T, as well as the change of the corresponding parameter sharing level Φ for APS-T along the searching trajectory.

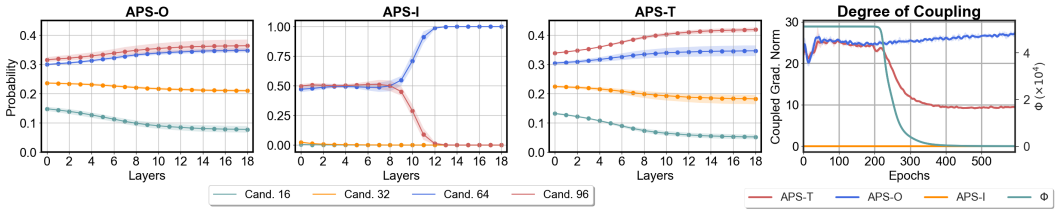


Figure 5: The left three figures show the layer-wise decision probabilities of controller π on ResNet-20 over the last 100 training epochs. The solid line denotes the expected logit values, and shadowed areas are 95% confidence intervals. The rightmost figure shows the averaged norm of coupled gradients of APS-O, APS-I and APS-T respectively, as well as the change of the corresponding parameter sharing level Φ for APS-T along the searching trajectory.

for ResNet18 and $\{8, 12, 16, 20\}$ for MobileNet-v2 respectively. We search for 160 epochs where the first 80 epochs are for warm-up training. The whole searching process can be finished within 24 hours for ResNet-18 and 48 hours for MobileNet-v2 on four NVIDIA Tesla-P40s.

5.2 The Effect of Parameter Sharing

We use ResNet-20 on CIFAR-10 for illustration. We denote the transitionary strategy as APS-T, and compare it against APS-O (ordinal selection) and APS-I (independent selection), which attain maximum and minimum of Φ respectively. Given no FLOPs constraint, the oracle architecture is supposed to attain the maximum channel capacity. Therefore, we perform the search without FLOPs constraint to compare how close the searched architecture is to this optimal oracle solution.

Network Optimization To inspect the optimization dynamics, we plot the accuracy curvature of training, evaluation, the variation of averaged alignment of gradients $(\cos(\mathbf{g}, \tilde{\mathbf{g}}))$ as well as the parameter sharing level Φ in Figure 4. It can be found that the accuracy of both ASP-O and ASP-T raises much faster than ASP-I. This indicates that the maximized parameter sharing can indeed accelerate the optimization especially during the warm-up period (first 200 epochs). From the rightmost figure, the alignment of gradients decreases with the annealed sharing level Φ when APS-T is applied. Nevertheless, it does not affect the accuracy much in late stages when APS-O, APS-I, and APS-T are mostly overlapped.

Architecture discrimination The architecture discrimination can be reflected by the probabilities of layer-wise channel decisions from the controller after searching. From Figure 5, APS-O cannot effectively distinguish the best choice (96) and the second best choice (64) throughout all layers. APS-I separates different candidates by a large margin especially in deep layers but is sometimes stuck in incorrect local optima[5]. This is possibly due to the incorrect reward from the insufficiently trained meta weights to the controller, such that the controller cannot explore better solution space. Finally, our APS-T confidently separates each candidate, and the optimal solution of maximum channel capacity can be stably attained across different runs. We also plot the variation of the averaged norm of coupled gradients and parameter sharing level Φ in the rightmost of Figure 5. The transitionary strategy successfully decreases the norm of coupled gradients that facilitates better architecture discrimination.

[5]Due to limited space we only present one group of results, and more results are shown in Appendix E.

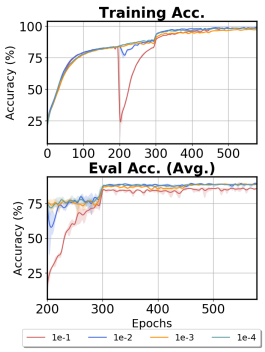


Figure 6: Accuracies with different learning rates of $\mathcal{P}$, $\mathcal{Q}$.

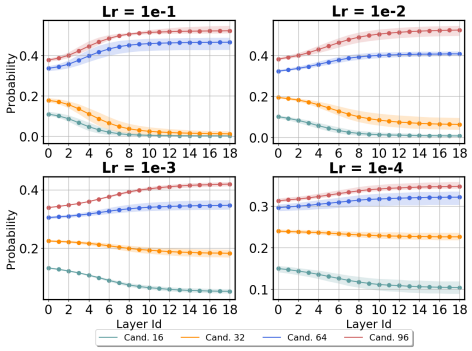


Figure 7: Probabilities of layer-wise channel decisions with different learning rates of $\mathcal{P}$, $\mathcal{Q}$.

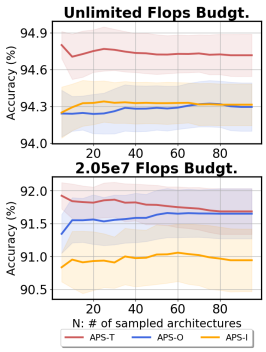


Figure 8: Accuracies of the top-N likely sampled models.

Transition Rate We study the learning rate τ of $\mathcal{P}$, $\mathcal{Q}$ that controls the speed of the transition. We enumerate over $\tau \in \{$1e-1, 1e-2, 1e-3, 1e-4$\}$, and the converged values of Φ are: 2.78e4, 2.20e3, 2.70e2, 5.05e4 respectively. We first plot accuracy curvatures in Figure 6. It can be found that with a large τ (e.g. 1e-1) makes the transition non-smooth, which leads to a sudden drop of accuracy. Then we plot the probabilities of layerwise channel decisions under different learning rates in Figure 7. We see that either large or small τ cannot separate different architectures with insufficiently optimized Φ, while only a proper learning rate (e.g. 1e-3) produces more discriminative candidates with lower Φ.

Sampled Architectures For practical deployment, it is necessary to check the overall qualities of sampled architectures. We generate top-N architectures via beam search according to the sequence likelihood by the controller, and train them from scratch to obtain the final performance. We conduct both unlimited FLOPs budget search and 2.05e7 FLOPs constraint ($\sim$50% of the original ResNet-20) search, and plot the accumulated accuracies of top-N likely models in Figure 8. Both mean and standard deviation are reported. It can be found that given no FLOPs constraint, APS-T clearly outperforms APS-O and APS-I, as the top-N models of APS-T distribute around the optimal solution. Under FLOPs constraint, APS-T still outperforms the other two. The improvement decreases as N increases, indicating that with APS-T, better architectures can be picked with higher probabilities.

5.3 Comparisons with state-of-the-arts

Finally, we compare APS-T against many state-of-the-art methods ranging from hand-crafted (HC) channel pruning [13, 5, 10], to automatic (Auto) channel search [6, 21, 37]. We perform the search under different FLOPs constraints, and then apply beam search to generate top-N architectures according to the sequence likelihood from the RL controller. We choose the architecture closest to the target FLOPs from the top-N candidates, and train it from scratch to obtain the final performance. The results on CIFAR-10 and ImageNet are shown in Table 1 and Table 2 respectively.

From Table 1, APS-T discovers architectures with better or comparable performance under various FLOPs constraints. Note that this is achieved with all layers sharing the same set of candidate widths, without domain expertise on the design of search space. Moreover, searching with 100% FLOPs of ResNet-20 and ResNet-56 increase 0.36% and 0.26% accuracy respectively comparing to base models. Finally, APS-T also outperforms APS-I and APS-O by a clear margin, demonstrating the effectiveness of the proposed transitionary method.

For ResNet-18 on ImageNet, APS-T consistently outperforms baselines under similar computational FLOPs. For instance, it achieves 69.34% top-1 accuracy given 41.8% reduction of FLOPs, surpassing the most competitive TAS by 0.19%. For MobileNet-v2, it achieves more than 0.7% gain of accuracy with only 3% more FLOPs comparing to MetaPrune, and higher accuracy to MetaPrune and AutoSlim without knowledge distillation under 314M FLOPs. Meanwhile, APS-T surpasses APS-I and APS-O under similar FLOPs constraints on both models. We also visualize the channel configuration of both ResNet-18 and MobileNet-v2 in Appendix E.

To further demonstrate APS-T, we draw the accuracy curvature of ResNet-20 and ResNet-18 against different FLOPs constraints in Figure 9. It can be found that our APS-T mostly outperforms TAS as well as uniform scaling strategy on both base networks.

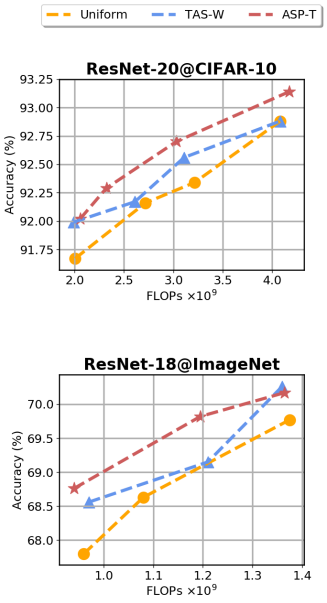
Table 1: Comparison of different algorithms for ResNet-20 and ResNet-56 on CIFAR10. Drops↓ denotes the decrease of accuracy comparing to base models, and Ratio↓ is the reduction of FLOPs. - stands for unavailable records. * denotes the results reported with knowledge distillation/depth search and † indicates results reported with pre-trained model, both of which are absent in our model.

Methods	Types	ResNet-20				ResNet-56			
		Accuracy	Drop↓	FLOPs	Ratio↓	Accuracy	Drop↓	FLOPs	Ratio↓
Original	-	92.78%	-	40.8M	0.0%	94.28%	-	126.0M	0.0%
CP [13]	*HC*	-	-	-	-	91.80%	1.00%	62.9M	50.0%
LCCL [5]	*HC*	91.68%	1.06%	26.1M	36.0%	92.81%	1.54%	78.1M	37.9%
SFP [10]	*HC*	90.83%	1.37%	24.3M	42.2%	93.35%	0.24%	59.4M	52.6%
FPGM [11]	*HC*	91.09%	1.11%	24.3M	42.2%	92.93%	0.66%	59.4M	52.6%
FPGM† [11]	*HC*	-	-	-	-	93.49%	0.10%	59.4M	52.6%
AMC [12]	*Auto*	-	-	-	-	91.90%	0.90%	62.9M	50.0%
TAS-W [6]	*Auto*	91.99%	0.89%	19.9M	51.3%	92.87%	1.59%	63.1M	49.9%
TAS-W* [6]	*Auto*	92.31%	0.57%	19.9M	51.3%	93.69%	0.77%	59.5M	52.7%
APS-O	*Auto*	91.61%	1.17%	19.1M	53.4%	92.93%	1.35%	57.7M	53.9%
APS-I	*Auto*	91.24%	1.54%	21.5M	47.5%	92.85%	1.43%	57.8M	53.8%
APS-T	*Auto*	92.02%	0.76%	20.6M	49.6%	93.42%	0.86%	60.3M	51.8%
APS-T	*Auto*	93.14%	-0.36%	41.7M	-2.3%	94.54%	-0.26%	122.5M	2.7%

Table 2: Comparison for ResNet-18 and MobileNet-V2 on ImageNet. * denotes original results with knowledge distillation.

Methods	Types	Top-1 Acc	Top-5 Acc	FLOPs	Ratio↓
Resnet-18 [9]	-	69.76%	89.08%	1.82G	0.0%
LCCL [5]	*HC*	66.33%	86.94%	1.19G	34.6%
SFP [10]	*HC*	67.10%	87.78%	1.06G	41.8%
FPGM [11]	*HC*	68.41%	88.48%	1.06G	41.8%
TAS [6]	*Auto*	69.15%	89.19%	1.21G	33.3%
APS-O	*Auto*	68.60%	88.44%	1.04G	42.9%
APS-I	*Auto*	68.32%	88.21%	1.05G	41.8%
APS-T	*Auto*	69.34%	88.89%	1.05G	41.8%
APS-T	*Auto*	70.17%	89.59%	1.36G	24.9%
APS-T	*Auto*	71.67%	90.36%	1.83G	-0.9%
MobileNet-V2 [27]	-	71.80%	91.00%	314M	0.0%
×0.65 scaling	*HC*	67.20%	-	140M	55.4%
MetaPrune [21]	*Auto*	68.20%	-	140M	53.3%
MetaPrune [21]	*Auto*	72.70%	-	300M	4.4%
AutoSlim [37]	*Auto*	72.49%	90.50%	305M	2.9%
AutoSlim* [37]	*Auto*	74.20%	-	305M	2.9%
APS-O	*Auto*	72.58%	90.76%	316M	-0.6%
APS-I	*Auto*	72.38%	90.54%	311M	1.0%
APS-T	*Auto*	68.96%	88.48%	156M	50.3%
APS-T	*Auto*	72.83%	90.75%	314 M	0.0%

Figure 9: Comparison under different FLOPs constraint.



6 Conclusion

In this paper, we pioneer to provide analysis of the effect of parameter sharing in automatic channel number search, and conduct a preliminary research on exploiting the strength of parameter sharing and avoiding its weakness. We first propose a general framework named affine parameter sharing (APS) to unify previous parameter sharing heuristics. Then we quantitatively measure APS and demonstrate how it affect the searching process. Empirical results show that parameter sharing brings efficient searching of network parameters but also results in less discrimination of architectures. Thus we are motivated to propose the transitionary APS strategy, such that a proper balance between searching efficiency and architecture discrimination can be achieved. Extensive experiments and analysis are conducted to demonstrate the effectiveness of our strategy.

Broader Impact

The goal of this paper is to build better understanding of parameter sharing in neural architecture search. Based on the observation and analysis, we propose a new parameter sharing scheme that enjoys both efficient searching and better architecture discrimination. Although the discussion is restricted to the channel number search for now, we hope it can shed more light in the general setting, and thereon inspire more general NAS algorithms that are both efficient and accurate. This research may benefit the recently popular area of automatic machine learning (AutoML). In AutoML, it saves much manual costs and eliminates repetitive labor to automatically design compact and high-performance models for various resource constrained platforms. These AI services can be readily applied in various applications and deployed on edge devices like smart phones. Abuse of these AI services, however, may bring excessive collection of personal data and increase the risk of malicious capturing private information. Besides, too little human intervention may leads to excessive trust in the model. The problem can be serious in cases such as automobile navigation. We believe that it is necessary for the community to conduct research to mitigate potential risks in AutoML.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14210717 of the General Research Fund, No. CUHK 2300174 and No. C5026-18GF of the Collaborative Research Fund), the National Key Research and Development Program of China (No. 2018AAA0100204, No. 2020AAA0103402), National Natural Science Foundation of China (No. 61972396, No. 61876182, No. 61906193), and the Strategic Priority Research Program of Chinese Academy of Science (No. XDB32050200). We sincerely thank Kuo Zhong and Jiajin Li for helpful discussions, as well as the anonymous reviewers for insightful suggestions.

References

[1] Haoli Bai, Jiaxiang Wu, Irwin King, and Michael R. Lyu. Few shot network compression via cross distillation. In *AAAI Conference on Artificial Intelligence*, pages 3203–3210, 2020.

[2] Gabriel Bender. Understanding and simplifying one-shot architecture search. In *Proceedings of the International Conference on Machine Learning*, pages 549–558, 2019.

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *Proceedings of the International Conference of Representation Learning*, 2019.

[4] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. Preprint arXiv:1907.01845, 2019.

[5] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2017.

[6] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*, pages 760–771, 2019.

[7] 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 IEEE Conference on Computer Vision and Pattern Recognition*, pages 10625–10634, 2020.

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

[9] 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.

[10] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2234–2240, 2018.

[11] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[12] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision*, pages 815–832, 2018.

[13] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1398–1406, 2017.

[14] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12084–12092, 2020.

[15] Deng Jia, Dong Wei, Socher Richard, Li Li-Jia, Li Kai, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248—-255, 2009.

[16] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[17] Alex Krizhevsky and Geffery Hinton. Learning multiple layers of features from tiny images. In *Technical report*, 2009.

[18] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation. Preprint arXiv:1911.13053, 2019.

[19] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 673–679, 2020.

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proceedings of the International Conference of Representation Learning*, 2019.

[21] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3296–3305, 2019.

[22] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. Preprint arXiv:2004.02164, 2020.

[23] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yong Guo, Peilin Zhao, Junzhou Huang, and Mingkui Tan. Disturbance-immune weight sharing for neural architecture search. Preprint arXiv:2003.13089, 2020.

[24] Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation search in channel-level for object detection. In *Advances in Neural Information Processing Systems*, pages 14290–14299, 2019.

[25] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *Proceedings of the International Conference on Machine Learning*, pages 4092–4101, 2018.

[26] Aloïs Pourchot, Alexis Ducarouge, and Olivier Sigaud. To share or not to share: A comprehensive appraisal of weight-sharing. Preprint arXiv:2002.04289, 2020.

[27] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. Preprint, 2018.

[28] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. Preprint arXiv:1904.02877, 2019.

[29] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path NAS: designing hardware-efficient convnets in less than 4 hours. In *Machine Learning and Knowledge Discovery in Databases - European Conference*, pages 481–497, 2019.

[30] 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.

[31] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12962–12971, 2020.

[32] Jiaxing Wang, Jiaxiang Wu, Haoli Bai, and Jian Cheng. M-NAS: meta neural architecture search. In *AAAI Conference on Artificial Intelligence*, pages 6186–6193, 2020.

[33] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[34] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[35] Jiaxiang Wu, Yao Zhang, Haoli Bai, Huasong Zhong, Jinlong Hou, Wei Liu, and Junzhou Huang. Pocketflow: An automated framework for compressing and accelerating deep neural networks. In *Advances in Neural Information Processing Systems (NIPS), Workshop on Compact Deep Neural Networks with Industrial Applications*, 2018.

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

[37] Jiahui Yu and Thomas S. Huang. Autoslim: Towards one-shot architecture search for channel numbers. Preprint arXiv:1903.11728, 2019.

[38] Kaicheng Yu, René Ranftl, and Mathieu Salzmann. How to train your super-net: An analysis of training heuristics in weight-sharing NAS. Preprint arXiv:2003.04276, 2020.

[39] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2017.