Pattern Recognition 136 (2023) 109215

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Neural operator search

Wei Li [a], Shaogang Gong [a], Xiatian Zhu [b,*]

[a] School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK
[b] Surrey Institute for People-Centred Artificial Intelligence, and Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, Surrey, GU2 7XH, UK

ARTICLE INFO

Article history:
Received 18 February 2021
Revised 21 October 2022
Accepted 24 November 2022
Available online 28 November 2022

Keywords:
Neural architecture search
Search space
Self-calibration operations
Dynamic convolution
Attention learning
Block design
Neural operation
Knowledge distillation

ABSTRACT

Existing neural architecture search (NAS) methods usually explore a limited *feature-transformation-only* search space, ignoring other advanced feature operations such as feature self-calibration by attention and dynamic convolutions. This disables the NAS algorithms to discover more advanced network architectures. We address this limitation by additionally exploiting feature self-calibration operations, resulting in a heterogeneous search space. To solve the challenges of operation heterogeneity and significantly larger search space, we formulate a *neural operator search* (NOS) method. NOS presents a novel heterogeneous residual block for integrating the heterogeneous operations in a unified structure, and an attention guided search strategy for facilitating the search process over a vast space. Extensive experiments show that NOS can search novel cell architectures with highly competitive performance on the CIFAR and ImageNet benchmarks.

1. Introduction

Recent advances of Neural Architecture Search (NAS) are remarkable in challenging tasks, e.g. image classification [1,2], object detection [3], and semantic segmentation [4,5], greatly alleviating the demands for human knowledge and interventions [6,7] by automating the laborious process of designing neural network architectures. One common scheme for the standard proxy-based neural architecture search methods [8–10] is to factorise the search space via repeatedly stacking the same cell structure, within which a computing block generates an output tensor $\boldsymbol{F}_k$ by combining the transformations of two input feature tensors $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ as follows:

$$\boldsymbol{F}_k = o^{i\to k}(\boldsymbol{F}_i) \oplus o^{j\to k}(\boldsymbol{F}_j) \quad \text{s.t.} \quad i < k \,\&\, j < k, \tag{1}$$

where $o^{i\to k}$ and $o^{j\to k}$ are the ith and jth primitive operations for feature transformation, selected from a candidate operation set $\mathcal{O}$, and $\oplus$ is the element-wise addition. Existing NAS methods use only the standard *feature learning/transformation* operations (convolution, pooling and identity mapping) as the building components.

Besides, extensive studies [11–16] have proven that other advanced operations for *feature self-calibration*, such as *attention learning* and *dynamic convolutions*, can bring great benefits for representation learning. For example, Hu et al. [11] proposes Squeeze-and-Excitation Networks to explicitly model inter-dependencies between channels by learning channel-wise self-attention. Jia et al. [14] presents Dynamic Filter Networks to generate context-aware filters for increasing the flexibility and adaptiveness of networks. However, these useful feature calibration elements have *never* been well exploited in NAS, significantly limiting the potentials of NAS which aims to automatically discover more sophisticated and advanced network architectures without human engineering.

In this work, we aim to address this limitation by extending the search space of NAS with feature self-calibration operations for scaling up the search boundary. This makes a *heterogeneous search space*. Consequently, the way of feature tensor interaction and combination is dramatically diversified, from the conventional addition operator $\oplus$ only to the combination of addition $\oplus$, multiplication $\odot$ for attention modelling, and dynamic convolution $\circledast$. In this regard, we call the proposed method ***Neural Operator Search*** (NOS).

Such a search space enhancement is critical since NAS is enabled to explore stronger and previously undiscovered network architectures, which opens a door to potentially take the NAS research to the next level. In the *no free lunch* saying, this also comes with two new challenges: (i) It is non-trivial and more challenging to assemble such heterogeneous tensors and operations (i.e. features, attentions and dynamic weights) in a unified comput-

* Corresponding author.
E-mail addresses: w.li@qmul.ac.uk (W. Li), s.gong@qmul.ac.uk (S. Gong), xiatian.zhu@surrey.ac.uk (X. Zhu).

https://doi.org/10.1016/j.patcog.2022.109215

ing block, as compared to the conventional homogeneous feature-tensor-to-feature-tensor transformation; (ii) The search space increases exponentially which leads to a much harder NAS problem. To address the first challenge, we formulate a *heterogeneous operator cell* characterised by a novel heterogeneous residual block. This block, formulated in a residual learning spirit [6], is designed specially for fusing all the different types of tensors and operations synergistically. To solve the second challenge, we propose leveraging the *attention transfer* [17] idea to facilitate the search behaviour across this significantly larger network space via following the attention guidance of a pretrained teacher model. As we will show, this guidance not only makes the search more efficient but also improves the search result.

We make three **contributions** in this work:

1. We present a novel heterogeneous search space for NAS characterised by richer primitive operations including both conventional feature transformations and newly introduced feature self-calibration. This breaks the conventional selection limit of candidate neural networks and enables the NAS process to find stronger architectures, many of which are impossible to be discovered in the conventional space. This opens new territories for supporting stronger NAS algorithms and new possibilities for most expressive architectures ever to be revealed.
2. We formulate a novel Neural Operator Search (NOS) method dedicated for NAS in the proposed heterogeneous search space, with a couple of key designs – heterogeneous residual block for fusing different types of tensor operations synergistically and attention guided search for facilitating the search process over a vast search space more efficiently and more effectively.
3. With extensive comparisons to the state-of-the-art NAS methods, the experiments show that our approach is highly competitive on both CIFAR and ImageNet-mobile image classification tests.

2. Related work

Neural Architecture Search. Since the seminal work by Zoph and Le [1], neural architecture search has gained a surge of interest, effectively replacing laborious human designs by the computational process. From the ***strategy*** point of view, NAS methods can be categorised into two types: (1) proxy-based [1,8–10,18–25] and (2) proxy-less [26–31] NAS. Specifically, to alleviate the computational cost during search, the proxy-based NAS methods search for building cells on proxy tasks, with one or more of following compromised strategies: starting with fewer cells; using a smaller dataset (e.g. CIFAR-10); learning with fewer epochs. Then, to transfer to the large-scale target task, one can build a network by stacking searched cells without further exploration. However, suffering from lacking of directness and specialisation, the searched cells by proxy-based NAS methods are not guaranteed to be optimal on the target task. In contrast, proxy-less NAS methods directly learns architectures on a target task by starting with an over-parameterised network (*supernet*) that contains all possible paths, in which the redundant paths are pruned to derive the optimised architecture. Notwithstanding significantly better results than proxy-based approaches, proxy-less NAS methods require massive computational cost and GPU memory consumption, due to learning with the vast-size *supernet*. From the ***optimisation*** point of view, existing NAS methods usually fall into three groups: reinforcement learning (RL) based methods, evolutionary algorithm (EA) based methods, and gradient differentiable (GD) methods. In particular, RL-based NAS methods [1,8,27] control the selection of architecture components in a sequential order with policy networks. EA-based NAS methods [32–35,35,36] employ the validation accuracies to guide the evolution of a population of initialised architectures. RL- and EA-based NAS methods usually suffer from low efficiency and high computational resource demand, due to the fundamental searching challenge in a discrete space. In contrast, GD-based NAS methods [10,37–39] conduct searching over a continuous space by relaxation or mapping, substantially reducing the search cost to a few GPU days. Also, beyond studying optimisation and search strategies, recent NAS studies have extended towards memory efficiency [18], cost-effectiveness [19], and operation fairness [21,25]. Whilst varying in the algorithmic aspects, all these works commonly explore the *feature-transformation-only* search spaces without more diverse and advanced operations as we investigate here. To show the NAS potential of the proposed richer search space with self-calibration learning operations, we take the efficient proxy-based GD optimisation due to the resource constraint.

Neural Operator. Most of existing NAS works [1,8,10,27,37,38] optimise in a feature *feature-transformation-only* search space, considering the element-wise addition $\oplus$ operator alone. Recently, some NAS methods attempt to search with more complex neural operators. For example, van Wyk and Bosman [40] incorporate various mathematical operators in search space using efficient evolutionary search for image restorations. Mozejko et al. [41] consider additional *scalar multiplication operator* and *channel concatenation* for searching image denoising architectures. In this work, we instead focus on exploring self-calibration operations in NAS search space for deep convolutional networks. Self-calibration is a type of mechanism enabling a network to dynamically perform input-conditional self-adjustment, which has been studied extensively in both the computer vision [11,14,42,43] and natural language processing (NLP) literature [15,44]. There are two typical paradigms of self-calibration: *self-attention learning* and *dynamic convolutions*, realised via an *element-wise multiplication* operator $\odot$ and a *dynamic convolution* operator $\circledast$, respectively. Despite showing significant efficacy, self-calibration is only exploited *independently* after architecture hand-design [11] or auto-search [27]. We move a step further by fully exploring the potential of self-calibration along with feature transformation in joint optimisation, bringing a richer search space for neural architecture search. A few prior works also consider heterogeneous learning with application on domain adaptation [45] and graph neural networks [46]. In contrast, we focus on searching heterogeneous neural operators within a NAS search space.

Knowledge Distillation. There are recent works that use knowledge distillation to help computer vision and NLP tasks. Three types of knowledge are usually considered in distillation: features [47,48], attention [17,49], and predictions [50]. We leverage the attention distillation with a different objective – alleviating the intrinsic training-test discrepancy issue of the proxy-based NAS strategy, particularly with a more expressive search space. This represents a novel exploitation of attention distillation [17].

3. Method

In this section, we start by formulating a *heterogeneous search space* for NAS (Section 3.1), followed by a dedicated *heterogeneous operator cell* to enable composing the heterogeneous operations in a unified computing block with synergistic interaction and cooperation (Section 3.2). To overcome the intrinsic architecture discovery challenges from more expressive search space, we further develop an *attention guided search* scheme (Section 3.3).

3.1. Heterogeneous search space

To enrich the NAS search space so that more advanced network architectures can be discovered, we introduce a *heterogeneous search space* $\mathbb{A}$ that considers three different types of rep-

resentation learning capabilities: (1) *Feature transformations*; (2) *Attention learning*; and (3) *Dynamic convolutions*. More concretely, we form three sets of primitive computing operations that produce *features, attentions* and *dynamic weights*, respectively. This novel search space generalises the conventional counterpart which is limited to the first type of operations [8,10], and incorporates the self-calibration learning capabilities (i.e. the second and third types) in NAS. Importantly, while the search space changes, the generic search strategies still apply therefore being largely open for collaborating with existing NAS methods. For instance, in the proxy-based NAS strategy we may first search for a computing cell with heterogeneous operations as the building block and then form the final network architecture by sequentially stacking multiple such cells layer-by-layer.

Next, let us describe the heterogeneous primitive operation set $\mathcal{O}$ which consists of the following three disjoint subsets: $\mathcal{O}_f$, $\mathcal{O}_a$ and $\mathcal{O}_d$, along with their aggregation or application operators.

Feature Transformation Operations $\mathcal{O}_f$. We adopt the feature transformation/learning operation set $\mathcal{O}_f$ same as in [10,51], including the following 7 operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 average pooling, 3×3 max pooling, and identity. Every operation $o_f \in \mathcal{O}_f$ takes as input a feature tensor and outputs another feature tensor, i.e. *homogeneous* feature-tensor-to-feature-tensor transformation. For multiple feature tensor aggregation, the element-wise addition operator $\oplus$ is typically used.

Attention Learning Operations $\mathcal{O}_a$. Inspired by recent designs of attention learning modules [11,42,43], we form the $\mathcal{O}_a$ by considering two types of attention learning prototypes: *spatial-wise* and *channel-wise* attentions. Specifically, a *spatial-wise* attention operation learns a saliency map for an input feature tensor in order to calibrate the importance of different spatial positions. In contrast, a *channel-wise* attention operation produces a vector of scaling factors from the aggregated global context of an input tensor for adaptively calibrating the channel dependency. To enforce attentive calibration on feature tensor, the element-wise multiplication operator $\odot$ is a typical choice for both *spatial-wise* and *channel-wise* attentions.

Dynamic Convolution Operations $\mathcal{O}_d$. Dynamic convolutions, designed for the sake of self-adaptation, *generate* dynamic kernel weights in accordance with the input feature tensor. It is often in form of depth-wise separable convolution as the feature transformation operation. Tailored for either NLP or dense prediction tasks, existing dynamic convolution designs [14,15] are not suitable for image classification with different problem nature. It hence needs to be reformulated in order to be effective for learning discriminative image representations. We consider two design principles: (i) structurally lightweight whilst (ii) functionally strong and powerful with great modelling capability.

To that end, we present a novel dynamic convolution structure specialised for cost-effective image classification, as shown in Fig. 1. Concretely, it consists of three compact modules composed in a unified cooperation: (a) a *bottleneck* module, to compress an input feature tensor by a ratio of r; (b) a *kernel transform* module, to learn latent representations with a kernel dimension of $k \times k$; (c) a *kernel decode* module, to read out the dynamic kernel weights with the channel dimension same as the input feature tensor. This design is motivated, in part, by the long-range dependency modelling [52,53] and global context aggregation [11,54], elegantly integrating their merits via a unified formulation. For the output of dynamic convolutions, we consider two common kernel sizes: 3×3 and 5×5. In a depth-wise manner, we apply a standard or dilated convolution operator $\circledast$ to transform the input feature tensor. It is noteworthy to point out that, this type of convolutional kernel is *specific* for each feature tensor of a particular image sample (i.e. dynamic), rather than learned from a training dataset and *fixed* for all the input samples (i.e. static) as the conventional convolutional operations in the feature transformation set.

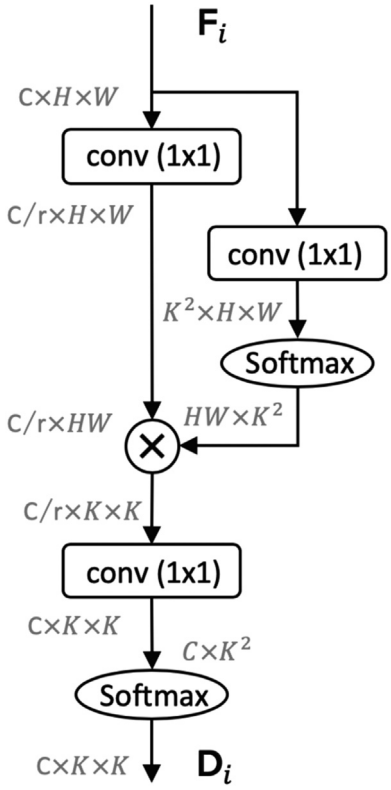

Fig. 1. Structure of the proposed dynamic convolutions for image classification. $\otimes$ denotes matrix multiplication.

3.2. Heterogeneous operator cell

Due to different natures of heterogeneous computing capabilities, a *unification* structure is needed for composing the primitive operations $\mathcal{O} = \mathcal{O}_f \cup \mathcal{O}_a \cup \mathcal{O}_d$ and aggregation/application operators $\mathcal{C} = \{\oplus, \odot, \circledast\}$ in such a way that their representation learning potentials can be well mined. To that end, we formulate a *heterogeneous operator cell*, a directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, joining conventional feature transformations and proposed self-calibration operations synergistically.

Formally, a heterogeneous operator cell consists of N ordered feature (tensor) nodes $\mathcal{V} = \{\boldsymbol{F}_k | , 1 <= k <= N\}$. Following [9], $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ are the outputs from the previous cells regarded as two *input nodes*, $\{\boldsymbol{F}_k\}_{k=3}^{N-1}$ denotes the *inner nodes* that perform computation, and the Nth node $\boldsymbol{F}_N$ is the cell *output node* formed as the concatenation of all the inner nodes, i.e. $\boldsymbol{F}_N = \texttt{concat}(\{\boldsymbol{F}_k\}_{k=3}^{N-1})$. The *edge* $e^{i \to k} = (i, k) \in \mathcal{E}$ specifies the connection between the ith and kth nodes (the information flow $i \to k$), associated with a specific operation $o^{i \to k}$ selected from the heterogeneous primitive operation set $\mathcal{O}$. The key is to design a computing block for the inner nodes with heterogeneous computations.

Heterogeneous Residual Block. It is non-trivial to design a heterogeneous computing block due to being *not* straightforward feature-tensor-to-feature-tensor transformation as in the conventional homogeneous operation. It involves self-calibrating the input feature tensor *itself* in addition to the homogeneous feature transformation. To facilitate adding the extra capacity, we formulate a heterogeneous residual block (see Fig. 2) characterised by a *surrogate node* k' in the computing block associated with each inner node k, for enabling richer feature tensor manipulations. This is in

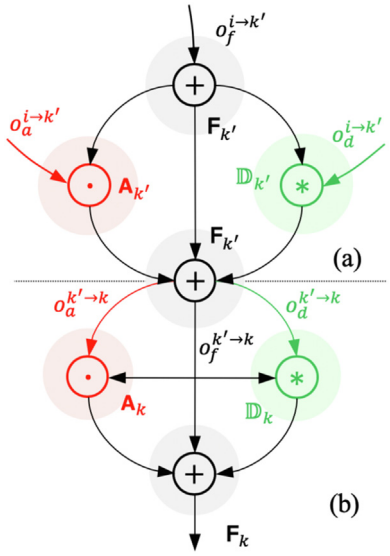

Fig. 2. Heterogeneous Residual block for formulating the inner node computation. (a) First-tier *individual* computation; (b) Second-tier *collective* computation.

a residual learning spirit [6], allowing to conduct self-calibration reliably.

Moreover, we design a two-tier computing hierarchy: the first tier for *individual* computation per input feature tensor to capture the specificity, and the second tier for *collective* computation on the set of all the input feature tensors as a whole to capture the intrinsic structural relations between feature tensors and the global input properties. The two tiers are connected by the surrogate node k'.

Formally, we take as input all the previous nodes $\{\boldsymbol{F}_i|, i < k\}$, process them separately with heterogeneous operations, and combine the processed results by summation (Fig. 2(a)) as:

$$\boldsymbol{F}_{k'} = \sum_{i<k} o_f^{i\to k'}(\boldsymbol{F}_i), \tag{2}$$

$$\boldsymbol{A}_{k'} = \sum_{i<k} o_a^{i\to k'}(\boldsymbol{F}_i), \tag{3}$$

$$\mathbb{D}_{k'} = \left\{o_d^{i\to k'}(\boldsymbol{F}_i)\right\}_{i<k} \tag{4}$$

where $\boldsymbol{F}_{k'}$, $\boldsymbol{A}_{k'}$, and $\mathbb{D}_{k'}$ are the three types of intermediate outputted tensors, i.e. features, attentions, and dynamic weights, respectively. These are subsequently aggregated into an *intermediate calibrated tensor*, i.e. the surrogate node $\boldsymbol{F}_{k'}$, using element-wise addition in-between on feature self-calibration and transformation as:

$$\boldsymbol{F}_{k'} = \underbrace{\boldsymbol{F}_{k'}}_{feature} \oplus \underbrace{(\boldsymbol{F}_{k'} \odot \boldsymbol{A}_{k'})}_{attention} \oplus \underbrace{\sum_{\boldsymbol{D}_{k'}\in\mathbb{D}_{k'}} \boldsymbol{F}_{k'} \circledast \boldsymbol{D}_{k'}}_{dynamic\ conv} \tag{5}$$

Next, $\boldsymbol{F}_{k'}$ is used as the input for the second-tier set-level collective computation (Fig. 2(b)). Likewise, we consider the same three types of operations:

$$\boldsymbol{F}_k = o_f^{k'\to k}(\boldsymbol{F}_{k'}), \tag{6}$$

$$\boldsymbol{A}_k = o_a^{k'\to k}(\boldsymbol{F}_{k'}), \tag{7}$$

$$\mathbb{D}_k = \left\{o_d^{k'\to k}(\boldsymbol{F}_{k'})\right\}, \tag{8}$$

and form the inner node $\boldsymbol{F}_k$ via further feature self-calibration and transformation as:

$$\boldsymbol{F}_k = \underbrace{\boldsymbol{F}_k}_{feature} \oplus \underbrace{(\boldsymbol{F}_k \odot \boldsymbol{A}_k)}_{attention} \oplus \underbrace{\sum_{\boldsymbol{D}_k\in\mathbb{D}_k} \boldsymbol{F}_k \circledast \boldsymbol{D}_k}_{dynamic\ conv} \tag{9}$$

In doing so, our heterogeneous residual block presents a two-tier combinatorial operations structure for each inner node, resulting in a more expressive search space (see Section 4.2).

3.3. Attention guided search optimisation in a heterogeneous search space

To showcase the effectiveness of the proposed heterogeneous search space and operator cell, we adopt the proxy-based NAS strategy, due to the computing resource constraints and the enormous search space. This search is done by constructing a small proxy network parametrised by $\boldsymbol{\Theta}$.

Attention Guided Search. Compared with proxyless search strategy, proxy-based NAS is more efficient but relatively less optimal due to *not* directly optimising the final network architecture. This training-test discrepancy problem can be worsened when the search space provides more flexibility and combinatorial capability, such as the proposed space. To solve this obstacle, we propose attention guided search, which optimises the proxy network in a knowledge distillation manner with an external guidance injected from a pre-trained teacher network into the NAS process. Importantly, this can accelerate the search process by reducing the training time of each proxy network.

Specifically, we leverage the attention transfer idea [17] that encourages a student (the proxy network in our case) to hierarchically imitate a teacher's hidden attention knowledge. Intuitively, this may benefit the search for self-calibration learning. Formally, let us denote a feature tensor at the jth stage of the teacher and student network as $\boldsymbol{F}_T^j$ and $\boldsymbol{F}_S^j$, separately. Attention transfer is realised by imposing an alignment loss function across the two networks as:

$$\mathcal{L}_{AT} = \frac{1}{2}\sum_{j\in\mathcal{J}} \| \frac{\boldsymbol{x}_S^j}{\|\boldsymbol{x}_S^j\|_2} - \frac{\boldsymbol{x}_T^j}{\|\boldsymbol{x}_T^j\|_2} \|_2, \tag{10}$$

where $\boldsymbol{x}_{S/T}^j = vec(\sum_i |\boldsymbol{F}_{S/T}^j(\cdot,\cdot,i)|^2)$ is the spatial-wise accumulated feature vector. An overview of attention guided search is depicted in Fig. 3.

Optimisation. For NAS optimisation, we adopt the DARTS method [10]. In our context, we conduct the continuous relaxation over all the possible heterogeneous operations $\mathcal{O}$ for making a continuous search space:

$$\overline{o}^{i\to j}(x) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\boldsymbol{a}_o^{i\to j}\right)}{\sum_{o'\in\mathcal{O}}\exp\left(\boldsymbol{a}_{o'}^{i\to j}\right)} o(x), \tag{11}$$

where an architecture vector $\boldsymbol{a}_o^{i\to j} \in \mathbb{R}^{|\mathcal{O}|}$ is used for each possible connection $i \to j$. We summarise the architecture vector of all the connections as a matrix $\boldsymbol{A} = \left[\boldsymbol{a}^1, \cdots, \boldsymbol{a}^{|\mathcal{E}|}\right] \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{O}|}$. With this relaxation, we can jointly optimise the architecture parameters $\boldsymbol{A}$ and the network weights $\boldsymbol{\Theta}$ in a fully gradient differentiable manner.

Equipped with the proposed attention guidance search, the search objective function is finally formulated as the following bilevel optimisation process:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_{train}(\boldsymbol{\Theta}, \boldsymbol{A}) + \lambda\mathcal{L}_{AT}(\boldsymbol{\Theta}, \boldsymbol{A}), \tag{12}$$

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} \mathcal{L}_{val}(\boldsymbol{\Theta}^*, \boldsymbol{A}) + \lambda\mathcal{L}_{AT}(\boldsymbol{\Theta}^*, \boldsymbol{A}), \tag{13}$$

where λ denotes the weighting hyper-parameter. For the first level Eq. (12), we learn the optimal parameters $\boldsymbol{\Theta}^*$ for a given architecture $\boldsymbol{A}$ w.r.t a training objective $\mathcal{L}_{train}$ and the attention alignment

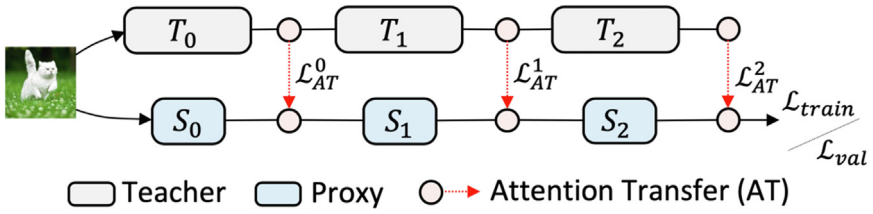

Fig. 3. Overview of attention guided search. T_i and S_i ($i \in \{0, 1, 2\}$) denote the ith stage of the teacher and proxy (student) networks.

Table 1
Evaluating the feature self-calibration operations on CIFAR10 and CIFAR100 with Top-1 and Top-5 error rate (%).

Model	Type	Kernels	CIFAR10		CIFAR100		FLOPS(M)	#Params(MB)
			Top-1	Top-5	Top-1	Top-5		
ResNet-18	-	-	4.95	0.22	23.61	7.16	555.42	11.17
+ Dynamic	Normal	3	4.63 [a]	0.13 [a]	22.63 [a]	6.44 [a]	+ 3.85	+ 0.03
		5	4.78 [a]	0.14 [a]	23.45 [a]	6.82 [a]	+ 7.62	+ 0.04
	Dilated	3	4.97 [b]	0.23 [b]	24.00 [b]	7.28 [b]	+ 3.85	+ 0.03
		5	4.92 [a]	0.17 [a]	23.75 [b]	7.20 [b]	+ 7.62	+ 0.04
+ Attention		Spatial	4.79 [a]	0.16 [a]	23.51 [a]	7.04 [a]	+ 1.08	+ 0.01
		Channel	4.83 [a]	0.19 [a]	23.20 [a]	6.89 [a]	+ 0.40	+ 0.15

[a] Better than the baseline.
[b] Worse than the baseline.

loss $\mathcal{L}_{AT}$. The second level Eq. (13) then explores the optimal architecture $\boldsymbol{A}^*$ over the heterogeneous search space $\mathbb{A}$ w.r.t a validation objective $\mathcal{L}_{val}$ and $\mathcal{L}_{AT}$. For image classification, $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ usually take the cross-entropy loss function.

Search Outcome. Once the above alternated optimisation is done, we derive an amenable cell architecture with heterogeneous operators. In practice, for each heterogeneous computing block we retain the top-2 strongest incoming operations with at least one feature transformation operation for the first-tier (Fig. 2(a)), and the top-1 strongest operation for the second-tier (Fig. 2(b)).

4. Experiments

Datasets. We evaluated the proposed NOS method on image classification using three common datasets. *CIFAR10/100:* Both CIFAR10 and CIFAR100 have 50K/10K train/test RGB images of size $32 \times 32 \times 3$, categorised into 10 and 100 classes, respectively [55]. *ImageNet:* We used the ILSVRC2012 version for large-scale image classification evaluation, containing 1.28M training images and 50K validation samples from 1000 object classes [56].

We first conduct preliminary experiments on CIFAR10/100 to select the heterogeneous primitive operations $\mathcal{O}$. To test the efficacy and transferability of NOS, we search the cell structures on CIFAR10 only, and compare the performance with existing methods on CIFAR10/100 and ImageNet.

4.1. Study of feature self-calibration operations

We conducted a controlled experiment to test the introduced self-calibration operations on CIFAR-10 and CIFAR-100. Specifically, for the proposed *dynamic convolutions*, we considered both normal and dilated convolutions and two kernel sizes (3×3 and 5×5). We adopted the channel-wise and spatial-wise *attention learning*. For the baseline model, we used ResNet-18 [6] with 4 stages in the backbone. To build a model with self-calibration, we added each self-calibration operation at the stages 1, 2, 3 of ResNet-18, respectively. For fair comparison, we trained each model in the same setting (see Appendix A). In 1, we summarised the model parameters and FLOPs in addition to the test set performance (error rates). We observed that: (1) Both attention operations and our normal dynamic convolutions outperform the baseline consistently; (2) Adding dilated dynamic convolutions causes performance drop in most cases. We hence exclude it from the candidate set; (3) Very marginal FLOPs and parameters increase from these self-calibration operations over the baseline, suggesting their high cost-effectiveness.

4.2. Cell search

Search Space. Following the setup of existing methods [10,32,51,61], we searched the convolutional architectures on CIFAR10. We constructed a small proxy network with 8 heterogeneous operator cells, and two reduction cells at 1/3 and 2/3 of the total network depth for feature shape reduction. Figure 4 illustrates the general model architecture. As found out above, the heterogeneous primitive operation set $\mathcal{O}$ contains 11 operations in total: $|\mathcal{O}_f| = 7$ feature transformation operations, $|\mathcal{O}_a| = 2$ attention learning operations, $|\mathcal{O}_d| = 2$ dynamic convolutions, respectively. We constructed the proposed heterogeneous operator cell ($\mathcal{G} = (\mathcal{V}, \mathcal{E})$) with $|\mathcal{V}| = 7$ nodes (2 input nodes, 4 inner nodes and 1 output node). So, all 4 heterogeneous residual blocks contain $|\mathcal{E}| = 18$ edges in total (14 first-tier connections and 4 second-tier connections). To derive the final cell architecture, we kept 2 first-tier connections and 1 second-tier connection for each block. As a result, there is a total number of $\prod_{n=1}^{4} \frac{(n+1)n}{2} \times 11^3 \approx 10^{14}$ possible choices, 5 orders of magnitude larger than the conventional size of $\prod_{n=1}^{4} \frac{(n+1)n}{2} \times 7^2 \approx 10^9$ as in [10,37,59].

Training. Following the setup of existing methods [10,32,51,61], we searched the convolutional architectures on CIFAR10. We constructed a small proxy network with 8 heterogeneous operator cells, and two reduction cells at 1/3 and 2/3 of the total network depth for feature shape reduction (see Appendix C). We used 25K images split from the training set for validation. We randomly initialised the architecture parameters $\boldsymbol{A} \in \mathbb{R}^{18 \times 11}$ in the normal distribution. We used a pre-trained PyramidNet-110 (bottleneck, $\alpha = 84$) [57] as the teacher model. We set the weight $\lambda = 10^3$ for attention guidance loss $\mathcal{L}_{AT}$. After 25 epochs of training on the proxy network, we derived the final heterogeneous operator cells from the architecture matrix $\boldsymbol{A}$. See Appendix A for more configurations for training the proxy and teacher networks.

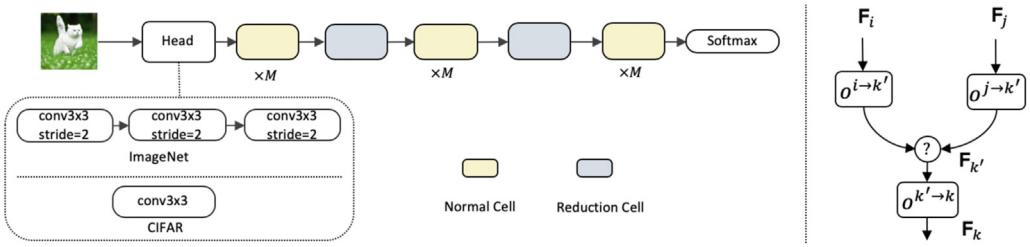

Fig. 4. (**Left**) The overall model architecture for CIFAR-10 and ImageNet, consisting of repeated Normal Cells and Reduction Cells. M is the stacking choice for the number of Normal Cells. Each cell contains 4 blocks. (**Right**) An example of two-tier block construction in cell: Each block takes two input features ($\boldsymbol{F}_i$, $\boldsymbol{F}_j$) from previous nodes; The operator (?) in a block is determined by the choices of two operations ($o^{i\to k'}$, $o^{j\to k'}$) in the first-tier; An extra operation ($o^{k'\to k}$) is selected in the second-tier.

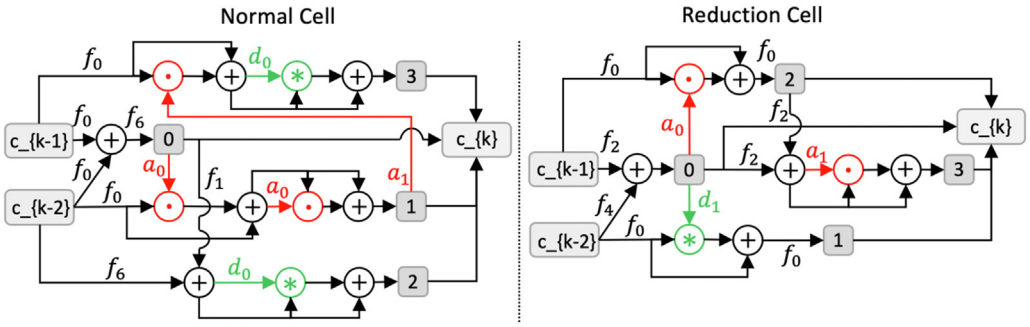

Fig. 5. Normal cell and reduction cell searched on CIFAR-10. f_0: `sep_conv_3x3`, f_1: `sep_conv_5x5`, f_2: `dil_conv_3x3`, f_4: `max_pooling`, f_6: `identity`, a_0: `spatial_attention`, a_1: `channel_attention`, d_0: `dynamic_conv_3x3`, d_1: `dynamic_conv_5x5`..

The search on CIFAR10 took only 8.4 h using a single NVIDIA Tesla V100 GPU. The searched heterogeneous operator cells by NOS is shown in Fig. 5, in which the self-calibration operators ⊙ and ⊛ appear in both first-tier and second-tier. For example, there are two *attention* operations in first-tier and two *dynamic convolutions* in second-tier in the normal cell.

4.3. Architecture evaluation

CIFAR. To measure the final image classification performance of the searched heterogeneous operator cells on CIFAR10 and CIFAR100, we created an evaluation network with 20 cells, 36 initial channels, and an auxiliary tower with loss weight 0.4. See Appendix A for more configurations for training the evaluation network. Due to high variance of results on CIFAR, we conducted 10 independent runs and reported both the best and average results. We summarised the results of NOS and the state-of-the-art proxy-based NAS methods[1] in Table 2. The comparisons show that: **(1)** NOS achieves the second best result on CIFAR10, whilst enjoying the smallest model parameters (only 2.6M). This suggests significant cost-effectiveness and compactness advantages of our method, in comparison to FairDARTS [21] and P-DARTS [20]. **(2)** Despite a significantly larger search space (10^{14} *vs.* 10^9 in [10,37,59,61]), NOS still yields high cost-effectiveness (only 0.35 GPU day). **(3)** Further, NOS reaches the best result on CIFAR100 by directly transferring the CIFAR10 searched network, outperforming P-DARTS [20] and GDAS [59] significantly. This challenging cross-dataset test indicates a superior transferability of the network searched by NOS.

ImageNet. To evaluate the transferability of architecture discovered by NOS on the large-scale ImageNet, we used the mobile setting same as in [10,37,59], where the number of multiply-add operations is restricted to be less than 600M at the input size of 224×224. Specifically, we constructed an evaluation network with 14 cells and 48 initial channels. An auxiliary tower with loss weight 0.4 was also applied. See Appendix A for more training details. Table 3 shows the ImageNet results in the mobile setting. Notably, compared to other state-of-the-art proxy-based NAS using gradient optimisation (GHN [58], DARTS [10], SNAS [37], GDAS [59], and BayesNAS [60]), the network searched by NOS on CIFAR10 can be successfully transferred. NOS discovers a cell structure that performs better with higher efficiency and lighter model (only 440M FLOPs). In addition, compared with latest state-of-the-art such as P-DARTS [20] and FairDARTS [21], NOS achieves a strong trade-off in terms of model performance, size and search cost, despite a much larger search space (10^{14} *vs.* 10^9 with P-DARTS and FairDARTS).

Table 2
Comparisons with the state-of-the-art architectures obtained by proxy-based NAS methods on CIFAR10 and CIFAR100 with Top-1 error rate (%).

Architecture	**Error (%)**		Params	Search Cost	
	CIFAR10	CIFAR100	(M)	GPUs	Days
PyramidNet [57][a]	3.92	20.11	2.5	-	-
ENAS [8]	2.89	-	4.6	1	0.5
DARTS(1st) [10]	3.00±0.14	-	3.3	1	1.5
DARTS(2nd) [10]	2.76±0.09	17.54	3.3	1	4.0
SNAS (moderate) [37]	2.85±0.02	-	2.8	1	1.5
GHN [58]	2.84±0.07	-	5.7	1	0.84
GDAS [59]	2.93	18.38	3.4	1	0.84
BayesNAS [60]	2.81±0.04	-	3.4	1	0.2
ASNG [61]	2.83±0.14	-	3.9	1	**0.11**
P-DARTS [20]	**2.50**	16.55	3.4	1	0.3
FairDARTS [21]	2.54	-	2.8	1	3
Random Baseline[b]	3.85	21.66	2.4	-	-
NOS (best)	2.53	**16.21**	2.6	1	0.35
NOS (average)	2.67±0.06	16.72±0.24	2.6	1	0.35

[a] The teacher model.
[b] Best architecture among 30 random samples.

[1] In our evaluation context, we primarily aim to accurately evaluate the effect of search space. To this end, we selectively compared with a set of more related state-of-the-art NAS methods, rather than exhaustively. For instance, we excluded ProxylessNAS [26] due to that it uses a different search strategy with a *supernet* (orthogonal to the search space factor).

Table 3
Comparisons with the state-of-the-art proxy-based architectures on ImageNet-mobile.

Architecture	Test Err. (%)		Params	×+	Search Cost
	top-1	top-5	(M)	(M)	(GPU-days)
GHN [58]	27.0	8.7	6.1	569	0.84
DARTS [10]	26.7	8.7	4.7	574	4.0
SNAS [37]	27.3	9.2	4.3	522	1.5
GDAS [59]	26.0	8.5	5.3	581	0.84
BayesNAS [60]	26.5	8.9	**3.9**	-	**0.2**
P-DARTS [20]	**24.4**	**7.4**	4.9	557	0.3
FairDARTS [21]	**24.4**	**7.4**	4.4	**440**	3
NOS	25.8	8.1	4.0	**440**	0.35

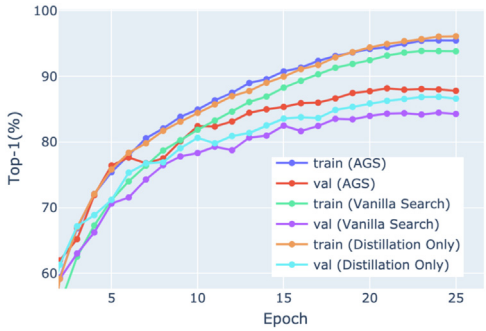

Fig. 6. The train and val set accuracies on CIFAR10.

Table 4
Testing attention guided search (AGS). w/o: without, w/: with.

AGS	**Test Error (%)**	
	CIFAR10	CIFAR100
w/o	3.44	18.80
w/	2.53	16.21

4.4. Further analysis

We evaluated attention-guided search (AGS) on CIFAR10/100 by comparing a NOS variant without attention transfer loss. The same training setting was used (Appendix A). We used a pretrained PyramidNet-110 as the teacher. Table 4 shows that learning with attention guidance can significantly benefit the NOS search process.

Distillation Effect. We examined the effect of knowledge distillation in the proposed Attention Guided Search (AGS). We conducted this analysis on CIFAR10. In this evaluation, we compared three methods: (1) Vanilla Search: Using the original DARTS search method; (2) Distillation Only: Using the attention transfer loss for training the proxy network only; (3) AGS: The proposed method (full). We tracked the model performance on both training (train) and validation (val) data sets. Figure 6 shows that (i) knowledge distillation brings a positive performance gain over the vanilla DARTS and (ii) using attention guidance for the network search can further improve the searched architecture. This suggests that distillation is effective to alleviate the architecture training-test discrepancy issue involved in the proxy-based NAS.

Space Generality. We tested the general effect of the proposed Attention Guided Search (AGS) using the original DARTS search space ($\mathcal{O}_f$ + zero operation) on CIFAR10. During the search process, we followed the same settings as DARTS with the first-order optimisation. We obtained the error rates: 3.00 $\pm$ 0.14 (DARTS) *vs.* 2.92 $\pm$ 0.05 (DARTS + AGS). This suggests a general efficacy of AGS over different search spaces.

Operation Effect. We further evaluated the heterogeneous operation effect using different search space combinations ($\mathcal{O}_f$, $\mathcal{O}_f \cup \mathcal{O}_a$, $\mathcal{O}_f \cup \mathcal{O}_d$, and $\mathcal{O}_f \cup \mathcal{O}_a \cup \mathcal{O}_d$). Here, AGS with the first-order optimisation is applied for the search process. Figure 7 shows that: (1) Both attention learning operations $\mathcal{O}_a$ and dynamic convolution operations $\mathcal{O}_d$ can bring non-trivial performance gain over feature learning operations; (2) The combination of $\mathcal{O}_a$ and $\mathcal{O}_d$ gives further accuracy boost. This validates the efficacy of our proposed heterogeneous operation search space.

5. Conclusion

In this work, we presented Neural Operator Search (NOS), characterised by a heterogeneous search space for neural architecture search (NAS). Specifically, NOS further introduces dynamic convolution and attention learning operations on top of the conventional feature transformation operations. This proposed search space expansion enables NAS to discover more expressive and previously undiscovered architectures, significantly expanding the search horizon and enriching the possible search outcomes. Moreover, we introduced a heterogeneous operator cell to integrate these different operations synergistically. To facilitate the learning process, we further proposed an attention guided search mechanism in a distillation manner. Extensive evaluations have validated the superiority of our method over a wide range of state-of-the-art NAS models on the standard image classification tasks.

Limitation. This work has a couple of limitations. *First*, we focus on studying the heterogeneous search space of NAS while leaving the heterogeneous model scaling behaviour to future work. *Second*, we mainly evaluate the quality of searched architectures by fine-tuning on ImageNet-1K image classification following existing works. The architectural transferability on other tasks (e.g., object detection and segmentation) may need further investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

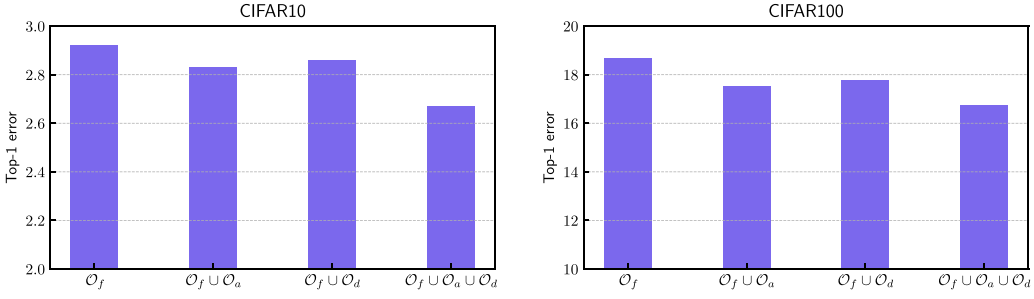

Fig. 7. Evaluating the heterogeneous operation search space on CIFAR10 and CIFAR100.

Acknowledgment

This work is supported by the China Scholarship Council, the Alan Turing Institute, and Innovate UK Industrial Challenge Project (98111-571149). We are especially grateful to the QMUL ITS Research group for their support.

Appendix A. Details of training configurations

A1. CIFAR

ResNet-18 and PyramidNet-110 (bottleneck, $\alpha = 84$). We followed the same training details as [57]. We trained these models for 300 epochs with batch size 32. The learning rate was initialised as 0.025, which was decayed by 10 every 30 epochs. The standard SGD optimiser with momentum of 0.9 was employed. We set a weight decay value of 1×10^{-4} to avoid overfitting. The data augmentations include horizontal flip and random crop.

Cell Search. For network parameters Θ of proxy network, we used SGD with an initial learning rate 0.025 and set the momentum value as 0.9. This learning rate was decayed to 0 with a cosine scheduler. A weight decay value of 3×10^{-4} was imposed to avoid over-fitting. For learning architecture matrix $\boldsymbol{A}$, we used the Adam optimiser with a fixed learning rate value 6×10^{-4} and set the weight decay to 1×10^{-3}.

Cell Evaluation. The evaluation network was trained from scratch directly for 600 epochs with batch size 128. Note that, the attention transfer was **not** involved for training. We set the weight decay values for CIFAR-10 and CIFAR-100 to 3×10^{-4} and 5×10^{-4} individually. The standard SGD optimiser with a momentum of 0.9 was applied. The initial learning rate was 0.25, decayed to 0 with a cosine scheduler. Following existing works [8–10,32], we performed two additional enhancements: the cutout regularisation [62] with length 16 and the drop-path [63] of probability 0.3.

A2. ImageNet

We trained the evaluation model for ImageNet using SGD optimiser for 300 epochs with batch size 512. We initialised the learning rate as 0.25 and reduced it to 0 by a linear scheduler. Learning rate warmup [64] was applied for the first 5 epochs to deal with the large batch size and learning rate.

References

[1] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, in: Proc. Int. Conf. on Learn. Rep., 2017.
[2] Y. Sun, B. Xue, M. Zhang, G.G. Yen, A particle swarm optimization-based flexible convolutional autoencoder for image classification, IEEE Trans. Neural Netw. Learn. Syst. 30 (8) (2018) 2295–2309.
[3] G. Ghiasi, T.-Y. Lin, Q.V. Le, NAS-FPN: learning scalable feature pyramid architecture for object detection, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019.
[4] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A.L. Yuille, L. Fei-Fei, Auto-DeepLab: hierarchical neural architecture search for semantic image segmentation, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019.
[5] V. Nekrasov, H. Chen, C. Shen, I. Reid, Fast neural architecture search of compact semantic segmentation models via auxiliary cells, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019.
[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016.
[7] D. Mellouli, T.M. Hamdani, J.J. Sanchez-Medina, M.B. Ayed, A.M. Alimi, Morphological convolutional neural network architecture for digit recognition, IEEE Trans. Neural Netw. Learn. Syst. 30 (9) (2019) 2876–2885.
[8] H. Pham, M.Y. Guan, B. Zoph, Q.V. Le, J. Dean, Efficient neural architecture search via parameter sharing, in: Proc. Int. Conf. Mach. Learn., 2018.
[9] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8697–8710.
[10] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: Proc. Int. Conf. on Learn. Rep., 2019.
[11] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7132–7141.
[12] L. Bertinetto, J.F. Henriques, J. Valmadre, P. Torr, A. Vedaldi, Learning feed-forward one-shot learners, in: Proc. Neur. Info. Proc. Sys., 2016, pp. 523–531.
[13] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual attention network for image classification, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 3156–3164.
[14] X. Jia, B. De Brabandere, T. Tuytelaars, L.V. Gool, Dynamic filter networks, in: Proc. Neur. Info. Proc. Sys., 2016, pp. 667–675.
[15] F. Wu, A. Fan, A. Baevski, Y.N. Dauphin, M. Auli, Pay less attention with lightweight and dynamic convolutions, in: Proc. Int. Conf. on Learn. Rep., 2019.
[16] X. Zhu, D. Cheng, Z. Zhang, S. Lin, J. Dai, An empirical study of spatial attention mechanisms in deep networks, arXiv (2019).
[17] S. Zagoruyko, N. Komodakis, Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer, in: Proc. Int. Conf. on Learn. Rep., 2017.
[18] H. Zhang, Y. Li, H. Chen, C. Shen, Memory-efficient hierarchical neural architecture search for image denoising, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020.
[19] D. Zhou, X. Zhou, W. Zhang, C.C. Loy, S. Yi, X. Zhang, W. Ouyang, EcoNAS: finding proxies for economical neural architecture search, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020.
[20] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: bridging the depth gap between search and evaluation, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1294–1303.
[21] X. Chu, T. Zhou, B. Zhang, J. Li, Fair DARTS: eliminating unfair advantages in differentiable architecture search, in: Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 465–480.
[22] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, C. Xu, ReNAS: relativistic evaluation of neural architecture search, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 4411–4420.
[23] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, X. Chang, Block-wisely supervised neural architecture search with knowledge distillation, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020.
[24] D. Zhou, X. Jin, X. Lian, L. Yang, Y. Xue, Q. Hou, J. Feng, AutoSpace: neural architecture search with less human interference, in: Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 337–346.
[25] X. Chu, B. Zhang, R. Xu, FairNAS: rethinking evaluation fairness of weight sharing neural architecture search, in: Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 12239–12248.
[26] H. Cai, L. Zhu, S. Han, ProxylessNAS: direct neural architecture search on target task and hardware, in: Proc. Int. Conf. on Learn. Rep., 2019.
[27] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, MnasNet: platform-aware neural architecture search for mobile, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2820–2828.
[28] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, FBNet: hardware-aware efficient convnet design via differentiable neural architecture search, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10734–10742.
[29] T.-J. Yang, Y.-L. Liao, V. Sze, NetAdaptV2: efficient neural architecture search with fast super-network training and architecture optimization, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 2402–2411.
[30] K. Yu, R. Ranftl, M. Salzmann, Landmark regularization: ranking guided super-net training in neural architecture search, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 13723–13732.
[31] J. Peng, J. Zhang, C. Li, G. Wang, X. Liang, L. Lin, Pi-NAS: improving neural architecture search by reducing supernet training consistency shift, in: Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 12354–12364.
[32] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: AAAI Conference on Artificial Intelligence, 2019.
[33] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, K. Kavukcuoglu, Hierarchical representations for efficient architecture search, in: Proc. Int. Conf. on Learn. Rep., 2018.
[34] Y. Sun, B. Xue, M. Zhang, G.G. Yen, Completely automated CNN architecture design based on blocks, IEEE Trans. Neural Netw. Learn. Syst. (2019).
[35] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, C. Xu, CARS: continuous evolution for efficient neural architecture search, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020.
[36] Y. Ci, C. Lin, M. Sun, B. Chen, H. Zhang, W. Ouyang, Evolving search space for neural architecture search, in: Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 6659–6669.
[37] S. Xie, H. Zheng, C. Liu, L. Lin, SNAS: stochastic neural architecture search, in: Proc. Int. Conf. on Learn. Rep., 2019.
[38] R. Luo, F. Tian, T. Qin, E. Chen, T.-Y. Liu, Neural architecture optimization, in: Proc. Neur. Info. Proc. Sys., 2018, pp. 7816–7827.
[39] W. Li, S. Gong, X. Zhu, Neural graph embedding for neural architecture search, in: AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 4707–4714.
[40] G.J. van Wyk, A.S. Bosman, Evolutionary neural architecture search for image restoration, 2019, pp. 1–8.
[41] M. Mozejko, T. Latkowski, L. Treszczotko, M. Szafraniuk, K. Trojanowski, Superkernel neural architecture search for image denoising, in: Workshop of IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 484–485.
[42] W. Li, X. Zhu, S. Gong, Harmonious attention network for person re-identification, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 2285–2294.
[43] J. Park, S. Woo, J.-Y. Lee, I.S. Kweon, BAM: bottleneck attention module, in: Proc. Bri. Mach. Vis. Conf., 2018.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. Neur. Info. Proc. Sys., 2017, pp. 5998–6008.
[45] X. Yang, C. Deng, T. Liu, D. Tao, Heterogeneous graph attention network for unsupervised multiple-target domain adaptation, tpami (2020).
[46] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, Y. Ye, Heterogeneous graph structure learning for graph neural networks, in: AAAI, vol. 35, 2021, pp. 4697–4705.
[47] J. Yim, D. Joo, J. Bae, J. Kim, A gift from knowledge distillation: fast optimization, network minimization and transfer learning, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4133–4141.
[48] C. Liu, Z. Cao, W. Li, Y. Xiao, S. Du, A. Zhu, Exploiting distilled learning for deep siamese tracking, in: IEEE International Conference on Pattern Recognition, 2020, pp. 577–583.
[49] W. Li, S. Gong, X. Zhu, Hierarchical distillation learning for scalable person search, Pattern Recognit. 114 (2021) 107862.
[50] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv (2015).
[51] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in: Proc. Eur. Conf. Comput. Vis., 2018, pp. 19–34.
[52] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7794–7803.
[53] Y. Cao, J. Xu, S. Lin, F. Wei, H. Hu, GCNet: non-local networks meet squeeze-excitation networks and beyond, arXiv (2019).
[54] J. Hu, L. Shen, S. Albanie, G. Sun, A. Vedaldi, Gather-excite: exploiting feature context in convolutional neural networks, in: Proc. Neur. Info. Proc. Sys., 2018, pp. 9401–9411.
[55] A. Krizhevsky, et al., Learning Multiple Layers of Features from Tiny Images, Technical Report, Citeseer, 2009.
[56] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[57] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5927–5935.
[58] C. Zhang, M. Ren, R. Urtasun, Graph hypernetworks for neural architecture search, in: Proc. Int. Conf. on Learn. Rep., 2019.
[59] X. Dong, Y. Yang, Searching for a robust neural architecture in four GPU hours, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1761–1770.
[60] H. Zhou, M. Yang, J. Wang, W. Pan, BayesNAS: a bayesian approach for neural architecture search, in: Proc. Int. Conf. Mach. Learn., 2019.
[61] Y. Akimoto, S. Shirakawa, N. Yoshinari, K. Uchida, S. Saito, K. Nishida, Adaptive stochastic natural gradient method for one-shot neural architecture search, in: Proc. Int. Conf. Mach. Learn., 2019.
[62] T. DeVries, G.W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv (2017).
[63] G. Larsson, M. Maire, G. Shakhnarovich, FractalNet: ultra-deep neural networks without residuals, 2017.
[64] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large minibatch SGD: training imagenet in 1 hour, arXiv (2017).

Wei Li received his PhD degree at Queen Mary University of London. His research interests include person re-identification, object detection, and deep learning.

Shaogang Gong is Professor of Visual Computation at Queen Mary University of London (since 2001), a Fellow of the Institution of Electrical Engineers and a Fellow of the British Computer Society. He received his DPhil (1989) in computer vision from Keble College, Oxford University. His research interests include computer vision, machine learning and video analysis.

Xiatian Zhu is a Senior Lecturer with Surrey Institute for People-Centred Artificial Intelligence, and Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, UK. He received his Ph.D. degree from the Queen Mary University of London. He won the Sullivan Doctoral Thesis Prize 2016. His research interests include computer vision, and machine learning.