Pattern Recognition 147 (2024) 110052

Contents lists available at ScienceDirect

Pattern Recognition

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

Neural architecture search: A contemporary literature review for computer vision applications

Matt Poyser *, Toby P. Breckon

Department of Computer Science, Durham University, UK

ARTICLE INFO

Keywords:
Neural architecture search
Classification
Detection
Segmentation

ABSTRACT

Deep Neural Networks have received considerable attention in recent years. As the complexity of network architecture increases in relation to the task complexity, it becomes harder to manually craft an optimal neural network architecture and train it to convergence. As such, Neural Architecture Search (NAS) is becoming far more prevalent within computer vision research, especially when the construction of efficient, smaller network architectures is becoming an increasingly important area of research, for which NAS is well suited. However, despite their promise, contemporary and end-to-end NAS pipeline require vast computational training resources. In this paper, we present a comprehensive overview of contemporary NAS approaches with respect to image classification, object detection, and image segmentation. We adopt consistent terminology to overcome contradictions common within existing NAS literature. Furthermore, we identify and compare current performance limitations in addition to highlighting directions for future NAS research.

1. Introduction

Recent acceleration within the deep learning domain [1] naturally follows the increased availability of public datasets that stems from the emergence of big data. Unsurprisingly, the complexity of the proposed network architectures is also increasing. As such, manually searching through this architecture space is less and less feasible, and we must rely on domain expertise to identify suitable networks for a given application. Neural Architecture Search (NAS) automatically traverses the architecture search space for a given task, and generates models that are competitive alongside hand-crafted state-of-the-art architectures.

Recent NAS capability within the image classification domain is demonstrably powerful [2], with generated convolutional neural network (CNN) models achieving accuracies of 97.57% (CIFAR-10 [3]) and 76.2% (ImageNet [4]). This is comparable to leading image classification performance [5]. However, there is relatively little NAS development outside pure CNN generation (e.g. transformer [6] and generative adversarial networks [7]), for which hand-crafted network architectures perform so well. Similarly, within the computer vision domain, considerations of NAS beyond image classification are underdeveloped; NAS for object detection and image segmentation architectures for example, receive less interest than their hand-crafted counterparts. To this end, we present a comprehensive review of recent NAS advancements, to best facilitate further insight and research in this area. We build upon existing – although now dated – surveys [8–12], that fail to consider NAS for computer vision outside of CNN image classification. Concurrent work [13,14] presents a less comprehensive overview to which the reader may find it helpful to refer. In either case, we hope that this review serves to best illustrate the contributions of prior NAS literature.

Furthermore, as NAS rapidly evolves within several distinct domains, ambiguities and inconsistencies have arisen in several methodological descriptions. Consequently, it is increasingly difficult for researchers approaching the NAS domain to meaningfully engage with the field and to clearly explain any proposed methodology with reasonable reproducibility. With this shortcoming in mind, we adopt a common terminology (following that of TuNAS [15]), to improve the comprehensibility and reproducibility of future NAS research. On this basis, our contributions are as follows:

- a systematic review providing the most comprehensive NAS survey of image classification, object detection, and image segmentation domains to date.
- an overview and taxonomy that for the first time uses consistent terminology across *all* contemporary NAS literature, resolving the ambiguities and inconsistencies emanating from the original NAS works.
- analysis of the NAS literature offering insights into the direction of promising future research.

* Corresponding author.
E-mail address: matthew.poyser@durham.ac.uk (M. Poyser).

https://doi.org/10.1016/j.patcog.2023.110052
Received 10 March 2023; Received in revised form 2 October 2023; Accepted 13 October 2023
Available online 24 October 2023

1.1. Review organisation

We first divide this review by computer vision task into image classification, object detection and image segmentation NAS methodologies. Subsequently, image classification is subdivided by NAS strategy into three key sub-areas: weight-sharing, gradient-based, and prediction-based, each of which correspond to a NAS search speed-up strategy. In all cases, we prioritise explaining how a given NAS method fits into the evolution of the NAS domain, while maintaining consistent terminology, to the best advantage of incoming researchers.

2. Neural architecture search: A quick overview

Rather than conventionally hand-crafting a neural network architecture, wherein layer operations are hand-selected and defined at each layer, NAS seeks to automatically generate the network architecture best suited to a given task, given a set of available operations. An overview of the general NAS process is illustrated in Fig. 1, which can be split into two key stages. First, the *search phase* involves traversing all architectures within the *search space*. Once the top performing architecture (or top-k architectures) is identified (termed *final searched architecture(s)*), it is retrained from scratch (the *evaluation phase*).

With the rise of NAS, a multitude of recent literature has addressed the scalability challenge which occurs due to the resultant large search space of all potential neural network architectures and their respective training costs. The seminal work of Zoph et al. [16] demonstrates the capability of recurrent neural networks (RNN) with reinforcement learning to generate network architectures automatically, with their network architecture outperforming existing hand-crafted state-of-the-art network architectures both in accuracy and speed for both image classification and language modelling tasks.

Since this influential paper [16], interest and research in NAS has accelerated [8]. Reinforcement learning [17–20] methods, as well as evolutionary [21–25] approaches have since been developed. Notably, MnasNet [19] adopts a reinforcement learning search strategy for both image classification and object detection, incentivised towards minimising inference latency. Unmistakeably however, such NAS search strategies are prohibitively expensive. More recently, and with more success in this regard, gradient and predictor based approaches have been developed [26–29], often in conjunction with weight-sharing techniques to improve convergence rate [30–32], notably by eliminating the need to train each architecture in the search space separately.

As such, we limit the literature covered by this survey to more recent NAS solutions for image scene analysis, where the training cost falls within a reasonable computational time limit. Moreover, several NAS *architecture* datasets exist that facilitate validation of the NAS framework performance, rather than their generated networks. Notably among these, NAS-Bench-101 [33] contains 5 million trained and evaluated models. NAS-Bench-201 [34] and NATS-Bench [35] also evaluate architecture size and topology, with a fixed architecture search space but more diagnostic information compared to NAS-Bench-101. An overview of literature covered by this review can be found in Tables 1 and 2, which provide a fair comparison (where possible) across common NAS application domains. In all cases, however, it is important to adhere to best practices when producing a NAS pipeline [36]. Radosavovic et al. [37] demonstrate that the process of constructing the search space is critical.

3. Image classification

Image classification is the primary challenge domain in which NAS operates within the computer vision field and for which existing literature is most comprehensive.

The reader may find it helpful to refer to the following key concepts:

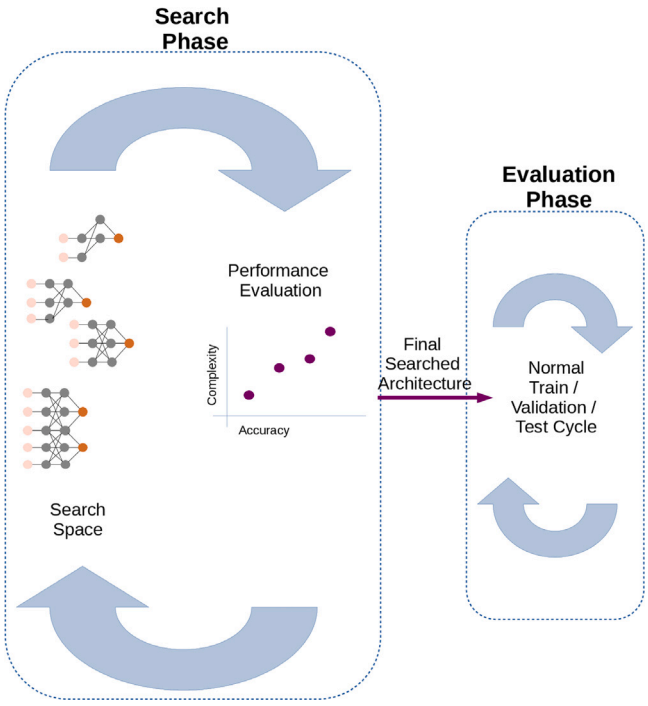

Fig. 1. An overview visualisation of the NAS process consistent across reinforcement learning, gradient-based and prediction-based approaches.

- *Optimisation-gap* — The performance after retraining a network architecture from scratch does not necessarily reflect its performance after the search phase. This is because 'shortcuts' are taken (e.g. weight sharing) to significantly improve NAS search speed. See Section 3.2.2.1 for more information within the context of gradient-based NAS approaches.
- *DAG (Directed Acyclic Graph)* — Reformulating a network architecture as a graph (see Fig. 2a), where nodes represent a layer, and edges represent the operations within a layer (e.g. convolution).
- *Search space* — The total set of architectures that will be considered and possibly evaluated by the NAS approach. The aim is to find the best possible architecture in the architecture space. In all cases, this architecture space is prohibitively large and it is impossible to evaluate the performance of each architecture. The NAS paradigm aims to optimise how search strategies iterate over the search space. For instance, the influential NASBench-101 [33] search space contains 423,000 unique architectures.
- *Final searched architecture* — The architecture that the NAS search phase produces. The NAS strategy considers this architecture to be the best architecture in the entire search space. The evaluation phase will retrain this architecture from scratch to generate a final model.
- *Topology* — As opposed to width and depth, topology refers to the operations (layers) within an architecture.

We provide an overview of NAS paradigms in the following order:

1. *Weight-sharing (Section 3.1)* — Introduced by ENAS [30]. By sharing weights between architectures during the NAS search phase, candidate architectures do not have to be trained separately to completion. Overall network architecture training costs during the NAS search phase can be substantially reduced.
2. *Gradient-based (Section 3.2)* — By relaxing the architecture space such that it is continuous (see the seminal DARTS [26] method),

Table 1
Overview of NAS approaches, their performance, and the general methodology employed: Evolutionary Algorithms (EA), Reinforcement Learning (RL), Gradient-Based (GB), Weight-Sharing (WS) and Prediction (Pred). † entails object detection. ‡ entails instance segmentation. Otherwise, reported results are for Image Classification. Results correspond to the results reported in their respective original paper, even when subsequent papers report higher performance or results generated using more comparable computational resources.

Reference	Technique							Top-1 Acc (CIFAR-10) (%)	Params (CIFAR-10)	Top-1 Acc (ImageNet) (%)	Params (ImageNet)
	Evolutionary algorithms	Reinforce-ment Learning	Gradient based	Prediction	Weight sharing	Sample based	One shot				
Zoph & Le [16] (2017)		✓				✓		96.35	37.4M	n/a	n/a
MnasNet [19] (2019)		✓				✓		n/a	n/a	76.7M	5.2M
ENAS [30] (2018)		✓			✓		✓	97.11	4.6M	n/a	n/a
SNAS [31] (2018)		✓			✓		✓	97.02	2.9M	72.7	4.3M
CAS [38] (2019)		✓			✓		✓	n/a	n/a	n/a	n/a
CNAS [32] (2020)		✓			✓		✓	97.40	3.7M	75.40	5.3M
BigNAS-S [39] (2020)					✓		✓	n/a	n/a	76.5	4.5M
BigNAS-M [39] (2020)					✓		✓	n/a	n/a	78.9	5.5M
BigNAS-L [39] (2020)					✓		✓	n/a	n/a	79.5	6.4M
BigNAS-XL [39] (2020)					✓		✓	n/a	n/a	80.9	9.5M
ProxylessNAS-R [40] (2018)		✓			✓		✓	97.70	5.8M	74.6	n/a
ProxylessNAS-G [40] (2018)			✓		✓		✓	97.92	5.7M	74.2	n/a
NASP [41] (2020)			✓		✓		✓	97.56	7.4M	73.7	9.5M
TuNAS [15] (2020)		✓			✓		✓	n/a	n/a	75.0	n/a
AttentiveNAS (largest) [42] (2021)					✓		✓	n/a	n/a	80.7	n/a
RNSGA-Net1 [43] (2022)	✓							96.11	3.01	n/a	n/a
Stage-Wise NAS [44] (2020)					✓	✓		95.68	7.27M	n/a	n/a
PAD-NAS [45] (2022)	✓				✓		✓	n/a	n/a	76.1	4.7M
GLiT-Tiny [46] (2021)	✓				✓		✓	n/a	n/a	76.3	7.2M
GLiT-Small [46] (2021)	✓				✓		✓	n/a	n/a	80.5	24.6M
GLiT-Base [46] (2021)	✓				✓		✓	n/a	n/a	82.3	96.1M
NEAS [47] (2021)	✓				✓		✓	n/a	n/a	80.0	n/a
BONAS [48] (2020)	✓				✓			97.57	3.3M	74.6	4.8M
DARTS [26] (2019)			✓		✓		✓	97.24	3.3M	73.3	4.7M
I-DARTS [49] (2019)			✓		✓		✓	97.63	3.8M	75.7	n/a
Wu et al. [50] (2020)			✓		✓		✓	97.50	3.5M	75.33	5.7M
E-DNAS [51] (2020)			✓		✓		✓	n/a	n/a	76.9	5.9M
ISTA-NAS [52] (2020)			✓		✓		✓	97.64	3.37M	76.0	5.65M
BMTAS [53] (2020)			✓		✓		✓	n/a	n/a	n/a	n/a
SMASH [54] (2018)			✓		✓		✓	94.47	4.6M	61.38	16.2M
unsupervised: DARTS [55] (2020)			✓		✓		✓	97.44	3.6M	n/a	n/a
FairNAS [56] (2021)	✓				✓		✓	98.2	n/a	77.5	5.9M
Pi-NAS [57] (2021)			✓		✓		✓	n/a	n/a	81.60	27.1M
EnTranNAS [58] (2021)			✓		✓		✓	97.78	7.68M	75.70	7.2M
EnTranNAS-DST [58] (2021)			✓		✓		✓	97.52	3.20M	76.20	7.0M
DOTS [59] (2021)			✓		✓		✓	97.51	3.5M	76.0	5.3M
Shapley-NAS [2] (2022)					✓		✓	97.57	3.6M	76.1	5.4M
P-DARTS [27] (2019)			✓		✓		✓	97.50	3.5M	75.9	5.4M
PC-DARTS [60] (2019)			✓		✓		✓	97.43	3.6M	75.8	5.3M
FP-DARTS [61] (2023)			✓		✓		✓	97.50	3.4M	76.3	5.3M
R-DARTS($l2$) [62] (2020)			✓		✓		✓	97.49	n/a	n/a	n/a
SETN [63] (2019)			✓		✓		✓	97.31	4.6M	74.3	5.4M
DNA [64] (2020)			✓		✓		✓	98.30	n/a	78.40	6.4M
Distribution Consistent [65] (2022)	✓				✓		✓	n/a	n/a	79.50	n/a
BossNAS [66] (2021)			✓		✓		✓	n/a	n/a	82.5	n/a
GDAS [67] (2019)			✓		✓		✓	97.07	3.4M	74.0	5.3M
GDAS-NSAS [68] (2020)			✓		✓		✓	97.27	3.54M	n/a	n/a
Landmark Regularization:SPOS [69] (2021)			✓		✓		✓	n/a	n/a	67.38	4.77M
Landmark Regularization:GDAS [69] (2021)			✓		✓		✓	n/a	n/a	68.82	5.07M

(*continued on next page*)

Table 1 (*continued*).

Landmark Regularization:NAO [69] (2021)			✓		✓		✓	n/a	n/a	68.89	4.49M
TAS [70] (2019)			✓		✓		✓	94.00	n/a	76.20	n/a
NetAdaptV2 [71] (2021)			✓		✓		✓	n/a	n/a	77.0	n/a
FBNet-C [72] (2019)			✓		✓		✓	n/a	n/a	74.9	5.5M
PNAS [73] (2019)				✓				96.59	3.2M	74.2	5.1M
PNAS-Large [73] (2019)				✓				n/a	n/a	82.9	86.1M
NAO [74] (2018)			✓			✓		96.82	10.6M	74.3	11.35M
NAO with pseudo morphological operations [75] (2022)			✓			✓		97.35	n/a	n/a	n/a
GBDT-NAS [76] (2020)				✓				n/a	n/a	76.6	5.7M
ReNAS [77] (2021)				✓				n/a	n/a	n/a	n/a
MdeNAS [28] (2019)				✓				97.45	3.61M	74.5	6.1M
NASWOT [29] (2021)				✓				n/a	n/a	n/a	n/a
NASBOT [78] (2018)				✓				91.31	n/a	n/a	n/a
Auto-Keras [79] (2019)				✓				96.40	n/a	n/a	n/a
BayesNAS [80] (2019)				✓				97.59	3.4M	73.5	3.9M
BANANAS [81] (2019)				✓				n/a	n/a	n/a	n/a
								COCO			
								AP		Params	
MnasNet [19] (2019) †		✓						23.0		4.9M	
DetNAS [82] (2019) †	✓				✓		✓	42.0		n/a	
SpineNET-49S [83] (2020) †		✓				✓		41.5		12M	
SpineNET-190 [83] (2020) †		✓				✓		52.1		163.6M	
NATS [84] (2019) †			✓		✓		✓	38.4		n/a	
NAS-FPN (AmoebaNet Backbone) [85] (2019) †		✓				✓		48.4		166.5M	
Auto-FPN [86] (2019) †			✓		✓		✓	44.3		n/a	
NAS-FCOS [87] (2020) †		✓				✓		46.1		89.4M	
OPANAS [88] (2021) †	✓				✓		✓	41.6		29.8M	
								mIOU (cityscapes)			
DPC [89] (2018) ‡								82.7			
Auto-DeepLab [90] (2019) ‡			✓		✓		✓	82.1			
DCNAS [91] (2021) ‡			✓		✓		✓	84.3			
EDNAS [51] (2020) ‡	✓							n/a			

conventional gradient-descent methods can be used to optimise the search space efficiently.[1]

3. *Prediction-based (Section 3.3)* — Introduce some auxiliary method to evaluate network architecture performance without training, circumventing training iterations and consequent high NAS search costs entirely.

The three NAS paradigms are further subdivided within each section by strategies with shared characteristics.

[1] Sometimes referred to as differentiable neural architecture search (DNAS).

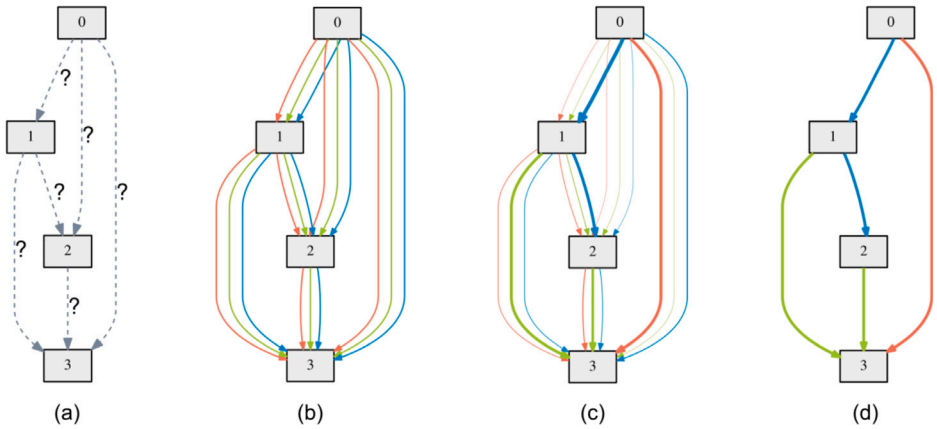

Fig. 2. DARTS search phase pipeline. (a) The architecture search space can be realised as a Directed Acyclic Graph (DAG): edges between nodes represent possible operations (e.g. convolution, max-pooling, etc.). (b) A layer is a softmax over all possible operations within the search space. (c) Using SGD, this super-network is trained in relation to which operations perform best, as well as the weights of the operations themselves. (d) A final searched network is selected from the top performing operations at each layer.

Table 2
Some literature reports the NAS search phase prediction performance in place of or as well as final searched architecture performance. In these cases we present the findings on NAS-Bench-201 on CIFAR-10 and ImageNet-16-120. † denotes experiments conducted on NATS-Bench, the successor to NAS-Bench-201.

NAS search algorithm	CIFAR-10	ImageNet-16-120
I-DARTS [49] (2019)	93.76	41.44
SE-NAS [92] (2021)	93.47 ± 0.14	45.66 ± 1.05
unsupervised: DARTS [55] (2020)	94.18 ± 0.24	46.27 ± 0.37
FairNAS [56] (2021)	93.23 ± 0.18	42.19 ± 0.31
Pi-NAS [57] (2021)	93.83 ± 0.00	n/a
Shapley-NAS [2] (2022)	94.37 ± 0.00	46.85 ± 0.12
Landmark Regularization:SPOS [69] (2021)	93.41 ± 0.43	n/a
Landmark Regularization:GDAS [69] (2021)	94.32 ± 0.28	n/a
Landmark Regularization:NAO [69] (2021)	93.53 ± 0.43	n/a
Distribution Constrained [65] (2022)	94.29 ± 0.07	46.41 ± 0.14
BossNAS [66] (2021) †	93.29	n/a
ReNAS [77] (2021)	93.99 ± 0.25	45.97 ± 0.49
RMI [93] (2022)	94.28 ± 0.10	46.34 ± 0.00
NASWOT [29] (2021)	92.96 ± 0.81	44.44 ± 2.10
FreeRea [94] (2023)	94.36 ± 0.00	46.34 ± 0.00

3.1. Weight-sharing

Weight-sharing approaches, first proposed in ENAS [30], reduce training time through the transfer learning of weights learnt for previously sampled architectures. In general, only a single network needs to be trained to convergence via the use of this technique. Across all weight-sharing NAS approaches, this single network represents the *entire* architecture search space, and is referred to as a "super-network". Subsequent sampled network architectures during the NAS search phase thereafter inherit initialisation weights from this super-network, and they require few or zero training epochs before their performance is sufficiently evaluated and ranked. As such, the total search phase cost of NAS is drastically minimised. Since only the super-network is trained to convergence during the NAS search phase, we refer to this approach as a *one-shot* method.[2] In general, one-shot methods aim to rank architecture performance using their shared weights [95] relative to *each other* rather than their absolute performance. Once the highest-ranked architecture is determined during the search phase, it is retrained and then fully optimised for the given task.

[2] "Weight-sharing" is a broad term to denote how different architectures considered by a given NAS methodology do not use independent weights. "One-shot" (or equivalently "super-network" under our terminology) methods necessarily employ weight-sharing but not vice-versa.

In their influential paper [30], ENAS first constructs a single DAG to represent the entire search space. First they fix the weights of the NAS controller (which determines which nodes and operations are sampled from the architecture search space). The entire super-network weights can be updated from the gradient of a single sampled cell (child architecture). This process is repeated until one entire pass of the dataset has been completed. Next, they again sample child network architectures, but this time their weights are fixed (using transfer learning from the previously learned weights) and instead train the controller using reinforcement learning. They proceed to alternate training the child network architectures and controller for several iterations, at which point the best-performing model is sampled, initialised with random weights, and retrained from scratch without the use of transfer learning weights.

Given an architecture α, its optimal parameters $\omega^*(\alpha)$ can be obtained by $\omega^*(\alpha) = \text{argmin}_\omega \mathcal{L}(\alpha, \omega)$, where $\mathcal{L}$ is the loss function. The performance of α can be measured by some metric $\mathcal{R}(\alpha, \omega^*(\alpha))$, such as its accuracy [30] or loss [31] on validation data, or even latency. The goal of reinforcement learning can thus be described as maximising the expectation of the reward $\mathcal{R}(\alpha, \omega^*(\alpha))$ to find an optimal policy, yielding a bi-level optimisation problem with policy π and search space Ω:

$$\begin{aligned} &\max_\theta \mathbb{E}_{\alpha\sim\pi(\alpha;\theta,\Omega)}\mathcal{R}(\alpha, \omega^*(\alpha)) \\ &s.t \quad \omega^*(\alpha) = \text{argmin}_\omega \mathcal{L}(\alpha, \omega) \end{aligned} \tag{1}$$

3.1.1. Improving the final architecture performance of weight-sharing NAS

Following the success of weight-sharing in [30], the most obvious next step is to find an architecture from the search space that is closer in performance to the best performing architecture. SNAS [31] identifies that ENAS assumes a Markov Decision Process that delays the reinforcement learning reward for architecture changes during the ENAS search phase. Using a differentiable reward function better rewards structural architecture decisions, improving search efficiency. CAS [38] develops a NAS learning paradigm whereby the cell structure evolves when trained on new datasets and domains, without loss in performance on the previous dataset. By introducing constraints on the learned weights, 'knowledge' is projected in an orthogonal direction, so that prior knowledge related to that of the previous dataset/domain is not lost.

CNAS [32] employs curriculum learning within the NAS search phase to improve the tractability of the objective function, yielding better quality architectures. The search space is divided into a series of smaller architecture spaces, where the number of searched operations is gradually increased. Curriculum learning overcomes the phenomenon where it is difficult for conventional one-shot methods [30] to satisfy the reinforcement learning objective function during early training epochs.

BigNAS [39] finds better architectures through addressing the optimisation-gap issue prevalent within NAS by adding regularisation and a novel initialisation strategy. If the super-network is sufficiently trained, BigNAS demonstrates that a simple grid-search strategy is enough to traverse the architecture search space efficiently. Additionally, network architectures can be sampled under memory or latency constraints.

3.1.2. Improving weight-sharing NAS speed

Despite the significant search stage acceleration that weight-sharing offers, the computation costs of architecture search can still be prohibitively expensive. ProxylessNAS [40] demonstrates the effectiveness of subsampling only a single path through the DAG at each iteration. Binary gates are introduced to simulate a mask for a given activation path during a given training iteration. Consequently, training is more stable and can be performed directly on large datasets such as ImageNet. Previous methods were only able to achieve meaningful performance on large datasets by first training on a smaller, proxy dataset. Two variants, ProxylessNAS-R and ProxylessNas-G are presented, which use reinforcement learning and gradient-based methods (described below) to traverse the architecture search space. Similarly, NASP [41] use an approach derived from the proximal algorithm [96] to ensure that only one operation in the DAG is updated with each iteration, drastically improving convergence rate.

Several search methods consider shrinking the NAS architecture space [15,42,44,45,97] without loss in performance in order to reduce the time it takes to traverse it. TuNAS [15] simulates smaller filter sizes via masking, which reduces the need to train architectures that only differ in filter size. Additionally, the number of available operations that can be selected is gradually reduced during the first 25% of epochs in the search phase. The authors additionally propose a novel reward function, as well as more aggressive weight-sharing.

AttentiveNas [42] improves upon the BigNAS search strategy by reducing the architecture space size in a Pareto-aware fashion. The best Pareto architectures achieve better accuracy than every other architecture in the search space with the same or less computational consumption. The worst Pareto architectures are dominated in performance by all other architectures with the same computational cost. Sampled candidate network architectures are only trained if they lie on the best or worst Pareto front. With this strategy, it is additionally trivial to impose a computational limit on the generated network architectures. Similarly, RNSGA-Net [43] adopts the R-NSGA-II genetic algorithm [98] to search for pareto-optimal architectures within a given region of interest.

Stage-Wise NAS [44] discard architectures according to a simple heuristic. By dividing architectures into ‘stages,’ the importance of a given stage can be identified. Fewer layers are attributed to stages with lower importance and the depth of the network architecture is progressively increased during the NAS search phase.

ANASOD [97] reduces the architecture search space without loss in performance via approximate operation distribution encoding. On the basis that there is little difference in performance between architectures with only slight differences, the architectures sharing a distribution of operations map to the same encoding. Using any other NAS strategy to search over these distribution *encodings* rather than the entire operation search space makes NAS optimisation more tractable.

PAD-NAS [45] reduce the architecture space by pruning operations. The pareto-optimal architectures (and the next best, according to non-domination rank [99]) are identified according to accuracy and latency. Operations that are not prevalent within these best architectures are pruned. As such, a one-shot training strategy in conjunction with evolutionary search based upon NSGA-II [99] is adopted.

3.1.3. Improving the weight-sharing NAS architecture space

Alongside improving search speed and accuracy of NAS approaches, the architecture space itself can be improved beyond CNN architecture search. GLiT [46] employs a one-shot NAS approach to optimise *transformer* architecture for image classification. Building upon a Multi-Head Attention (MHA) block as the basis for the search space, a locality module is further introduced such that each searched MHA cell has a varying distribution of convolution-based locality modules (capturing local information within an image) and self-attention modules (capturing global information within an image). Adopting SPOS [100], GLiT divides the search space into disjointly searching for a) optimal distribution of local and global sub-modules and b) detailed architecture of modules given optimal distribution in (a). As a result, a vast transformer architecture search space can be efficiently searched without compromising memory requirements.

Conversely, NEAS [47] searches for the best *ensemble* of classifiers. Where GLiT adopts SPOS to divide the search space, NEAS instead shrinks the search space by first computing an approximation for architecture similarity and operator quality. The worst performing ensemble architectures with respect to diversity and quality are then dropped. During the search phase, weights are shared between the lowest layers of the remaining ensemble networks at a given iteration.

It should be noted that weight-sharing techniques do not necessitate one-shot methodology. BONAS [48] identifies the sensitivity to network initialisation of one-shot approaches, as well as high memory requirements. To this end, Bayesian Optimisation is used to identify similar network architectures. Weights are shared amongst these similar network architectures so that they can be trained simultaneously.

3.2. Gradient-based

Until now, we have considered weight-sharing techniques that optimise the architecture topology without updating the weights inherited from the super-network. Differentiable approaches [26,27,101] build upon the weight-sharing technique through the application of stochastic gradient descent and other well-used deep learning techniques by relaxing the search space such that it is continuous. Consequently, the convergence rate of the architecture is drastically improved. In general, this approach proffers the fastest architecture search without significantly impacting performance, but at the expense of a high GPU-memory intensity.

DARTS [26] constructs a shallow super-network in which each layer is a softmax over all possible operations within the architecture search space (Fig. 2a). This allows traversal of the search space with gradient descent (Fig. 2b). Once the super-network is trained (Fig. 2c), they extract the top-k best-performing operations at each layer, and ‘evaluate’ a deep neural network under the resultant restricted architecture search space to produce a final optimised model (Fig. 2d).

3.2.1. DARTS-like NAS approaches

I-DARTS [49] builds upon DARTS by instead considering a softmax operation before operations rather than after them. This removes the restriction upon the final model to consider at most one operation between each layer, thereby widening the search space, without reducing convergence rate. We can perceive such an approach as Mixed-Path in that it does not require a single operation between each node (Single-Path, [26,40]), or multiple (but with a consistent number of) operations between each node (Multi-Path, [102,103]), see Fig. 3. Wu et al. [50] adopt an alternative Mixed-Path approach. Rather than using softmax to relax the super-network graph into a continuous DAG, each intermediate node output is computed as a scaled linear combination of the *feature maps* of the previous nodes. By using Sparse Group Lasso regularisation [104], nodes and operations may be filtered out such that there is no rigid constraint on the node or path structure. E-DNAS [51] further introduces flexibility within the proposed architecture by explicitly searching for the optimal kernel size as well as the weights of convolutional layers.

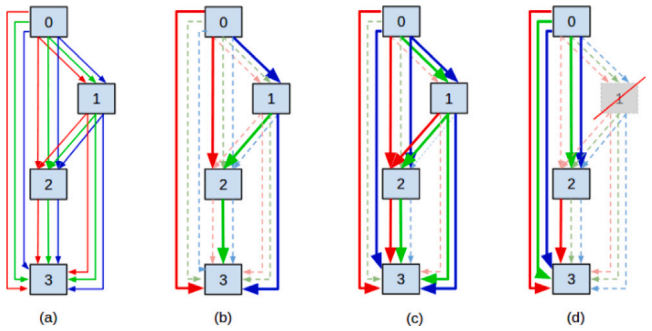

Fig. 3. Different NAS architecture configurations as presented by Wu et al. [50]. (a) A NAS super-network where each coloured edge denotes a single operation, and all possible operations are considered between nodes through continuous relaxation (softmax layer). (b) Single-Path architecture; exactly one operation is selected in the final searched architecture. (c) Multi-Path architecture; exactly n operations are considered and aggregated between each and every node (for some pre-defined n). (d) Mixed-Path architecture; no limitations on operation number between given nodes in the final searched architecture.

ISTA-NAS [52] adopts an alternative strategy compared to [50] towards fulfilling the sparsity constraint. By projecting the continuous relaxation of operations onto the sparse constraint, its LASSO formulation [105] can be solved with the ISTA [106] algorithm. Consequently, ISTA-NAS enables the same size (width, depth, batch-size) super-network to be used in both the NAS search and evaluation phase, due to the sparse and more efficiently encoded super-network. Additionally, this has the benefit of minimising the optimisation-gap problem (see below). DNAL [107] introduce a novel Scaled Sigmoid activation function that can be utilised alongside existing NAS approaches, wherein the sparsity constraint is imposed on the activation function rather than the architecture parameters themselves, improving performance.

SE-NAS [92] improve DARTS search efficiency by shrinking the operation search space. After initial training stages of the supernetwork, operations with lower importance are more likely to exist in weaker architectures and can be more readily discarded. Due to weight sharing between architectures however, operator importance can be easily misrepresented. Therefore, if there is little variance between architecture performance within a layer, confidence in the architecture ranking is presumed worse, and operations at these layers are retained.

BMTAS [53] employs a NAS pipeline within a differentiable search space best adapted for multi-task learning. Through masking (to simulate training one sub-network architecture at a time) and a novel resource-aware objective function, their pipeline formulates and traverses the search space in a way that promotes general purpose features (operations) within the final NAS-generated architecture.

SMASH [54] uses an auxiliary HyperNetwork [108] network to generate the weights of the network architecture itself. A super-network is generated to encompass the architecture search space. Akin to conventional gradient-based NAS solutions, an architecture is sampled from the super-network. However, where its weights would normally be inherited directly, instead they are generated by a HyperNetwork trained a-priori.

Yan et al. [55] employ a variational graph isomorphism autoencoder before traversing the architecture search space. They conclude that this autoencoder out-performs state-of-the-art autoencoders [109, 110] and best captures the local structure information of neural network architectures such that similar structures cluster better in the latent space. Traversing the search space such that the next most similar, unevaluated network architecture is evaluated in the next iteration, they are able to smooth the NAS search phase, leading to better overall performance.

Simon et al. [111] adapt DARTS such that convolutional layers have an additional noise injection module. Weights associated with this module learn how much noise to inject into a given input such that DARTS successfully trains in the presence of label noise. Indeed, the results indicate that the modified DARTS method achieves superior performance in the presence of noise, without sacrificing performance on clean data.

3.2.2. Addressing DARTS-like strategy drawbacks

Despite the multitude of developments directly upon DARTS, it is not without its drawbacks [48,112], and has received heavy criticism. To some extent, such problems can be minimised by careful super-network training schemes, including additional batch normalisation, prevention of over-regularisation, and reduced dropout [95]. Fair-NAS [56] formally identifies weaknesses in training a one-shot model in general, and proposes strict fairness (all single paths through the super-network are attributed equal optimisation). Their work is deployable atop all two-stage NAS strategies, which they choose to demonstrate within an evolutionary search strategy based on NSGA-II [99].

Yu et al. [112] identify the importance of random seed during the search phase. Indeed some search strategies, especially DARTS, perform worse than random with respect to some seeds. The authors' findings align with [95] in that the *ranking* of architectures yields a poor reflection of their performance after the evaluation phase. Further, weight-sharing is detrimental to the NAS search phase. Finally, good performance from architectures can be attributed to the heavy search space restrictions, such that even random search over the space yields high performing architectures.

3.2.2.1. The optimisation-gap. The predominant issue with DARTS however, is perhaps the optimisation-gap problem, in which searched architecture performance does not necessarily correlate with its performance after being re-trained during the NAS evaluation phase. One obvious reason for this is that one-shot methods are ranking networks relative to each other rather than their absolute performance [95]. The super-network architecture in the NAS search phase generally differs considerably from a derived sub-network architecture in the NAS evaluation phase. In fact, many NAS approaches only search for a cell structure during the search phase (owing to available computational resources), which is then stacked before being retrained during the NAS evaluation phase.

Pi-NAS [57] addresses the optimisation-gap by considering the image inputs to the NAS solution rather than directly addressing NAS operation and topology selection strategies. Introducing negative samples to a training iteration draws benefits from the well-founded contrastive learning domain. This ensures correct loss descent to deliver a more accurate architecture ranking. Furthermore, a given input image is augmented four times to be passed through separate super-network paths to yield better architecture ranking. As such, the optimisation-gap is minimised as final searched architectures better resemble their standalone performance.

EnTranNAS-DST [58] further addresses the optimisation-gap by representing non-derived connections in the final searched model as zero-weighted connections. As such, the propagation of the super-network in the NAS search phase is the same as the propagation of the final searched architecture in the NAS evaluation phase, eliminating the gap between the two.

3.2.2.2. The relationship between operation and topology. DARTS derives its topology from the best performing operation, but optimal operation selection does not guarantee optimal topology. Indeed, with rank correlation analysis, DOTS [59] shows that a given searched cell within DARTS can be sub-optimal for precisely this reason. To this end, by decoupling operation search and topology search, they yield more optimal final searched topology.

Shapley-NAS [2] reconsiders how to derive the final searched architecture from a super-network trained by DARTS. Given that there is often a complex relationship between operations, simply selecting the strongest operation at each edge is flawed. Instead, the Shapley score [113,114] for each operation can be approximated to quantify its contribution. Moreover, employing Shapley score in place of gradient descent better trains the super-network during the search phase.

3.2.2.3. Skip-connections in DARTS. P-DARTS [27] builds on the DARTS method, and minimises the optimisation-gap by progressively growing the NAS network depth during the search phase. However, they identify that skip-connections within searched architectures are commonly considered optimal by NAS approaches because they minimise the instability and difficulty of training a deep network during the early stages of NAS search [115]. As a result however, the final architecture performance deteriorates. To compensate, P-DARTS incorporates operation-level dropout after skip connections, as well as a cap on the number of skip connections that can occur in the final architecture. The authors achieve over 1% lower error rate while offering an order of magnitude lower search time than DARTS, and a two-orders-of-magnitude lower search time than its predecessors.

PC-DARTS [60] corroborates that the weight-free operations (skip-connections, max-pooling) are prevalent within NAS-generated architectures since they increase training stability in early NAS iterations. To alleviate this, the authors suggest an alternative approach, whereby only a few of the available operations are considered at each epoch. However, this in turn introduces further instability into the training process, which is ameliorated by applying edge normalisation (with negligible computational overhead). FP-DARTS [61] similarly reduce considered operations by dividing the operation space into two and searching on each operation set in parallel. The final architecture is then derived from both searched architectures. Zela et al. [62] additionally attribute the prevalence of skip connections within DARTS-generated architectures to exploding eigenvalues during the NAS search phase. By increasing $l2$ regularisation when the dominant eigenvalue exceeds a threshold, DARTS performance is increased across the board.

SETN [63] refer to this phenomenon as the "Matthew effect". Quickly converging architectures appear as better sampling candidates: they consist of fewer convolution layers, and thus perform poorly overall after retraining from scratch [62,116] (for instance a surplus of skip-connections at the expense of convolution layers). Consequently, the best performing network architectures when fully trained might be ignored. SETN adopts a single-path weight-sharing approach (Fig. 3b), but acknowledge that randomly sampling the path [54,95] can lead to an unnecessary consideration of poorly performing network architectures. Instead, a stochastic operation and input selection strategy is proposed that avoids the Matthew effect, while simultaneously adopting an evaluator to minimise the selection of poorly performing architectures.

3.2.2.4. Detrimental weight inheritance. Weight-sharing may lead to a poor evaluation of an architecture. A potentially powerful architecture could be attributed a weak evaluation since it inherits inappropriate weights [64]. Solutions to this phenomenon tend to simultaneously address the optimisation-gap issue by ensuring architecture weights sampled from the super-network more closely resemble their final real world performance. DNA [64] divides the architecture search space into blocks with similar architectures, and weights are shared only within the blocks. Distribution Consistent NAS [65] adopts a comparable strategy, whereby architectures sharing at least one operator are iteratively sampled and their weights updated. The (layer-wise) architecture space is divided into clusters of architectures sharing operators at a given layer, via K-means clustering. The super-network can thus be jointly optimised not only according to its parameters but also to its topological structure (i.e. between which architectures the weights can be shared [65]). BossNAS [66] employs block-wise NAS similar to [64] within a CNN-transformer hybrid network. By constructing a searchable cell that can simulate both convolution and transformer network architectures, and a fabric [117] consisting of several such cells that optionally can halve the spatial resolution, the searched architecture resembles either conventional CNN, transformer, or a mixture of the two.

GDAS [67] samples one architecture of the super-network at each training iteration in an attempt to i) reduce the memory requirement during training and ii) increase efficiency, and by extension the convergence rate of the network. Furthermore, the authors suggest that searching for the best reduction cell can be ignored during the NAS process, since they can be effectively hand-crafted and contribute less to overall network architecture performance. They claim to produce state-of-the-art results in a fraction of the time, but acknowledge that the results are not necessarily fair without re-implementing existing methods and evaluating them on the same experimental setup. NSAS [68] adopts an alternative approach to CAS (Section 3.1) to prevent 'forgetting' prior knowledge (the previous network architecture performance reduces under weights learned by a new architecture), through introducing a novel loss function that penalises such an occurrence. The NSAS solution is interleaved within the existing GDAS framework, denoted GDAS-NSAS.

Landmark Regularization [69] addresses the weight inheritance problem by improving the quality of the weights of the super-network themselves. Prior to super-network training, randomly sampled standalone architectures (landmarks) are trained to convergence. During the training of the super-network, the performance of the sub-network architectures is preserved with an additional regularisation term within the loss function. Super-network performance divergence from known landmark performance can therefore be minimised such that a sampled architecture more closely resembles its real-world performance. However, regularisation is only introduced after a sufficient warm-up period to minimise detrimental performance during early training iterations.

3.2.3. Optimising NAS for small network architectures

While NAS helps improve performance compared to conventional hand-crafted neural network training, one significant benefit it offers is directly optimising for network inference speed and memory cost. TAS [70] adopts a gradient-based search strategy to search for optimal size (network width and depth) instead of topology. Superfluous channel inputs (determining network width) and layers (network depth) are pruned. By designing a novel loss function, complex architectures are penalised, and the best performing network in terms of both accuracy and complexity is identified. Using knowledge transfer with a KD algorithm [118], the searched pruned network architecture inherits weights from the trained super-network.

Other approaches allow for explicit searching of network architecture depth. NetAdaptV2 [71] enable architectures to be efficiently derived from the trained super-network with respect to their latency constraints. A new searchable channel-level bypass connection (CBC) is introduced, whereby all output channels of a given layer can be bypassed to simulate the removal of the entire layer. FBNet [72] employs a differentiable NAS strategy wherein cells at different network architecture depths are searched from different architecture spaces, across expansion rate, kernel size, and group number (for group convolution). Further, they demonstrate that optimising network latency is a superior measurement over optimising FLOPs towards generating small and fast networks. SVD-NAS [119] propose an algorithm to optimise the search for low latency network architectures via the substitution of architecture layers with those optimised for FLOPs (low-rank approximation). The results are presented for gradient-based NAS frameworks, but can be deployed alongside any two-stage NAS approach.

3.3. Performance prediction

An alternative method to reduce NAS training speed is to forgo training to completion entirely and instead *predict* how well a given network architecture will perform from its behaviour after minimal training. These methods can entirely circumvent some of the drawbacks illustrated above, such as weight inheritance within gradient-based NAS, as weights are not necessarily assigned at all. Performance prediction can either be generated from an auxiliary model, or estimated after incomplete training directly.

3.3.1. Performance prediction with auxiliary models

While training an auxiliary model introduces its own computational overhead, in the extreme case only this one model might need to be trained in the entire end-to-end NAS search stage. Converse to previously described NAS strategies, no model from the architecture needs to be trained at all before its performance can be evaluated.

One of the earliest examples to in part adopt this strategy is PNAS [73]. All shallow network architectures in the search space are quickly trained and used as training samples for an auxiliary predictor network. Progressively more complex network architectures are constructed and evaluated by the auxiliary network. Of these, the k-best are selected, trained, and used as training samples for the auxiliary network. The process is repeated until network architectures of sufficient complexity are generated and the predictor network guides the search through the architecture space.

NAO [74] employs an auxiliary predictor network to predict network architecture accuracy from its continuous representation (generated from a second auxiliary network). Gradient ascent can then be applied to determine the best network architecture embedding. Finally, a decoder network is used to extract a generated network architecture from its continuous representation. Despite employing a gradient-based search strategy, we include NAO within this section as an architecture sampled during the search phase is not evaluated in a conventional manner (i.e. network propagation with images), but by an auxiliary predictor network. Morphological [75] NAO improve performance by adding novel morphological operations into the search space.

Wen et al. [120] train their own (graph convolutional-based) predictor regression model, wherein N architectures from the NAS-Bench-101 search space are sampled along with their validation accuracies. Indeed, their network converges faster and more accurately than the best identified predictor adopted by NAS-Bench-101 [33] (Regularized Evolution [121]). The regression model is further trained on the ProxylessNAS [40] search space, yielding competitive models for ImageNet.

Baker et al. [122] formulates architecture performance prediction within a Bayesian framework. They train a predictor network upon both features (architecture parameters and hyperparameters), time-series validation accuracy data (i.e. validation accuracy of a given network at several different epochs, for many networks), and first and second order validation accuracy differences. They train an ensemble of sequential regression models where each successive model uses an additional point from the time-series data. The final predictor network can determine whether a given partially trained network architecture is worth terminating or continuing to train, and therefore is sufficient for fast hyperparameter optimisation algorithms such as Hyperband [123, 124].

GBDT-NAS [76] employs a gradient boosting decision tree (GBDT) in order to predict the performance of the neural network architecture during the search phase. They further corroborate that pruning the search space into a smaller but well-performing space allows the NAS controller to sample the best architectures with higher probability [37]. By using GBDT to assess the contribution of an operation, they are able to prune architectures that employ weaker operations.

ReNAS [77] encodes a given architecture into a feature tensor representing an adjacency matrix of the operations between given nodes. A predictor is trained to map feature tensors to architecture performance, wherein the predictor network prioritises preserving architecture *ranking*. Absolute performance (MSE loss between a given architecture and its performance) is considered less important than ranking, since the relative performance between two architectures is more important during the search phase than their absolute performance.

RMI [93] reformulates the NAS search phase as an operation selection challenge for a given edge in an architecture. In turn, this edge can be represented as a one-hot vector, enabling representation of an architecture as a matrix, for input to a random forest. Architectures are derived by the forest, through an iterative selection-update process using a novel RMI score based on mutual information and approximated by Hilbert–Schmidt Independence Criterion (HSIC) [125]. Once a sufficient number of architectures have been generated, the average (mode) operation for each edge is chosen for the final architecture (Fig. 4).

3.3.2. Estimating performance after incomplete training

An alternative prediction-based NAS approach without the use of an auxiliary network is to quickly evaluate the performance of a network without training it to completion. Given an efficient evaluation process, all architectures in the architecture space can be directly evaluated quickly. MdeNAS [28] posits that network architectures that perform well after minimal iterations perform well after convergence, and demonstrate their hypothesis within a multinomial distribution framework, achieving state-of-the-art results six times faster than concurrent (non-prediction-based) NAS methodology.

NASWOT (NAS Without Training) [29] examines the network architecture performance after being trained on a single minibatch and accurately predicts its performance after full training. The local linear maps of network architectures that perform best will be independent across data point samples. Equivalently, a well-performing model must be able to distinguish between the local linear operators associated with each data point in order to model a complex target function. A poor-performing network architecture's operators will 'activate' similarly for different images, and thus the image inputs are difficult to disentangle (their respective activation matrix will appear denser - Fig. 5). Their pipeline is able to achieve near-state-of-the-art accuracy in mere seconds.

3.3.3. Performance-prediction strategy drawbacks

Of course, performance prediction strategies are not without their limitations. Mok et al. [126] suggest that several prediction-based strategies are inherently flawed. Estimating network performance at initialisation often employs the neural tangent kernel (NTK), on which Frobenius Norm (utilised by RMI [93]) and other common techniques are based. They demonstrate that modern DNN violate assumptions necessary to adopt NTK, because they evolve non-linearly during training.

FreeRea [94] also acknowledge the limitations that NTK methods yield, and build upon the earlier genetic REA [127] algorithm by independently mutating parent cells and then uniformly sampling the resultant cell genes to better explore the network architecture space. FreeRea assigns a more appropriate fitness score to a given network architecture cell by adopting a modified Synflow [128] approach to evaluate the summed contribution of network architecture weights, wherein the weights are scaled down logarithmically ('LogSynflow'). Additionally, architectures with skip connections are in fact *encouraged* to yield practical deep network architecture training.

3.3.4. Bayesian optimisation

In general, Bayesian Optimisation (BO) considers a function $f(x)$ that is complex or unknown (thus behaving as a "black box"). In the context of NAS, we can denote f as the performance of a given architecture x. To optimise f, we require some kernel (k) that considers the distance between two inputs (x, x_1). Furthermore, we require an acquisition function $a(f, k, x)$, a measure of an expected loss of evaluating f at x, given a kernel function k. For clarity, let us consider the architectures x, x_1, where $k(x, x_1)$ is very small (i.e. the architectures are similar). If $f(x)$ is high (i.e. architecture x performs well), we would be wise to compute $f(x_1)$, as architecture x_1 is likely to perform well. If $f(x)$ is low (i.e architecture x performs poorly), we should instead compute $f(x_2)$, for some alternative architecture x_2. There is little benefit in computing $f(x_1)$ as it will be similar to $f(x)$, while computing $f(x_2)$ enables better exploration of the entire search space. Provided that a is more easily computable than f, a BO approach to architecture selection via maximising a should be efficient.

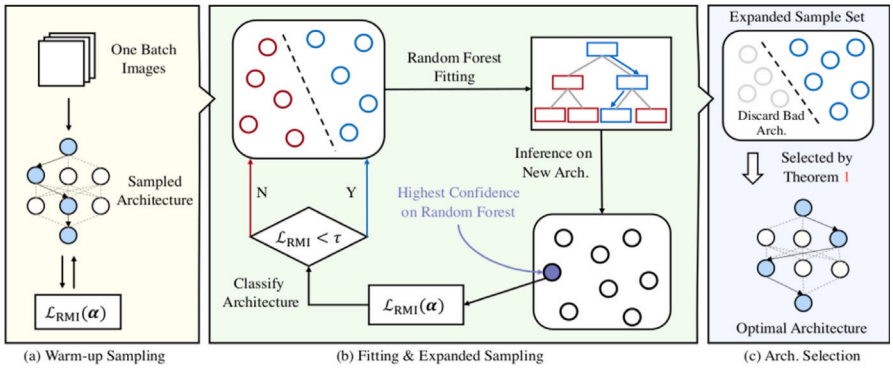

Fig. 4. Three step NAS process from RMI [93] paper. (a) RMI score is used to classify good and bad architectures from the search space, additionally warming-up the random forest. (b) Architectures are selected according to random forest confidence. The architecture performance is estimated via RMI score, both classifying the architecture as good or bad, and for training the forest. (c) The most common operation at each edge from the best architectures is selected to generate the final architecture.

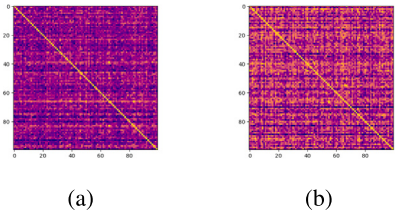

Fig. 5. Visualisation of the ability of a generated network to distinguish between given image inputs. Row i, column j corresponds to the hamming distance between the binary codes representing the activation pattern of the ReLU operations of the given neural network architecture, induced by image i and image j. The matrix is normalised such that the distance between the codes induced by identical images (the diagonal) is 1. High-performing network architectures (a) therefore have fewer off-diagonal elements.

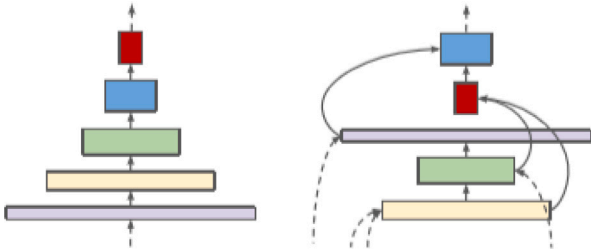

Fig. 6. Scale-decreasing vs Scale-permuted network from SpineNet [83] paper. The width of the block indicates feature resolution and the height indicates feature dimension.

Indeed, NASBOT [78] first adopted BO strategy for NAS, utilising expected improvement as the acquisition function. The authors define k as the OTMANN distance, a measure of the structural similarities between two architectures, weighted by their computational contribution to the network as a whole. This distance is computed efficiently via optimal transport algorithm [129].

Auto-Keras [79] adopts an alternative BO configuration, using an upper-confidence bound acquisition function, with the edit-distance as kernel definition. This can be formulated as an approximate dynamic programming algorithm that can be efficiently minimised under an equivalent bipartite graph matching problem. BayesNAS [80] instead adopt an entropy-based acquisition function with an incorporated hierarchical ARD prior [130].

BANANAS [81] identifies the drawbacks with rudimentary BO strategy, given the resource-intensive distance function computation. Instead they propose using a predictor network to remove the need for a distance function entirely. Consider an architecture encoding as a binary mask of the entire search space, where there is a 1 if that path (the series of operations from input to output) exists in the architecture. Given this path-encoding representation of an architecture, a neural network can predict its performance. By taking an ensemble of m predictors, the mean and standard deviation of the m predictions for an input architecture can be computed. Mean and standard deviation are inherent to Bayesian Optimisation acquisition functions, and improving their reliability improves overall optimisation performance. The authors determined upper-confidence bound acquisition function with a mutation optimisation strategy to be the best BO operation configuration.

4. Object detection

In general, NAS towards more complex tasks than image classification, such as detection and segmentation is less studied. Within these domains, there is an understandable focus on architecture space definition to capture the additional requirements for object detection and segmentation heads. Most strategies presented below can be combined with the advancements highlighted in the image classification domain above.

4.1. Searching for backbone architectures

DetNAS [82] identify the unsuitability of older NAS strategies (notably non-gradient based) when searching for detection backbone architectures due to the level of granularity required, and thus the necessity to pretrain architectures on ImageNet. They propose DetNAS instead, which, much like gradient-based NAS strategies, generates a super-network that requires only one pretraining cycle on ImageNet [100]. In contrast to gradient-based strategies however, only one path is sampled during each iteration, and thus proposed architectures have entirely independent weights. Furthermore, super-network training and search space traversal is decoupled, allowing convergence to be achieved by an evolutionary algorithm rather than gradient-based.

SpineNet [83] employs a reinforcement learning (RL) NAS strategy to determine backbone architectures for object detectors. However, they posit that common leading scale-decreased backbones (e.g. ResNet [131]) may be unsuitable for detection architectures due to the loss of spatial information within down-sampling. This information may not be fully recovered by subsequent decoder networks, including FPN [132]. As such, generated architectures contain a (fixed, scale-decreased) stem followed by a learned scale-permuted network consisting of several blocks. Each block does not necessarily need to connect to a subsequent block corresponding to the next lowest resolution (scale-decreasing). Instead, blocks can connect to blocks of varying resolution, upsampling or downsampling as required (see Fig. 6).

NATS [84] considers a gradient-based NAS approach for object detection backbone architectures. In order to achieve the required level

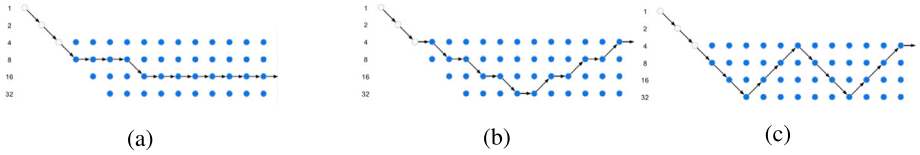

Fig. 7. Examples from Auto-DeepLab [90] paper of the different network architectures that can be captured by their search space. Spatial resolution only ever doubles, halves, or remains unchanged in a given layer. Maximum downsample rate is 32. (a) DeepLabv3 [133]. (b) Encoder–decoder architecture, successfully deployed within semantic segmentation by Conv-Deconv [134]. (c) Stacked hourglass [135] architecture.

of granularity, lest found backbone architectures generate too coarse features [82], NATS further decomposes the search space beyond path-level strategies to the channel-level. Each channel at each operation is assigned its own parameter, allowing the channel search space to be continuous for gradient search.

4.2. Searching for FPN and detection head network architectures

Conversely to backbone search, [85,86] consider the FPN network architecture as the search space within their NAS frameworks. NAS-FPN [85] employs reinforcement learning to iterate over the FPN search space in their framework. They propose a 'general' FPN-block, whereby two feature layers are sampled and pooled to generate a new feature layer, that is itself samplable. The authors further propose a simple but effective strategy to realise the accuracy-speed tradeoff, whereby the FPN architecture can be stacked since its input and output are feature layers of identical scales.

Auto-FPN [86] opts for a gradient-based NAS framework to generate detector architectures. Similar to NAS-FPN [85], FPN network architecture is generalised within the search space, but the generalisation is further extendable to PANet [136] and SSD [137] style pyramidal network architectures. Furthermore, the authors consider a head cell within the search space, to optimise classification and bounding box regression. Their Auto-FPN network architecture yields less accurate results than the concurrent work [85], but with a fraction of the resources required during training.

Aligning motivations with Auto-FPN, NAS-FCOS [87] benefits from searching for both a competitive FPN as well as bounding box regressor head. They are able to achieve state-of-the-art performance by generating network architectures based upon the FCOS [138] anchor-free network architecture space. An FPN architecture is discovered while the regressor head is frozen. A regressor head is then searched for, using the found (frozen) FPN architecture. The top 10 searched head architectures are then selected for full training to determine the best single FCOS-based network architecture.

OPANAS [88] applies the NAS strategy when searching for an optimal FPN architecture within visual object detection. Representing a node as a feature map, and edges between nodes as possible information paths, an FPN super-network can be constructed as a DAG akin to commonplace NAS solutions. Here, the information paths represent different pyramid architectures (top-down, bottom-up, scale-equalising, fusing-splitting, skip-connect and none). The optimal aggregation of information paths can be derived from a trained super-network through an evolutionary algorithm, yielding a final optimal FPN architecture.

5. Image segmentation

The image segmentation domain poses a new style of problem compared to previous vision-based challenges, namely capturing long-range dependencies between features for dense (pixel-wise) prediction [89,91]. Common solutions include scale image pyramids [139–141], encoder–decoder networks [142–144] and atrous convolution resampling [145–147].

DPC [89] constructs a novel search space for dense prediction, encapsulating both spatial pyramid pooling and atrous separable convolutions to capture the aforementioned multi-scale contexts. With a random sampling search strategy, they are among the first to adapt NAS towards image segmentation, outperforming hand-crafted architectures for scene parsing, person part segmentation and semantic image segmentation.

Auto-DeepLab employs a different strategy wherein gradient-based search is adopted to find cells optimised for dense prediction. In addition to searching for optimal convolutional fabric [148] cells, the hierarchical network level search space is also traversed. High level spatial resolutions are thereby preserved as the inter-connectedness of a searched cell is not pre-defined, but explicitly searched for (Fig. 7).

DCNAS [91] builds upon the trellis search space [90], constructing a densely connected search space. By using a fusion module that efficiently aggregates semantic information between layers, the resource-intensity during search is minimised such that the same dataset can be used for the NAS search and evaluation phase, minimising the optimisation-gap (see [40,58]).

EDNAS [51] present a multi-task scene-understanding (image segmentation, depth prediction, and surface-normal estimation) NAS algorithm that focuses on the generation of network architectures that are optimal with respect to latency for given hardware. By designing a search space best suited to the placement of Inverted Bottleneck [149] blocks within an EfficientNet [150] backbone, an evolutionary search algorithm [127] can find architectures optimised for edge platforms.

6. Discussion

Clearly, there is a definitive bias towards NAS for image classification over object detection and image segmentation. This can be attributed to the complexity of image classification architecture, which until recently, could be more easily trained end-to-end. As such, these CNN architectures are well suited for architecture search within the NAS pipeline. While extending CNN architecture search to backbones to object detection and image segmentation is possible, the result is not as impressive. Indeed, suitable backbone architecture for these problems is only half the challenge. However, with the rise of object detection transformer architecture, which achieves very high performance on common datasets, and whose modularity is well suited towards NAS, one can expect this phenomenon to disappear. It is unsurprising therefore, that NAS for transformers is receiving increased popularity in recent literature [46,66]. We further note that implementing the architecture space for NAS in line with the modifications implemented within ConvNeXt architecture [5], which is capable of state-of-the-art results in both image classification and object detection, is a strong candidate for future research within the NAS domain. Many of the modifications proposed in ConvNeXt alter the architecture to remove any inductive bias stemming from local pixel relationships in place of long-distance pixel relationships. The same architecture changes can be readily applied to the NAS architecture space.

The focus on architecture space *traversal* is evident. Conversely, even in the case of image classification, the generation of resource-efficient architecture for which NAS is so well-suited is limited. Where such strategies do exist [42,46,70,71,73], only [46] considers architecture beyond pure CNN or domains other than image classification. Since NAS can ultimately be reduced to a ranking of architectures, and thus introducing resource constraints into the ranking is both sufficient

and efficient, this pattern is unfounded and there is much benefit to be gained here.

Finally, we note that dataset optimisation is hardly considered within NAS frameworks, with the exception of Pi-NAS [57]. Hard example mining and curriculum learning, which are prevalent within conventional network training, receive no attention within NAS (excluding CNAS [32] which utilises a curriculum for preparing the architecture space rather than the dataset). Considering a given dataset is often iterated over more times within NAS than manual training, there is no justification for this.

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.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by Durham University, United Kingdom, the European Regional Development Fund Intensive Industrial Innovation Grant No. 25R17P01847 and Petards Joyce-Loebl Ltd.

References

[1] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, Intell. Neurosci. 2018 (2018).
[2] H. Xiao, Z. Wang, Z. Zhu, J. Zhou, J. Lu, Shapley-NAS: Discovering operation contribution for neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11892–11901.
[3] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep., 2009.
[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[5] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A convnet for the 2020s, in: Conference on Computer Vision and Pattern Recognition, 2022, pp. 11966–11976.
[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[7] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.C. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[8] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, X. Wang, A comprehensive survey of neural architecture search: Challenges and solutions, 2020, arXiv e-prints.
[9] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: A survey, J. Mach. Learn. Res. 20 (2019) 55:1–55:21.
[10] M. Wistuba, A. Rawat, T. Pedapati, A survey on neural architecture search, 2019, arXiv e-prints.
[11] T.M. Hospedales, A. Antoniou, P. Micaelli, A.J. Storkey, Meta-learning in neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2022) 5149–5169.
[12] Y. Liu, Y. Sun, B. Xue, M. Zhang, G.G. Yen, K.C. Tan, A survey on evolutionary neural architecture search, IEEE Trans. Neural Netw. Learn. Syst. 34 (2) (2023) 550–570.
[13] J. Kang, J.K. Kang, J. Kim, K. Jeon, H. Chung, B. Park, Neural architecture search survey: A computer vision perspective, Sensors 23 (3) (2023) 1713.
[14] D. Baymurzina, E.A. Golikov, M.S. Burtsev, A review of neural architecture search, Neurocomputing 474 (2022) 82–93.
[15] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P. Kindermans, Q.V. Le, Can weight sharing outperform random architecture search? An investigation with tunas, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2020, pp. 14311–14320.
[16] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, in: 5th International Conference on Learning Representations, 2017.
[17] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, in: 5th International Conference on Learning Representations, 2017.
[18] H. Cai, T. Chen, W. Zhang, Y. Yu, J. Wang, Efficient architecture search by network transformation, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 2787–2794.
[19] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, MnasNet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.
[20] Y. Gao, H. Yang, P. Zhang, C. Zhou, Y. Hu, Graph neural architecture search, in: Proceedings of the 29th International Joint Conference on Artificial Intelligence, 2020, pp. 1403–1409.
[21] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Netw. 5 (1) (1994) 54–65.
[22] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning, Omni Press, 2010, pp. 807–814.
[23] T. Elsken, J.H. Metzen, F. Hutter, Efficient multi-objective neural architecture search via lamarckian evolution, in: 7th International Conference on Learning Representations, 2019.
[24] Y. Sun, B. Xue, M. Zhang, G.G. Yen, Evolving deep convolutional neural networks for image classification, IEEE Trans. Evol. Comput. 24 (2) (2020) 394–407.
[25] Y. Sun, B. Xue, M. Zhang, G.G. Yen, J. Lv, Automatically designing CNN architectures using the genetic algorithm for image classification, IEEE Trans. Cybern. 50 (9) (2020) 3840–3854.
[26] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: 7th International Conference on Learning Representations, 2019.
[27] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive DARTS: bridging the optimization gap for NAS in the wild, Int. J. Comput. Vis. 129 (3) (2021) 638–655.
[28] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, Q. Tian, Multinomial distribution learning for effective neural architecture search, in: International Conference on Computer Vision, 2019, pp. 1304–1313.
[29] J. Mellor, J. Turner, A.J. Storkey, E.J. Crowley, Neural architecture search without training, in: Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 139, 2021, pp. 7588–7598.
[30] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 80, 2018, pp. 4095–4104.
[31] S. Xie, H. Zheng, C. Liu, L. Lin, SNAS: stochastic neural architecture search, in: 7th International Conference on Learning Representations, 2019.
[32] Y. Guo, Y. Chen, Y. Zheng, P. Zhao, J. Chen, J. Huang, M. Tan, Breaking the curse of space explosion: Towards efficient NAS with curriculum search, in: Proceedings of the 37th International Conference on Machine Learning, 2020.
[33] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, F. Hutter, NAS-bench-101: Towards reproducible neural architecture search, in: Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 97, 2019, pp. 7105–7114.
[34] X. Dong, Y. Yang, NAS-Bench-201: Extending the scope of reproducible neural architecture search, in: International Conference on Learning Representations, 2020.
[35] X. Dong, L. Liu, K. Musial, B. Gabrys, NATS-Bench: Benchmarking NAS algorithms for architecture topology and size, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[36] M. Lindauer, F. Hutter, Best practices for scientific research on neural architecture search, 2019, arXiv e-prints.
[37] I. Radosavovic, R. Kosaraju, R. Girshick, K. He, P. Dollar, Designing network design spaces, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10425–10433.
[38] R. Pasunuru, M. Bansal, Continual and multi-task architecture search, in: Proceedings of the 57th Conference of the Association for Computational Linguistics, 2019, pp. 1911–1922.
[39] J. Yu, P. Jin, H. Liu, G. Bender, P. Kindermans, M. Tan, T.S. Huang, X. Song, R. Pang, Q. Le, BigNAS: Scaling up neural architecture search with big single-stage models, in: Proceedings of the European Conference on Computer Vision, Vol. 12352, 2020, pp. 702–717.
[40] H. Cai, L. Zhu, S. Han, Proxylessnas: direct neural architecture search on target task and hardware, in: 7th International Conference on Learning Representations, 2019.
[41] Q. Yao, J. Xu, W.-W. Tu, Z. Zhu, Efficient neural architecture search via proximal iterations, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020, pp. 6664–6671.
[42] D. Wang, M. Li, C. Gong, V. Chandra, AttentiveNAS: Improving neural architecture search via attentive sampling, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2021, pp. 6418–6427.
[43] L. Tong, B. Du, Neural architecture search via reference point based multi-objective evolutionary algorithm, Pattern Recognit. 132 (2022) 108962.
[44] A. Jordao, F. Yamada, M. Lie, W.R. Schwartz, Stage-wise neural architecture search, in: International Conference on Pattern Recognition, 2020.

[45] X. Xia, X. Xiao, X. Wang, M. Zheng, Progressive automatic design of search space for one-shot neural architecture search, in: Winter Conference on Applications of Computer Vision, IEEE, 2022, pp. 3525–3534.
[46] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, W. Ouyang, Glit: neural architecture search for global and local image transformer, in: International Conference on Computer Vision, 2021, pp. 12–21.
[47] M. Chen, J. Fu, H. Ling, One-shot neural ensemble architecture search by diversity-guided search space shrinking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16530–16539.
[48] H. Shi, R. Pi, H. Xu, Z. Li, J.T. Kwok, T. Zhang, Bridging the gap between sample-based and one-shot neural architecture search with BONAS, in: Advances in Neural Information Processing Systems, 2020.
[49] Y. Jiang, C. Hu, T. Xiao, C. Zhang, J. Zhu, Improved differentiable architecture search for language modeling and named entity recognition, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2019, pp. 3585–3590.
[50] Y. Wu, A. Liu, Z. Huang, S. Zhang, L.V. Gool, Neural architecture search as sparse supernet, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021, pp. 10379–10387.
[51] A.A.M. Javier Garcia López, F.M. Noguer, E-DNAS: Differentiable neural architecture search for embedded systems, in: International Conference on Pattern Recognition, 2020.
[52] Y. Yang, H. Li, S. You, F. Wang, C. Qian, Z. Lin, Ista-nas: Efficient and consistent neural architecture search by sparse coding, Adv. Neural Inf. Process. Syst. 33 (2020).
[53] D. Bruggemann, M. Kanakis, S. Georgoulis, L. Van Gool, Automated search for resource-efficient branched multi-task networks, in: BMVC, 2020.
[54] A. Brock, T. Lim, J.M. Ritchie, N. Weston, SMASH: one-shot model architecture search through HyperNetworks, in: 6th International Conference on Learning Representations, 2018.
[55] S. Yan, Y. Zheng, W. Ao, X. Zeng, M. Zhang, Does unsupervised architecture representation learning help neural architecture search? Adv. Neural Inf. Process. Syst. 33 (2020).
[56] X. Chu, B. Zhang, R. Xu, FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search, in: International Conference on Computer Vision, 2021, pp. 12219–12228.
[57] 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: International Conference on Computer Vision, 2021, pp. 12334–12344.
[58] Y. Yang, S. You, H. Li, F. Wang, C. Qian, Z. Lin, Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 6667–6676.
[59] Y. Gu, L. Wang, Y. Liu, Y. Yang, Y. Wu, S. Lu, M. Cheng, DOTS: decoupling operation and topology in differentiable architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12311–12320.
[60] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, H. Xiong, PC-DARTS: partial channel connections for memory-efficient architecture search, in: 8th International Conference on Learning Representations, 2020.
[61] W. Wang, X. Zhang, H. Cui, H. Yin, Y. Zhang, FP-DARTS: Fast parallel differentiable neural architecture search for image classification, Pattern Recognit. 136 (2023) 109193.
[62] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, F. Hutter, Understanding and robustifying differentiable architecture search, in: 8th International Conference on Learning Representations, 2020.
[63] X. Dong, Y. Yang, One-shot neural architecture search via self-evaluated template network, in: International Conference on Computer Vision, 2019, pp. 3680–3689.
[64] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, X. Chang, Block-wisely supervised neural architecture search with knowledge distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1986–1995.
[65] J. Pan, C. Sun, Y. Zhou, Y. Zhang, C. Li, Distribution consistent neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10884–10893.
[66] C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, X. Chang, BossNAS: Exploring hybrid CNN-transformers with block-wisely self-supervised neural architecture search, in: International Conference on Computer Vision, IEEE, 2021, pp. 12261–12271.
[67] X. Dong, Y. Yang, Searching for a robust neural architecture in four GPU hours, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1761–1770.
[68] M. Zhang, H. Li, S. Pan, X. Chang, S.W. Su, Overcoming multi-model forgetting in one-shot NAS with diversity maximization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7806–7815.
[69] K. Yu, R. Ranftl, M. Salzmann, Landmark regularization: Ranking guided super-net training in neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13723–13732.
[70] X. Dong, Y. Yang, Network pruning via transformable architecture search, in: Advances in Neural Information Processing Systems, 2019, pp. 759–770.
[71] T. Yang, Y. Liao, V. Sze, NetAdaptV2: Efficient neural architecture search with fast super-network training and architecture optimization, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2021, pp. 2402–2411.
[72] 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10734–10742.
[73] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search, in: Automated Machine Learning - Methods, Systems, Challenges, in: The Springer Series on Challenges in Machine Learning, Springer, 2019, pp. 63–77.
[74] R. Luo, F. Tian, T. Qin, E. Chen, T.-Y. Liu, Neural architecture optimization, in: Advances in Neural Information Processing Systems, Vol. 31, 2018, pp. 7816–7827.
[75] Y. Hu, N. Belkhir, J. Angulo, A. Yao, G. Franchi, Learning deep morphological networks with neural architecture search, Pattern Recognit. 131 (2022) 108893.
[76] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, T.-Y. Liu, Neural architecture search with GBDT, 2020, arXiv e-prints.
[77] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, C. Xu, ReNAS: Relativistic evaluation of neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4411–4420.
[78] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, E.P. Xing, Neural architecture search with Bayesian optimisation and optimal transport, in: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, 2018, pp. 2020–2029.
[79] H. Jin, Q. Song, X. Hu, Auto-keras: An efficient neural architecture search system, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2019, pp. 1946–1956.
[80] H. Zhou, M. Yang, J. Wang, W. Pan, BayesNAS: A Bayesian approach for neural architecture search, in: Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 97, 2019, pp. 7603–7613.
[81] C. White, W. Neiswanger, Y. Savani, Bananas: bayesian optimization with neural architectures for neural architecture search, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021, pp. 10293–10301.
[82] Y. Chen, T. Yang, X. Zhang, G. MENG, X. Xiao, J. Sun, DetNAS: Backbone search for object detection, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[83] X. Du, T.-Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q.V. Le, X. Song, SpineNet: Learning scale-permuted backbone for recognition and localization, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2020, pp. 11589–11598.
[84] J. Peng, M. Sun, Z.-X. Zhang, T. Tan, J. Yan, Efficient neural architecture transformation search in channel-level for object detection, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[85] G. Ghiasi, T.-Y. Lin, Q.V. Le, NAS-FPN: Learning scalable feature pyramid architecture for object detection, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2019, pp. 7029–7038.
[86] H. Xu, L. Yao, Z. Li, X. Liang, W. Zhang, Auto-FPN: Automatic network architecture adaptation for object detection beyond classification, in: IEEE/CVF International Conference on Computer Vision, 2019, pp. 6648–6657.
[87] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, Y. Zhang, NAS-FCOS: Fast neural architecture search for object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.
[88] T. Liang, Y. Wang, Z. Tang, G. Hu, H. Ling, OPANAS: one-shot path aggregation network architecture search for object detection, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2021, pp. 10195–10203.
[89] L. Chen, M.D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, J. Shlens, Searching for efficient multi-scale architectures for dense image prediction, in: Advances in Neural Information Processing Systems, 2018, pp. 8713–8724.
[90] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A.L. Yuille, L. Fei-Fei, Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation.
[91] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, W. Ren, DCNAS: densely connected neural architecture search for semantic image segmentation, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2021, pp. 13956–13967.
[92] Y. Hu, X. Wang, L. Li, Q. Gu, Improving one-shot NAS with shrinking-and-expanding supernet, Pattern Recognit. 118 (2021) 108025.
[93] X. Zheng, X. Fei, L. Zhang, C. Wu, F. Chao, J. Liu, W. Zeng, Y. Tian, R. Ji, Neural architecture search with representation mutual information, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11912–11921.
[94] N. Cavagnero, L. Robbiano, B. Caputo, G. Averta, FreeREA: Training-free evolution-based architecture search, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2023, pp. 1493–1502.

[95] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, Q. Le, Understanding and simplifying one-shot architecture search, in: Proceedings of the 35th International Conference on Machine Learning, Vol. 80, 2018, pp. 550–559.
[96] N. Parikh, S.P. Boyd, Proximal algorithms, Found. Trends Optim. 1 (3) (2014) 127–239.
[97] X. Wan, B. Ru, P.M. Esperança, F.M. Carlucci, Approximate neural architecture search via operation distribution learning, in: Winter Conference on Applications of Computer Vision, 2022, pp. 3545–3554.
[98] K. Deb, J. Sundar, Reference point based multi-objective optimization using evolutionary algorithms, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, 2006, pp. 635–642.
[99] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[100] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, J. Sun, Single path one-shot neural architecture search with uniform sampling, in: Proceedings of the European Conference on Computer Vision, Vol. 12361, 2020, pp. 544–560.
[101] K. Ahmed, L. Torresani, MaskConnect: Connectivity learning by gradient descent, in: Proceedings of the European Conference on Computer Vision, Vol. 11209, 2018, pp. 362–378.
[102] S. You, T. Huang, M. Yang, F. Wang, C. Qian, C. Zhang, Greedynas: towards fast one-shot NAS with greedy supernet, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2020, pp. 1996–2005.
[103] M. Cho, M. Soltani, C. Hegde, One-shot neural architecture search via compressive sensing, 2019, arXiv e-prints.
[104] N. Simon, J. Friedman, T. Hastie, R. Tibshirani, A sparse-group lasso, J. Comput. Graph. Statist. 22 (2) (2013) 231–245.
[105] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 267–288.
[106] I. Daubechies, M. Defrise, C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Comm. Pure Appl. Math. 57 (11) (2004) 1413–1457.
[107] Q. Guo, X.-J. Wu, J. Kittler, Z. Feng, Differentiable neural architecture learning for efficient neural networks, Pattern Recognit. 126 (2022) 108448.
[108] D. Ha, A.M. Dai, Q.V. Le, HyperNetworks, in: 5th International Conference on Learning Representations, 2017.
[109] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: 2nd International Conference on Learning Representations, 2014.
[110] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, arXiv e-prints.
[111] C. Simon, P. Koniusz, L. Petersson, Y. Han, M. Harandi, Towards a robust differentiable architecture search under label noise, in: Winter Conference on Applications of Computer Vision, 2022, pp. 3584–3594.
[112] K. Yu, C. Sciuto, M. Jaggi, C. Musat, M. Salzmann, Evaluating the search phase of neural architecture search, in: 8th International Conference on Learning Representations, 2020.
[113] L.S. Shapley, A.E. Roth, The Shapley value : essays in honor of Lloyd S. Shapley / edited by Alvin E. Roth, 1988, vii, 330.
[114] L. Shapley, 17. A value for n-person games, in: Contributions to the Theory of Games (AM-28), Volume II, Princeton University Press, 2016, pp. 307–318.
[115] R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in Neural Information Processing Systems, Vol. 28, 2015, pp. 2377–2385.
[116] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, Z. Li, DARTS+: improved differentiable architecture search with early stopping, 2019, arXiv e-prints.
[117] S. Saxena, J. Verbeek, Convolutional neural fabrics, in: Advances in Neural Information Processing Systems, 2016, pp. 4053–4061.
[118] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: The Conference on Neural Information Processing Systems Workshop, 2014.
[119] Z. Yu, C.-S. Bouganis, SVD-NAS: Coupling low-rank approximation and neural architecture search, in: Proceedings of the Winter Conference on Applications of Computer Vision, 2023, pp. 1503–1512.
[120] W. Wen, H. Liu, Y. Chen, H.H. Li, G. Bender, P. Kindermans, Neural predictor for neural architecture search, in: Proceedings of the European Conference on Computer Vision, Vol. 12374, 2020, pp. 660–676.
[121] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: The 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 4780–4789.
[122] B. Baker, O. Gupta, R. Raskar, N. Naik, Accelerating neural architecture search using performance prediction, in: 6th International Conference on Learning Representations, Workshop Track Proceedings, 2018.
[123] L. Li, K.G. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (2017) 185:1–185:52.
[124] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, F. Hutter, SMAC3: a versatile Bayesian optimization package for hyperparameter optimization, J. Mach. Learn. Res. 23 (2022) 54:1–54:9.
[125] A. Gretton, O. Bousquet, A.J. Smola, B. Schölkopf, Measuring statistical dependence with Hilbert-Schmidt norms, in: Algorithmic Learning Theory, 16th International Conference, Vol. 3734, 2005, pp. 63–77.
[126] J. Mok, B. Na, J.-H. Kim, D. Han, S. Yoon, Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training? in: Conference on Computer Vision and Pattern Recognition, 2022, pp. 11861–11870.
[127] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: The 33rd Conference on Artificial Intelligence, 2019, pp. 4780–4789.
[128] H. Tanaka, D. Kunin, D.L.K. Yamins, S. Ganguli, Pruning neural networks without any data by iteratively conserving synaptic flow, in: Advances in Neural Information Processing Systems 33, 2020.
[129] C. Villani, Optimal Transport: Old and New, in: Grundlehren der mathematischen Wissenschaften, 2008.
[130] R.M. Neal, Bayesian Learning for Neural Networks, Springer-Verlag, Berlin, Heidelberg, 1996.
[131] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[132] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2017, pp. 936–944.
[133] L. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, 2017, arXiv e-prints.
[134] H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in: International Conference on Computer Vision, 2015, pp. 1520–1528.
[135] A. Newell, K. Yang, J. Deng, Stacked hourglass networks for human pose estimation, in: Proceedings of the European Conference on Computer Vision, Vol. 9912, 2016, pp. 483–499.
[136] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2018, pp. 8759–8768.
[137] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C.-Y. Fu, A. Berg, SSD: Single shot MultiBox detector, in: Proceedings of the European Conference on Computer Vision, 2016.
[138] Z. Tian, C. Shen, H. Chen, T. He, FCOS: Fully convolutional one-stage object detection, in: Proc. Int. Conf. Computer Vision, 2019.
[139] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.
[140] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in: International Conference on Computer Vision, IEEE Computer Society, 2015, pp. 2650–2658.
[141] P.H.O. Pinheiro, R. Collobert, Recurrent convolutional neural networks for scene labeling, in: International Conference on Machine Learning, in: JMLR Workshop and Conference Proceedings, Vol. 32, JMLR.org, 2014, pp. 82–90.
[142] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.
[143] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495.
[144] G. Lin, A. Milan, C. Shen, I.D. Reid, RefineNet: Multi-path refinement networks for high-resolution semantic segmentation, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2017, pp. 5168–5177.
[145] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848.
[146] M. Holschneider, R. Kronland-Martinet, J. Morlet, P. Tchamitchian, A real-time algorithm for signal analysis with the help of the wavelet transform, in: Wavelets, 1990, pp. 286–297.
[147] A. Giusti, D.C. Ciresan, J. Masci, L.M. Gambardella, J. Schmidhuber, Fast image scanning with deep max-pooling convolutional neural networks, in: IEEE International Conference on Image Processing, 2013, pp. 4034–4038.
[148] S. Saxena, J. Verbeek, Convolutional neural fabrics, in: Advances in Neural Information Processing Systems, 2016, pp. 4053–4061.
[149] M. Sandler, A.G. Howard, M. Zhu, A. Zhmoginov, L. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[150] M. Tan, Q.V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning, Vol. 97, 2019, pp. 6105–6114.

Matt is a Computer Vision / Deep Learning Ph.D. student at Durham University. He has a computer science background, and his areas of interest include Neural Architecture Search, Image Classification, Object Detection, Tracking, and Image Segmentation. In his free time, he enjoys DnD, MTG, and walking.