



Neural Architecture Search for Dense Prediction Tasks in Computer Vision

Rohit Mohan¹ · Thomas Elsken² · Arber Zela¹ · Jan Hendrik Metzen² · Benedikt Staffler² · Thomas Brox¹ · Abhinav Valada¹ · Frank Hutter^{1,2}

Received: 14 February 2022 / Accepted: 16 March 2023 / Published online: 15 April 2023
© The Author(s) 2023

Abstract

The success of deep learning in recent years has led to a rising demand for neural network architecture engineering. As a consequence, neural architecture search (NAS), which aims at automatically designing neural network architectures in a data-driven manner rather than manually, has evolved as a popular field of research. With the advent of weight sharing strategies across architectures, NAS has become applicable to a much wider range of problems. In particular, there are now many publications for dense prediction tasks in computer vision that require pixel-level predictions, such as semantic segmentation or object detection. These tasks come with novel challenges, such as higher memory footprints due to high-resolution data, learning multi-scale representations, longer training times, and more complex and larger neural architectures. In this manuscript, we provide an overview of NAS for dense prediction tasks by elaborating on these novel challenges and surveying ways to address them to ease future research and application of existing methods to novel problems.

Keywords Deep learning · Neural architecture search · AutoML · Object detection · Semantic segmentation

Communicated by Jingdong Wang.

Rohit Mohan and Thomas Elsken have contributed equally to this work.

✉ Rohit Mohan
mohan@cs.uni-freiburg.de

Thomas Elsken
thomas.elsken@de.bosch.com

Arber Zela
zelaa@cs.uni-freiburg.de

Jan Hendrik Metzen
janhendrik.metzen@de.bosch.com

Benedikt Staffler
benediktsebastian.staffler@de.bosch.com

Thomas Brox
brox@cs.uni-freiburg.de

Abhinav Valada
valada@cs.uni-freiburg.de

Frank Hutter
fh@cs.uni-freiburg.de

¹ University of Freiburg, Freiburg, Germany

² Bosch Center for Artificial Intelligence, Robert Bosch GmbH, Stuttgart, Germany

1 Introduction

With the advent of deep learning, features are no longer manually designed but rather learned in an end-to-end fashion from data, resulting in impressive results for various problems, such as image recognition (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), machine translation (Bahdanau et al., 2015), or reasoning in games (Silver et al., 2016). This, however, leads to a new design problem: the feature engineering process is replaced by engineering neural network architectures (Cheng et al., 2020a; Girshick, 2015; Girshick et al., 2014; Goodfellow et al., 2014; He et al., 2016; Howard et al., 2017; Liu et al., 2016; Long et al., 2015; Mohan & Valada, 2020; Redmon et al., 2016; Ren et al., 2015; Ronneberger et al., 2015; Simonyan & Zisserman, 2015; Szegedy et al., 2016, 2017; Tan & Le, 2019; Zhang et al., 2018; Zhong et al., 2020b). This architectural engineering is especially prevalent for dense prediction tasks in computer vision, such as semantic segmentation, object detection, optical flow estimation, or disparity estimation. These tasks typically require complex neural architectures, often composed of various components, each having a different purpose, e.g., extracting features at different scales, feature fusion across levels, or dedicated architectural heads

for, e.g., generating bounding boxes or making class predictions.

Unfortunately, manually designing neural network architectures comes with some major drawbacks, reminding of the drawbacks of manually designing features. Firstly, it is a time-consuming and error-prone process, requiring human expertise. This dramatically limits access to deep learning technologies since architecture engineering expertise is rare. Secondly, performance will be limited by the human imagination. Inspired by learning features from data rather than manually designing them, it seems natural to also replace the manual architecture design by learning architectures from data. This process of automating architectural engineering is commonly referred to as *neural architecture search* (NAS).

Until recently, NAS research has mostly focused on image classification problems, such as CIFAR-10 or ImageNet, due to the demand for computational resources in the order of hundreds or thousands of GPU days that early methods required (Real et al., 2019; Zoph et al., 2017, 2018). Compared to image classification, dense prediction tasks have barely been addressed even though they are of high practical relevance for applications, such as autonomous driving (Huang & Chen, 2020) or medical imaging (Litjens et al., 2017). These problems are intrinsically harder than image classification for several reasons: they typically come with longer training times as well as higher memory footprints due to high-resolution data, and they also require more complex neural architectures. These differences lead to even higher computational demands and make the application of many NAS approaches problematic. Early works on NAS for dense prediction tasks (e.g., Chen et al., 2018b; Ghiasi et al., 2019) are thus limited to optimizing only small parts of the overall architectures, while still requiring enormous computational resources even though employing various tricks for speeding up the search process.

Fortunately, recent weight-sharing approaches via one-shot-models (Bender et al., 2018; Cai et al., 2019; Liu et al., 2019b; Pham et al., 2018; Saxena & Verbeek, 2016; Xie et al., 2019) have dramatically reduced the computational costs to essentially the same order of magnitude as training a single network, making NAS applicable to a much wider range of problems. This led to an increasing interest in developing NAS approaches tailored toward dense prediction tasks. However, due to the complex nature of the problem, these approaches vary vastly, as illustrated in Fig. 1. With this survey, we aim to provide guidance on the most important design decisions.

This manuscript is structured as follows: in Sect. 2, we briefly review NAS and discuss its application for dense prediction tasks in general. In the remaining sections, we focus on the specific problems of semantic segmentation (Sect. 3) and object detection (Sect. 4) and conclude by discussing other less-studied but promising applications (Sect. 5).

2 Towards NAS for Dense Prediction Tasks

In this section, we first describe the general framework of neural architecture search (NAS) in Sect. 2.1. We then discuss the challenges of NAS for dense prediction tasks in Sect. 2.2. Then, a few basic elements of NAS, namely, the search space, the search strategies, performance estimation strategies, and hardware-awareness is briefly discussed in the context of classification and dense prediction tasks in Sects. 2.3, 2.4, 2.5, and 2.6, respectively.

2.1 General NAS Framework

Neural architecture search (NAS) is typically framed as a bi-level optimization problem

$$\begin{aligned} \min_{A \in \mathcal{A}} \quad & \mathcal{L}_{val}(D_{val}, A, w_A^*) \\ \text{s.t.} \quad & w_A^* \in \arg \min_w \mathcal{L}_{train}(D_{train}, A, w), \end{aligned}$$

with the goal of finding an optimal neural network architecture A within a search space \mathcal{A} with respect to a validation loss function \mathcal{L}_{val} , a validation data set D_{val} and weights w_A^* of the architecture obtained by minimizing a training loss function \mathcal{L}_{train} on a training data set D_{train} . NAS methods can be categorized based on three factors (Elsken et al., 2019b), namely, search space, search strategy, and performance estimation. Figure 2 presents the different dimensions of NAS algorithm. Please refer to the surveys by Elsken et al. (2019b), Wistuba et al. (2019) or White et al. (2023) for a more thorough general NAS overview.

2.2 Challenges of Dense Prediction Tasks for NAS

Many NAS methods (Chu et al., 2021; Xu et al., 2019b; Zoph et al., 2018) have shown that improvements in architectures optimized for classification tend to also improve the performance of dense prediction tasks such as semantic segmentation and object detection. However, these architectures are sub-optimal as they mainly bring performance boosts due to capturing generic image features. A task-specific architecture will be an optimal solution as shown by, e.g., Tan et al. (2020). Here, EfficientNet-B3 Tan and Le (2019), a NAS-based classification network, with an FPN achieves about 4% lower mean average precision for the task of object detection than the task-specific NAS-based BiFPN combination. This is mainly due to the fact that the upstream task of classification needs only global contextual features whereas, the downstream task of object detection, requires highly precise localization features as well resulting in a vast difference between them. In other terms, the objective that is optimized during searching for architecture is different from the eventual objective of interest, i.e., there is an optimization gap.

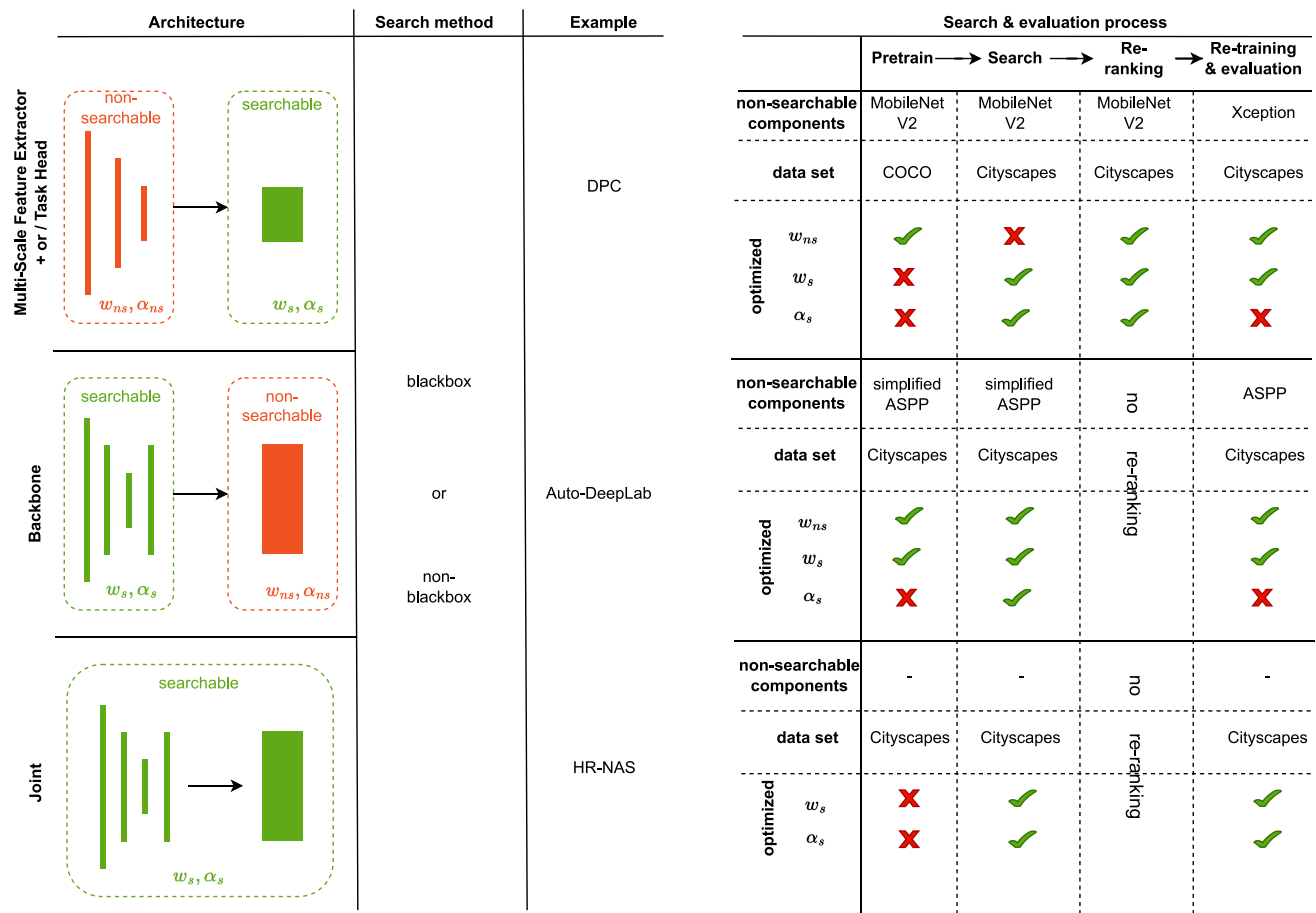


Fig. 1 Visualization of the widely differing architecture search processes. Left: illustration of the overall architecture and which components of the architecture are searchable. DPC Chen et al. (2018b) fix the encoder and search for a dense prediction cell to encode multi-scale information, while Auto-DeepLab Liu et al. (2019a) search for the encoder and augment it with a fixed module for multi-scale feature aggregation. Lastly, HR-NAS search for all involved components of the network. DPC employs a simple blackbox optimization strategy, namely a combination of random search and local search, while

Auto-DeepLab and HR-NAS leverages a one-shot model and gradient-based NAS (Liu et al., 2019b). Right: summary of (i) the different training phases (pretraining, architecture search, re-ranking, re-training, and final evaluation), (ii) non-searchable components in each stage, and (iii) parameters that are optimized in each stage (weights associated with the non-searchable architectural component w_{ns} , weights associated with the searchable architectural component w_s , searchable architectural components α_s)

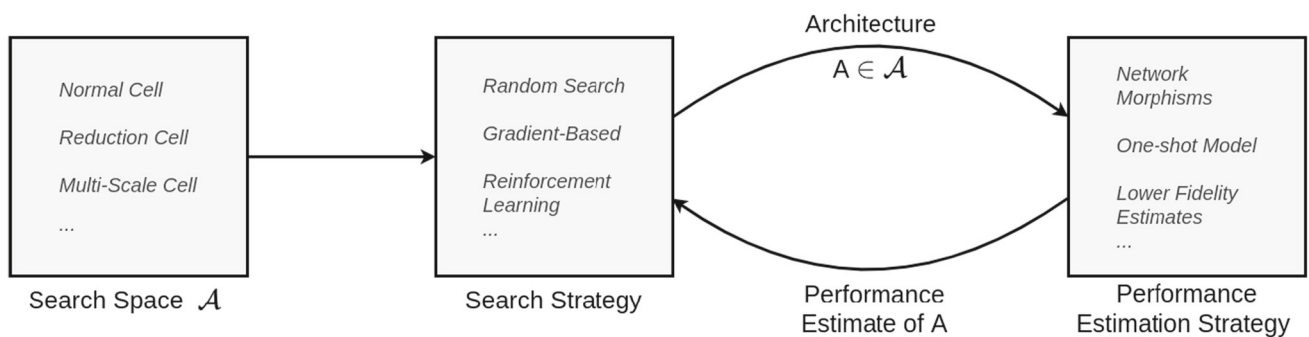


Fig. 2 Different dimensions of NAS algorithms. A search strategy selects an architecture from a predefined search space. The architecture is passed to a performance estimation strategy, which returns the estimated performance to the search strategy

Image classification tasks generally involve searching for encoder-like architectures over smaller spatial resolutions. The dataset available for these tasks can have training samples in the millions. CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015), two of the popular classification task datasets have a resolution of 32×32 and 224×224 , respectively. Additionally, ImageNet consists of over 14 million images. In contrast, dense prediction tasks that are far more complicated than classification tasks necessitate the incorporation of more complex architectures, such as multi-scale feature aggregation (Lin et al., 2017) and long-range context capturing (Chen et al., 2018a) modules while operating on datasets of smaller samples with higher spatial resolution. For example, Cityscapes (Cordts et al., 2016), one of the prominent datasets for semantic segmentation consists of about 5K images of resolution 1024×2048 . The increase in complexity of tasks significantly increases the need for higher resolution images and accordingly intensive annotation efforts. For instance, highly accurate localization of bounding boxes in object detection or of keypoints in pose estimation as well as segmentation of fine structures in semantic segmentation, all require relatively high-resolution images compared to classification tasks. This results in the existing wide disparity between classification and dense prediction task needs.

Hence, when applying NAS for dense prediction problems, researchers do not only have to address novel task-specific challenges, but rather they also have to deal with the even larger optimization gap, which is already existing for image classification problems (Xie et al., 2021), due to an increase in the task complexity.

These challenges can easily lead to high computation costs and memory footprints or even computational infeasibility due to resource constraints. Additionally, dense prediction tasks involve a complicated training pipelines due to the norm of pre-training on datasets other than the target for increasing generalization to achieve better performance. To summarize, NAS approaches for dense prediction tasks should be able to capture architectural variations that are needed for higher resolution images while being efficient in search.

2.3 Search Space

The *search space* defines which architectures can be discovered in principle. For the classification tasks, we search for the encoder architecture while for dense prediction tasks, in addition to the encoder we also search for multi-scale, long-range context encoding modules and task-specific heads. Searchable components of architecture can be architectural hyperparameters, such as the number of layers, the number of filters, or kernel sizes for convolutional layers, but also the layer types themselves, e.g., whether to use a convolutional or a pooling layer. Furthermore, NAS methods can optimize

in which form layers are connected to each other, i.e., they search for the topology of the graph associated with a neural network.

Building prior knowledge about neural network architectures into a search space can simplify the search. For instance, for the encoder of classification networks, inspired by popular manually designed architectures, such as ResNet (He et al., 2016) or Inception-v4 (Szegedy et al., 2017), Zhong et al. (2018) and Zoph et al. (2018) proposed to search for repeatable building blocks (referred to as cells) rather than the whole architecture. These building blocks are then simply stacked in a pre-defined manner to build the full model. Restricting the search space to repeating building blocks limits methods to only optimize these building blocks rather than also discovering novel connectivity patterns and ways of constructing architectures on a macro level from a set of building blocks. Yang et al. (2020) show that the most commonly used search space is indeed very narrow in the sense that almost all architectures perform well. As a consequence, simple search methods, such as random search can be competitive (Elsken et al., 2017; Li & Talwalkar, 2019; Yu et al., 2020b) on classification tasks. We note that this does not necessarily hold for richer, more diverse search spaces (Bender et al., 2020; Real et al., 2020). In contrast, one could also build as little prior knowledge as possible into the search space, e.g., by searching over elementary mathematical operations (Real et al., 2020), however, this would significantly increase the search cost. In general, there is typically a trade-off between search efficiency and the diversity of the search space. Hence, for dense prediction tasks, it is common to employ already pre-optimized blocks from state-of-the-art image classification networks (Bender et al., 2020; Chen et al., 2019; Guo et al., 2020a; Shaw et al., 2019; Wu et al., 2019a), such as MobileNetV2 (Sandler et al., 2018), MobileNetV3 (Howard et al., 2019) or ShuffleNetV2 (Ma et al., 2018) and solely search over their architectural hyperparameters (e.g., the kernel size or the number of filters).

On the other hand, for other parts of the architecture, search spaces are typically built around well-performing manually designed architectures. For example, Chen et al. (2018b) search for a dense prediction cell inspired by operations from DeepLab (Chen et al., 2017, 2018a) and PSPNet (Zhao et al., 2017), Xu et al. (2019a) build their space to contain FPN (Lin et al., 2017) and PANet (Liu et al., 2018b), and the space of Liu et al. (2019a) contains the architectures proposed by Noh et al. (2015), Newell et al. (2016) and Chen et al. (2017).

2.4 Search Strategies

Common *search strategies* used to find an optimal architecture within a search space for classification tasks are black-box optimizers, such as evolutionary algorithms (Elsken et

al., 2019a; Liu et al., 2018a; Real et al., 2017, 2019; Stanley & Miikkulainen, 2002), reinforcement learning (Baker et al., 2017a; Zhong et al., 2018; Zoph et al., 2017, 2018) or Bayesian optimization (Kandasamy et al., 2018; Mendoza et al., 2016; Oh et al., 2019; Ru et al., 2021b; Swersky et al., 2013; White et al., 2019). For dense prediction tasks, e.g., Ghiasi et al. (2019), Chen et al. (2020a), Du et al. (2020), Wang et al. (2020b), Bender et al. (2020) use reinforcement learning while Chen et al. (2019) employ an evolutionary algorithm. However, these methods typically require training hundreds or thousands of architectures and thus result in high computational costs. Several newer methods tailored towards NAS go beyond this blackbox view to overcome these immense computational costs.

For instance, a currently (as of 2022) prominent class of methods are *one-shot* approaches (Bender et al., 2018; Pham et al., 2018; Saxena & Verbeek, 2016), which we discuss in more detail in the next Sect. 2.5. One-shot models enable gradient-based optimization of the architecture via a continuous relaxation of the architecture search space (Liu et al., 2019b). In their work, the authors, compute a convex combination from a set of operations $\mathbb{O} = \{o_0, \dots, o_m\}$ instead of fixing a single operation o_i (eg. convolution or pooling). Hence, with the given input and output layer, x and y , respectively. The output y is computed as follows

$$y = \sum_{i=0}^m \alpha_i o_i(x), \quad \alpha_i \geq 0, \quad \sum_{i=0}^m \alpha_i = 1 \quad (1)$$

where the convex coefficients α_i effectively parameterize the network architecture. Following, the network weights and the network architecture are optimized by alternating gradient descent steps on training data for weights and on validation data for architectural parameters such as α . Eventually, a discrete architecture is obtained by choosing the operation i^* with $i^* = \arg \max_i \alpha_i$ for every layer. In this line of research, rather than making a discrete decision for choosing one out of many candidate operations (such as convolution or pooling), a weighted sum of candidates is used, whereas the weights can then be interpreted as a parameterization of the architecture. Many methods for dense prediction tasks (Guo et al., 2020a; Liu et al., 2019a; Saikia et al., 2019; Xu et al., 2019a; Zhang et al., 2019) also use these gradient-based techniques, often in its first-order approximation for computational reasons.

2.5 Performance Estimation Strategies

The objective function to be optimized by NAS methods is typically the performance an architecture would obtain after

running a predefined (or also optimized) training procedure. However, this true performance is typically too expensive to evaluate. Therefore, various methods for *estimating the performance* have been developed. A common strategy to speed up training is to employ lower-fidelity estimates [e.g., training for fewer epochs, training on subsets of data or down-sampled images, and using down-scaled architectures in the search phase (Baker et al., 2017b; Chrabaszcz et al., 2017; Zela et al., 2018; Zhou et al., 2020; Zoph et al., 2018)]. For dense prediction tasks, computational costs can be further saved by employing common approaches that include the use of pre-trained models (Chen et al., 2018b, 2020a; Guo et al., 2020a; Nekrasov et al., 2019; Wang et al., 2020b), caching of features generated by a backbone (Chen et al. 2018b; Nekrasov et al. 2019; Wang et al. 2020b), or using not just a smaller but potentially also different backbone architecture (Chen et al., 2018b; Ghiasi et al., 2019) in the search process. Lower-fidelity estimates are often used in multiple search phases. In the first stage, architectures are screened in a setting where they are cheap to evaluate (e.g., by using the aforementioned tricks). Once a pool of well-performing architectures or candidate operations is identified, this pool is re-evaluated in a setting closer to the target setting (e.g., by scaling the model up to the target size or by training for more iterations). For example, Chen et al. (2018b) explore 28,000 architectures in the first stage with a downscaled and pre-trained backbone, which is frozen during the search. The authors then choose the top 50 architectures found and train all of them fully to convergence. Rather than selecting top-performing architectures, Guo et al. (2020a) propose a sequential screening of the search space to identify and remove poorly performing operations from the search space. All components of the architecture can then be jointly optimized on the reduced search space, which would have been infeasible on the full space due to memory limitations.

Another popular approach is to employ weight sharing between architectures within one-shot models (Bender et al., 2018; Liu et al., 2019b; Pham et al., 2018; Saxena & Verbeek, 2016) as this overcomes the need for training thousands of architectures. Rather than considering different architectures independently of each other, a single one-shot model is built to subsume all possible elements of the search space. As such instead of applying a single operation (such as 3×3 convolution) to a node, the one-shot model comprises several candidate operations for each node (namely, 3×3 convolution, 5×5 convolution, max pooling, etc).

Individual architectures are then simply subgraphs of the one-shot model and the weights of the one-shot model are shared across subgraphs. This means that, given the one-shot model, one can jointly train all architectures from the search space by only a single training run, namely by

training the one-shot model, rather than training each architecture independently. The performance of an architecture can then be obtained by inheriting the corresponding weights from the one-shot model and evaluating the architecture. While one-shot methods are very popular due to their efficiency, they also come with two major drawbacks: (i) these approaches implicitly assume that the performance of architectures when evaluated with weights inherited from the one-shot model strongly correlates with the performance with weights obtained by independently training the architecture. Whether this assumption holds, or in which settings it holds, is an ongoing discussion (Bender et al., 2018; Pourchot et al., 2020; Xie et al., 2021; Yu et al., 2020a, b; Zela et al., 2020b; Zhang et al., 2020). (ii) One-shot approaches have been shown to not work robustly across datasets and benchmarks for various reasons (Chen & Hsieh, 2020; Elsken et al., 2021; Wang et al., 2021; Xie et al., 2021; Zela et al., 2020a).

Other works for the classification task, focus on *predicting* the performance of neural network architectures, e.g., via trainable surrogate models (Dudziak et al., 2020; Siems et al., 2020; Wen et al., 2020), considering learning curves (Baker et al., 2017b; Domhan et al., 2015; Klein et al., 2017; Ru et al., 2021a) or zero-cost methods that are typically based on the statistics of an architecture or a single forward pass through the architecture (Abdelfattah et al., 2021; Lee et al., 2019; Mellor et al., 2021). We refer the interested reader to White et al. (2021) for a recent overview and comparison of such approaches.

2.6 Hardware-Awareness

Recently, many researchers also consider the resource consumption of neural networks, e.g., in terms of latency, model size, or energy consumption as objectives in NAS, since these are severely limited in many applications of deep learning. The importance of this fact is reflected by a whole line of research on manually designing top-performing yet resource-efficient architectures (Gholami et al. 2018; Howard et al. 2017; Iandola et al. 2016; Ma et al. 2018; Sandler et al. 2018; Zhang et al. 2018). Many NAS methods also consider such requirements for dense prediction tasks by now, e.g., Zhang et al. (2019), Liu et al. (2019a), Shaw et al. (2019), Lin et al. (2020), Li et al. (2020), Chen et al. (2020a, b), Bender et al. (2020) and Guo et al. (2020a). Typically this is achieved by either adding a regularizer penalizing excessive resource consumption to the objective function (Cai et al., 2019; Tan et al., 2019) or by multi-objective optimization (Elsken et al., 2019a; Lu et al., 2019). We refer the interested reader to Benmeziane et al. (2021) for a more general discussion on this topic (Fig. 3).

3 Semantic Image Segmentation

3.1 Design Principles

Semantic segmentation refers to the task of assigning a class label to each pixel of an image. The semantic segmentation model is trained to learn a mapping $f : \mathbb{R}^{w \times h \times c} \mapsto \mathbb{P}^{w \times h}$, where $w \times h$ refers to the spatial resolution, c to the number of input channels, and $\mathbb{P} = \{(p_0, \dots, p_{C-1}) \mid p_i \in [0, 1] \wedge \sum_{i=0}^{C-1} p_i = 1\}$, with C being the number of classes. In contrast to image classification, which focuses on global semantic information, this task necessitates effective integration of global and local information. However, the object location precision needed in this task far surpasses what is needed in dense prediction tasks like object detection, human pose estimation, etc. Further, as discussed in Sect. 2.2, the datasets for these tasks consists of a relatively small number of high-resolution images due to huge annotation effort requirement in terms of cost and time. Thus, NAS approaches for these tasks need to redefine search spaces to embody the task-specific needs while being computationally plausible. Popular data sets for semantic segmentation include PASCAL VOC (Everingham et al., 2015), Cityscapes (Cordts et al., 2016), ADE20K (Zhou et al., 2016, 2017), CamVid (Brostow et al., 2008a, b), and MS COCO (Lin et al., 2014).

Several years of manual neural architecture engineering for semantic segmentation have identified several concepts that can be used when designing a search space for NAS:

1. *Encoder–decoder macro-architecture* (Long et al., 2015; Ronneberger et al., 2015): “what is in the image?” and “where?” are the two questions that need to be addressed for each pixel of the image to solve the task of semantic segmentation. A popular way is to address these questions one at a time in a sequential manner. First by capturing long-range global or semantic context to address “what is in an image?”. Such representations can be generated by increasing the effective receptive field of a network. A common way to do so is by employing an encoder that gradually decreases the spatial resolution of its input while increasing the number of convolution layers to output high-level or semantically rich low-resolution feature representations. Doing so allows an increase in the receptive field in a computationally efficient way and is akin to image classification encoders. As such feature extractors pre-trained for image classification on ImageNet are often used as the encoder for this task. Following, to answer the “where?” question, a decoder is employed that gradually upsamples the high-level low-resolution features to generate high-resolution representations that encompass both semantic and fine features to produce the pixel-level semantic predictions.

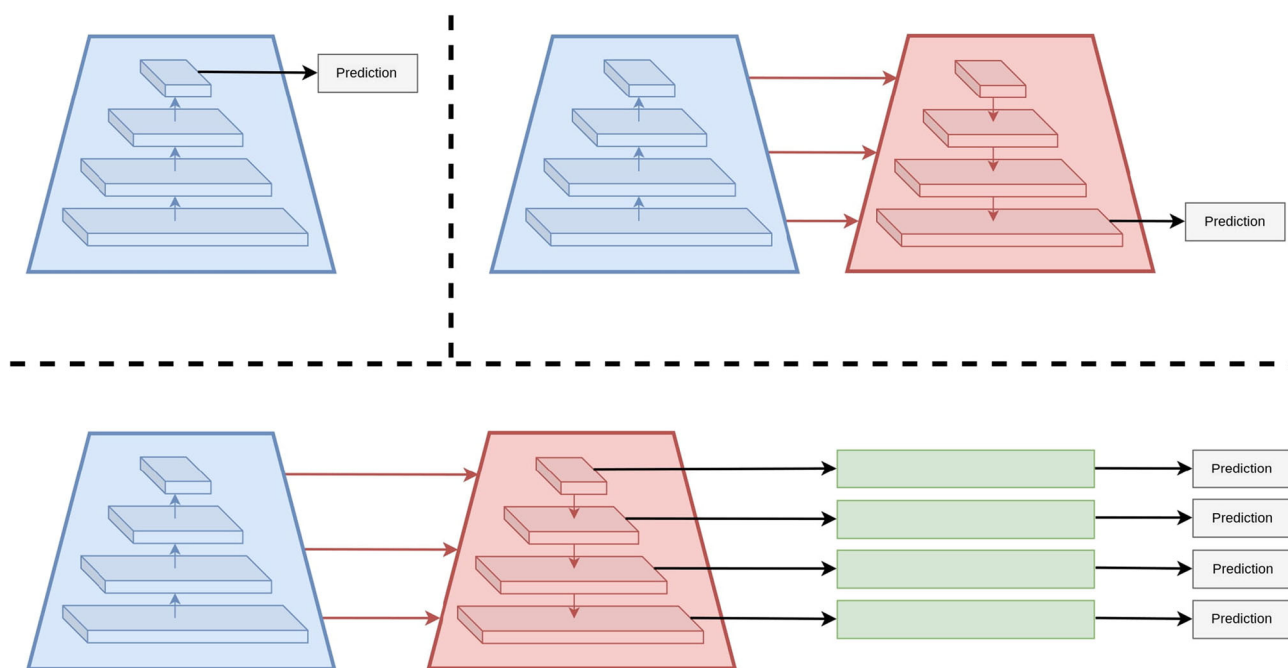


Fig. 3 High-level illustration of architectures employed for different tasks. Top left: typical encoder-like architecture (blue) for image classification problems; predictions are made based on low-resolution but semantically strong features. Top right: typical architecture for tasks like semantic segmentation; semantically strong features are generated for all scales through augmenting the encoder with a decoder (red). Bot-

tom: semantically strong features from all scales serve as the input for the object detection head (green); note that the feature maps within the encoder and decoder might be densely connected and feature maps in the decoder might be connected to any other feature map in the encoder as well as the decoder (Color figure online)

2. *Skip connections* (Ronneberger et al., 2015): while the encoder–decoder macro-architecture efficiently addresses the “what?” question, a decrease in spatial resolution also comes with the loss of localization features. As such the decoder has limitations on how accurate fine features it can recover from the semantically rich low-resolution representations. This problem is commonly addressed by incorporating skip connections at a higher resolution between the encoder and decoder. The popular U-Net architecture (Ronneberger et al., 2015) introduces these skip connections between identical spatial resolutions of the encoder and the decoder to supplement the representations on the decoder side with adequate spatial information resulting in improved object boundary segmentations.
3. *Common building blocks*: building blocks used in neural architectures for image classification, such as residual or dense blocks, can be readily re-used in the encoder architecture. Moreover, search spaces for neural cells for image classification can also be utilized when applying NAS to the encoder part of semantic segmentation.
4. *Multi-scale integration*: augmenting encoder–decoder-like macro-architectures with a specific component that supports multi-scale integration helps capture long-range dependencies. (Atrous) spatial pyramid pooling (ASPP)

(Chen et al., 2018a; He et al., 2015) is one popular approach to this.

5. *High-resolution macro-architecture* (Fourure et al., 2017; Wang et al., 2020a): high-resolution representations are necessary for semantic segmentation as it is a position-sensitive task but at the same time requires low-resolution representations to capture contextual features. The encoder–decoder architecture follows the scheme of sequentially generating high-level low-resolution features first from low-level high-resolution representations to obtain contextual information. And then recovering the spatially precise high-level high-resolution features from the computed low-resolution features. In contrast, a high-resolution network focuses on maintaining high-resolution representations that are semantically strong while being spatially precise throughout. These networks usually achieve this by maintaining high-to-low resolution convolution streams in parallel. In the aforementioned parallel streams, often, the low-level high-resolution features are aggregated with the upsampled high-level low-resolution features to boost high-resolution representations. Wang et al. (2020a) further improves the fusion scheme by opting for aggregation of features in a bidirectional manner. Thus, boosting both high- and low-representations. A NAS search space for

semantic segmentation based on these design principles can thus learn

- (a) building blocks/cells used in the encoder,
- (b) the downsampling strategy of the encoder,
- (c) the building blocks/cells of the decoder,
- (d) the upsampling strategy of the decoder,
- (e) where and how to add skip connections between encoder and decoder
- (f) how to perform multi-scale integration, and
- (g) building blocks to maintain strong high-resolution representations.

Learning only (a) and/or (c) would be similar to the so-called *micro* search since the backbone/decoder is fixed and the NAS algorithm only searches for the optimal structure of the building blocks. On the other hand, we shall refer to a *macro* search for approaches that optimize for at least one of the other components besides (a) and/or (c). While the components above are the canonical components to be optimized, we would also like to note that a promising direction for future work on NAS for semantic segmentation is to define search spaces that allow exploring architectures that do not follow the conventional design principles (Du et al., 2020).

3.2 NAS for Semantic Segmentation

In this section, we present various NAS approaches for semantic segmentation. For ease of discussion, we group the methods based on search strategies, namely, gradient-based, random and local search, reinforcement learning, and evolutionary algorithms. Subsequently, when applicable, we divide the methods based on the components of the network searched. Here, we assume a network consists of the backbone and multi-scale feature extractor that encapsulates the decoder as its two major components. Therefore, the possible sub-categories are backbone, multi-scale feature extractor, and joint. Table 1 presents the overview of the methods discussed in this section. Additionally, Table 2 reports the mIoU performance of methods benchmarked on Cityscapes and PASCAL VOC.

3.2.1 Gradient-Based

Multi-scale Feature Extractor Wu et al. (2019b) follows the encoder–decoder layout and focuses on finding a connectivity structure for the decoder. First, a pre-trained classifier is transformed into a densely connected network referred to as Fully Dense Network (FDN). Following the connectivity between the encoder and the decoder as well as within the decoder is made learnable. The approach aims to (1) select the most important decoder stages from the L num-

ber of stages, and (2) the input features for each selected stage. Each connection in FDN contains a binary weight to indicate its importance. However, to facilitate optimization, the weights are relaxed from discrete to continuous numbers between 0 and 1 via a novel sparse loss function that forces the weights to be sparse. Finally, the optimization is done in a differentiable manner with gradient descent. Further, to speed up the search process and reduce memory footprint, it uses lightweight MobileNet-V2 (Sandler et al., 2018) as the pre-trained image classifier and assumes the connectivity structure discovered is equally optimal for other stronger backbones. The search time amounts to about 18 h on a single Nvidia P100 GPU with 16GB memory.

Backbone Auto-DeepLab (Liu et al., 2019a) builds upon DARTS (Liu et al., 2019b) to perform a search for the optimal backbone. It uses the initially designed DARTS search space for image classification to learn optimal cell structure at the micro level while incorporating task-specific principles at the macro level. To perform a macro level search, it first employs a fixed two-layer stem structure at the beginning that scales down the spatial resolution by a factor of 2 at each layer. Following, a total of L number of layers with the unknown spatial resolution are optimized where each layer may differ in spatial resolution by at most 2. More specifically, a cell $C^{l,s}$ at layer l that outputs a tensor with spatial resolution s , can learn to process input tensors from previous layers with output tensors with resolutions $s/2$, s or $2s$. This is performed by continuously relaxing these discrete choices as it is done for the operation choices inside the cells. This results in multiple network outputs, each having a different spatial resolution. Each of these outputs is connected with an ASPP module (Chen et al., 2018a). For optimizing the architectural weights (of both cell and macro architecture), the authors utilize first-order DARTS where two disjoint sets generated by partitioning the training set are used for the alternating optimization strategy. The use of disjoint sets prevents the architecture from overfitting on the small training set that is usually available for semantic segmentation. Further, the search is performed using crops from half the resolution of the image to avoid high computation requirements caused by high-resolution images.

A line of follow-up work improves Auto-DeepLab in various directions. Shaw et al. (2019), Chen et al. (2020b), and Lin et al. (2020) search for efficient architectures (e.g., by means of latency) by adding a regularizer for hardware-costs and propose several powerful search spaces. Shaw et al. (2019) employs stronger building blocks such as inverted residual blocks while constraining the macro-architecture with manually designed search spaces of different sizes. Chen et al. (2020b) introduces zoomed convolution and group convolution in the search space for cell structure to obtain efficient building blocks while increasing flexibility

Table 1 Overview of different NAS methods for semantic segmentation

Method	Search space		Search method	Performance estimation		Resource efficiency considered	Search cost	Dataset	Searched on	Transferred to
	Backbone	Multi-level features extractor		Weight sharing	Pre-training					
Auto-DeepLab (Liu et al., 2019a)	✓	✗	GB	✓	✗	✗	Small	Cityscapes	PASCAL VOC, ADE20K	
SqueezeNAS (Shaw et al., 2019)	✓	✗	GB	✓	✗	✓	Small	Cityscapes		
CAS (Zhang et al., 2019)	✓	✓	GB	✓	✓	✓	Small	Cityscapes	CamVid	
HR-NAS (Ding et al., 2021)	✓	✓	GB	✓	✓	✗	Small	Cityscapes, ADE20K		
FNA (Fang et al., 2020)	✓	✗	GB	✓	✓	✓	Small	Cityscapes		
DPC (Chen et al., 2018b)	✗	✓	RS+LS	✗	✓	✗	high	Cityscapes	Pascal VOC	
FasterSeg (Chen et al., 2020b)	✓	✗	GB	✓	✗	✓	Small	Cityscapes	CamVid, BDD	
GAS (Lin et al., 2020)	✓	✗	GB	✓	✗	✓	small	Cityscapes	CamVid	
Nekrasov et al. (2019)	✗	✓	RL	✗	✓	✗	Small	PASCAL VOC, BSD, MS COCO	MS COCO, MPII, NYUDv2	
SparseMask (Wu et al., 2019b)	✗	✓	GB	✗	✓	✓	Small	PASCAL VOC	ADE20K, MSRA-B, BSD	
DCNAS (Zhang et al., 2021)	✓	✗	GB	✓	✗	✗	Small	Cityscapes	PASCAL VOC, ADE20K, PASCAL-Context	
NAS-Unet (Weng et al., 2019)	✓	✗	GB	✓	✗	✗	Small	PASCAL VOC	Promise L2, Chaos, NERVE	
C2FNAS (Yu et al., 2020c)	✓	✗	EA+RS	✓	✗	✗	High	MSD Pancreas	MSD 10	
V-NAS (Zhu et al., 2019)	✓	✗	GB	✓	✗	✗	Small	NIH pancreas	MSD lung, pancreas	

Search methods are evolutionary algorithm (EA), local search (LS), gradient-based (GB), reinforcement learning (RL), and random search (RS), respectively. Weight sharing refers to weight sharing via one-shot models (Bender et al., 2018; Pham et al., 2018). Pretraining refers to ImageNet pretraining. Since the search costs depend on the hardware and are also not explicitly mentioned in each paper, we only categorize them as “small” and “high”. We assign the cost label “small” to methods that can be run within a week on a server with eight GPUs, i.e., in less than 56 GPU days. Methods with “high” search costs typically employ a large-scale, distributed infrastructure, resulting in hundreds or thousands of GPU/TPU days of compute

through the choice of different channel expansion ratios. To do so, they propose a differentially searchable super kernel that restricts searching of the expansion ratio within a single conventional kernel and sampling from a pre-defined large set of ratios. Gumbel-Softmax is used to implement the aforementioned super kernel. Further, by having each of the L -layer cell outputs comprising two feature maps of different resolutions, the macro search space allows searching of multi-resolution branches. However, to limit the computation cost of the expanded search space, the search begins from $\times 8$ scaled-down version of the original resolution in contrast to $\times 4$ of Auto-DeepLab. On the other hand, Lin et al. (2020) eliminates the usual constraint of cell-sharing across the whole architecture by using a graph convolution network-based guiding module to model inter-cell relationships, thus, making efficient search possible. Additionally, Zhang et al. (2021) addresses DARTS' (and therefore also Auto-DeepLab's) problem of keeping the entire one-shot model in memory; this is done by sampling paths in the one-shot model rather than training the entire model at once, similar to the approaches by Xie et al. (2019) and Dong and Yang (2019). Due to the memory efficiency, the search is directly conducted on the target space and data set rather than employing a proxy task.

Semantic segmentation is particularly important for medical image analysis (Ronneberger et al., 2015) and consequently, NAS methods are also applied to optimize on medical image data sets. NAS-Unet (Weng et al., 2019) employs ProxylessNAS (Cai et al., 2019) to automatically search for a set of downsampling and upsampling cells that are connected using an Unet-like (Ronneberger et al., 2015) backbone. Zhu et al. (2019) considers 3D medical image segmentation and extends DARTS to encoder–decoder architectures used for volumetric medical image segmentation.

Multi-scale Feature Extractor Rather than searching for optimized building blocks for the encoder and/or the decoder, Wu et al. (2019b) propose to search for the connectivity pattern between the two components, which is typically fixed in other work. The encoder and decoder are first densely connected, where each connection is weighted by a real-valued parameter. This real-valued parameterization of the connections allows for gradient-based optimization as in DARTS (Liu et al., 2019b). The authors also propose a loss function for inducing a sparse connectivity pattern.

Joint While previous work considers optimizing either the encoder or decoder, Customizable Architecture Search (CAS) (Zhang et al., 2019) searches for both an optimal backbone and multi-scale feature extractor, however in a sequential manner. For the backbone, a normal cell (which preserves the spatial resolution and number of feature maps) and a reduction cell (which reduces the spatial resolution

and increases the number of feature maps) are optimized. Once these two cells have been determined, a multi-scale cell is optimized to learn how to integrate spatial information from the backbone. Following, HR-NAS (Ding et al., 2021) focuses on the better encoding of multiscale image contexts in the search space while maintaining high-resolution representations that are easily customizable for different dense prediction tasks in addition to semantic segmentation. To do so, they define a multi-branch search space comprising convolutions and transformers. It defines the constitution of the network in terms of two modules, namely, parallel and fusion. Starting with a high-resolution branch at a feature resolution of $1/4$ of the input image size, the fusion module gradually adds high-to-low resolution branches while the parallel module connects the multi-resolution branches. The searching block used for both modules is the same and consists of two paths. The first path is a MixConv (Tan & Le, 1907) whilst the second path is a lightweight transformer that aims to capture better global contexts. To enable searching for various tasks, HR-NAS introduces a resource-aware channel/query-wise fine-grained search strategy that adopts the progressive shrinking NAS paradigm to generate lightweight models by flexibly removing convolutional channels and transformer queries while exploring optimal features for each task.

3.2.2 Random and Local Search

Chen et al. (2018b) employ NAS for dense prediction tasks in order to search for a better multi-scale feature extractor called dense prediction cell (DPC) given a fixed backbone network. The proposed search space is a micro search space that contains, e.g., atrous separable convolutions with different rates or average spatial pyramid pooling inspired by DeepLabv3 (Chen et al., 2018a). They run a combination of random search and local search to optimize the dense prediction cell given a fixed, pretrained backbone, which, despite the use of a series of proxy tasks, still required 2600 GPU days. In follow-up work, Nekrasov et al. (2020) extend their work to semantic video segmentation by learning a *dynamic cell* that learns to aggregate the information coming from previous and current frames to output segmentation masks.

3.2.3 Reinforcement Learning

Multi-scale Feature Search Nekrasov et al. (2019) considers a fixed encoder network and searches for an optimal decoder architecture together with the respective connections to the encoder layers. The decoder architecture is modeled as a sequence of cells sharing the same structure that processes the inputs from the encoder layers. The authors utilize various heuristics to speed-up architecture search. For example, they freeze the weights of the encoder network and train only the decoder part (as already done in DPC) and early-stop training

of architectures with poor performance. Moreover, a knowledge distillation loss (Hinton et al., 2015) is employed as well as an auxiliary cell to reduce the training time. Rather than using random and local search, a controller trained with reinforcement learning is employed to sample candidate architectures, similar to Zoph et al. (2018).

3.2.4 Evolutionary Algorithm

Backbone for the task of 3D medical image segmentation, Coarse-to-Fine NAS (C2FNAS) (Yu et al., 2020c) uses a search space inspired by the one employed in Auto-DeepLab (Liu et al., 2019a) and an evolutionary strategy operating on clusters of similar networks to search for the macrostructure of their model, whilst the operation choices inside the cells of the macrostructure are randomly sampled similarly to the protocol by Li and Talwalkar (2019).

4 Object Detection

4.1 Design Principles

Object detection (Liu et al., 2020) refers to the task of identifying if/how many objects of predetermined categories are present in input (e.g., an image) and, for each identified object, determining its category as well as its spatial localization. Spatial localization can be represented in different ways, with the most common one being a 2D bounding box in image space, encoded by a 4D real-valued vector. However, other representations, such as pixel-wise segmentation, are possible as well. In this case, the task is referred to as instance segmentation. However up to now, there is no work in this direction to the best of our knowledge, so we focus on 2D bounding box estimation. We note that in contrast to semantic segmentation, deep learning-based object detection often has a post-processing step that maps from dense network outputs to a sparse set of object detections, e.g., using non-maximum suppression. However, this post-processing is typically fixed and not used during training; and thus also ignored during NAS (we note that applying NAS to this post-processing would be an interesting future direction). Moreover, deep learning-based object detection can be split into one-stage and two-stage approaches. Two-stage approaches first identify the presence and extent of an arbitrary object at a position and thereupon apply a region classifier to the identified object region to classify the category of the object and (optionally) refine its spatial localization. In contrast, single-stage approaches directly predict the presence of an object, its class, as well as its spatial localization in a single forward pass.

Since objects can have vastly different scales, typically multi-scale approaches are applied for single-stage object

detection. This can be achieved by either attaching “detection heads” at layers of different spatial resolutions or by combining features of different layers; effectively, this results in certain network outputs (those corresponding to lower resolutions) specializing on larger objects and higher resolution outputs on smaller ones. In this case, the dense prediction task can be framed as $f : \mathbb{R}^{w \times h \times c} \mapsto [\mathbb{P}^{w \times h} \times \mathbb{R}^{w \times h \times b}, \mathbb{P}^{w/2 \times h/2} \times \mathbb{R}^{w/2 \times h/2 \times b}, \dots]$, where $w \times h$ refers to the spatial resolution, c to the number of input channels, b to the parameters encoding the spatial localization, and $\mathbb{P} = \{(p_{-1}, p_0, \dots, p_{C-1}) | p_i \in [0, 1] \wedge \sum_{i=-1}^C p_i = 1\}$, with C being the number of classes and -1 corresponding to the “no object” class.

Many of the design principles of semantic segmentation carry over to object detection. However, there are also notable differences:

- Since the network requires a dense and multi-scale output, a further design choice is how “detection heads” generating these multi-scale outputs are attached to the main network. The heads’ architecture itself is another open design choice.
- Two-stage object detection can impose complex interdependencies between the architectures of the two stages, making the design of a search space covering both stages together challenging.

4.2 NAS for Object Detection

In this section, we present several NAS methods for object detection. We first group the approaches based on the strategy used for search, namely, evolutionary algorithms, gradient-based, reinforcement learning, and local search. Following, we sub-categorize the methods based on the components of the network searched, if applicable. Here, we assume a network consists of the backbone, multi-scale feature extractor, and the task-specific head as its three major components. Table 3 presents the overview of the methods discussed in this section.

4.2.1 Evolutionary Algorithms

Backbone Acknowledging the gap between the tasks of image classification and object detection that can lead to sub-optimal network architectures, DetNAS (Bender et al., 2020; Chen et al., 2019) proposes a framework to facilitate backbone search for object detection. It consists of three steps. In the first step, DetNAS pre-trains a supernet in a path-wise (Guo et al., 2020b) manner where in each iteration a single path is sampled for feedforward and backward propagation. This ensures that the relative performance of candidate networks is reflected by the supernet. Following, the supernet is fine-tuned for the detection task by adding

Table 2 Performance comparison of various NAS methods for semantic segmentation

Method	Pre-training		MS COCO		Dataset (mIoU)		PASCAL VOC		Params (M)	
	ImageNet	Cityscapes	Val (%)	Test (%)	Val (%)	Test (%)	Val (%)	Test (%)	Val (%)	MACs (B)
Auto-DeepLab-S (Liu et al., 2019a)	✗	✗	✗	79.7	79.9	333.3	71.68	—	—	10.2
SqueezeNAS LAT Large (Shaw et al., 2019)	✗	✓	✓	—	—	333.3	78.31	82.5	—	10.2
CAS (Zhang et al., 2019)	✓	✓	✓	73.6	72.5	19.6	—	—	—	1.9
HR-NAS (Zhang et al., 2019)	✓	✗	✗	71.6	70.5	—	—	—	—	—
FNA (Fang et al., 2020)	✓	✗	✗	75.9	—	4.6	—	—	—	3.9
DPC (Chen et al., 2018b)	✗	✓	✓	—	78.0	313.9	—	—	—	5.7
FasterSeg (Chen et al., 2020b)	✗	✗	✗	80.9	82.7	—	—	87.9	—	—
GAS (Lin et al., 2020)	✓	✗	✗	73.1	71.5	28.2	—	—	—	4.4
Nekrasov et al. (2019)	✓	✗	✗	72.3	73.5	—	—	—	—	—
SparseMask (Wu et al., 2019b)	✓	✗	✗	—	—	—	78.0	—	4.5	2.6
DCNAS (Zhang et al., 2021)	✓	✗	✗	—	83.1	—	—	85.4	—	52
				—	—	—	—	86.9	—	—

— denotes that the metric has not been reported for the corresponding method in its respective manuscript

Table 3 Overview of different NAS methods for object detection

Method	Search space		Search method		Performance estimation		Resource efficiency considered	Search cost	Dataset	Type
	Multi-level features extractor	Task-specific head(s)	EA	GB	Weight sharing	Pre-training				
DetNAS (Chen et al., 2019)	✓	✗	EA	✓	✓	✓	✗	Small	PascalVOC, MS COCO	TS/OS
NATS (Peng et al., 2019)	✓	✗	GB	✓	✓	n/a	n/a	Small	MS COCO	TS/OS
TuiNAS (Bender et al., 2020)	✓	✗	RL	✓	✗	✓	✓	Small	MS COCO	OS
MobileDets (Xiong et al., 2020)	✓	✗	RL	✓	✗	✓	✓	Small	MS COCO	OS
SP-NAS (Jiang et al., 2020)	✓	✗	LS	✓	✓	✓	✓	Small	MS COCO, ECP, PASCAL, VOC, BDD	TS
FNA (Fang et al., 2020)	✓	✗	GB	✓	✓	✓	✓	Small	MS COCO	OS
Joint-DetNAS (Yao et al., 2021)	✓	✗	HL	✗	✓	✗	✗	High	MS COCO	TS
NAS-FPN (Ghiasi et al., 2019)	✗	✓	RL	✗	✗	✗	✗	High	MS COCO	TS/OS
AutoDet (Li et al., 2021)	✗	✓	SA	✗	✗	✗	✗	Small	MS COCO	TS/OS
OPANAS (Liang et al., 2021)	✗	✓	EA	✗	✓	✗	✗	Small	MS COCO	TS/OS

Table 3 continued

Method	Search space		Search method		Performance estimation		Resource efficiency considered	Search cost	Dataset	Type
	Multi-level feature extractors	Task-specific head(s)	Weight sharing	Pre-training	Searched on	Transferred to				
MinasFPN (Chen et al., 2020a)	✓	✗	RL	✗	✓	✓	high	MS COCO	OS	
Auto-FPN (Xu et al., 2019a)	✓	✓	GB	✓	✓	✓	Small	Pascal VOC, MS COCO, BDD	VG, ADE TS/OS	
NAS-FCOS (Wang et al., 2020b)	✓	✓	RL	✗	✓	✗	Small	Pascal Voc COCO	MS OS	
FAD (Zhong et al., 2020a)	✗	✓	GB	✓	n/a	✗	Small	Pascal Voc, MS COCO	OS	
Hit-detector (Guo et al., 2020a)	✓	✓	GB	✓	✓	✓	Small	MS COCO	TS/OS	
SM-NAS (Yao et al., 2020)	✓	✓	EA	✗	✓	✓	High	MS COCO	Pascal TS/OS	
iNAS (Gu et al., 2021)	✓	✓	EA	✗	✓	✓	Small	DUTS-TR ECSSD, DUT-O, UTS- TE, HKU- IS, PASCAL- S	TS/OS	
SpineNet (Du et al., 2020)	✓	✗	RL	✗	✗	n/a	High	MS COCO	TS/OS	

Search methods are evolutionary algorithm (EA), local search (LS), gradient-based (GB), reinforcement learning (RL), and simulated annealing (SA), respectively. Weight sharing refers to weight sharing via one-shot models (Bender et al., 2018; Pham et al., 2018). Pretraining refers to ImageNet pretraining. Since the search costs depend on the hardware and are also not explicitly mentioned in each paper, we only categorize them as “small” and “high”. We assign the cost label “small” to methods that can be run within a week on a server with eight GPUs, i.e., in less than 56 GPU days. Methods with “high” search costs typically employ a large-scale, distributed infrastructure, resulting in hundreds or thousands of GPU/TPU days of compute

a detection head, again in a path-wise manner. Finally, the architecture search is executed on the trained supernet by using an evolutionary controller to select and evaluate paths in the supernet. The search space is designed based on the light-weight ShuffleNetv2 (Ma et al., 2018) block to be computationally feasible.

Multi-scale Feature Search OPANAS (Liang et al., 2021) proposes a novel search space of FPNs comprising six heterogeneous information paths, namely, top-down, bottom-up, fusing-splitting, scale-equalizing, skip-connect, and none. Here, the FPN candidates are represented by a densely-connected acyclic graph, and an efficient one-shot search method is employed to obtain the optimal path aggregation architecture. To do so, OPANAS first trains a super-net and then uses an evolutionary algorithm to choose the optimal candidate. It performs the search on an input size of 800×500 and samples 1/5 images from the training set to reduce the search cost. The optimal architecture found then is trained on 1333×800 input resolution.

Joint SM-NAS (Yao et al., 2020) presents a two-stage coarse-to-fine search strategy to find an optimal end-to-end object detection framework. It focuses on first searching for the best combination of backbone, multi-scale feature extractor, region-proposal network as well as detection head (either one- or two-stage), with a set of possible candidates for each component (e.g. ResNet or MobileNet V2 as a backbone or different versions of FPNs for multi-scale feature fusion). To do so, some initial random combination of modules is generated and an evolutionary algorithm is used to mutate the best architecture on the Pareto front to provide suitable candidate models. Following, the best-performing combinations of these components are then fine-tuned on a more fine-grained level, e.g., by optimizing the number of channels in the chosen backbone to achieve the optimal balance of low-level features for localization and high-level features for classification. Further, iNAS (Gu et al., 2021) proposes a device-aware search scheme that trains salient object detection models once and subsequently finds high-performing models with low latency on multiple devices. It employs latency-group sampling where according to a given latency lookup table their layer-wise search space is divided into several latency groups. Next, to obtain a model in a specific latency group, blocks are sampled at each layer of the latency group. This ensures balanced samples in the global latency range when the groups are divided adequately even though the local latency group can remain imbalanced resulting in an overall balanced exploration of the search space.

4.2.2 Gradient-Based

Backbone Rather than searching for architecture from scratch, Peng et al. (2019) proposes to transform a given, well-performing backbone according to the need of the task. Consequently, aiming to increase the effective receptive field size of convolutions with dilation rates to capture long-range contextual features with minimal loss of fine features, the authors search over various dilation rates. For each dilation rate, channels are grouped to allow for different dilation rates for different groups. The parameters of convolutional kernels are inherited from a pre-trained model and shared across all rates to avoid additional parameters. Gradient-based architecture search on a continuous relaxation of the search space is used to then search for the optimal dilation rates for each channel group. In contrast, Fang et al. (2020) adapts the backbone for object detection tasks by changing the depth, width, or kernels of the network via parameter remapping techniques involving batch normalization statistics, weight importance, and dilation.

Multi-scale Feature and Head Xu et al. (2019a) propose Auto-FPN, a method for searching for a multi-scale feature extractor and a detection head, based on a continuous relaxation and gradient-based optimization as done by Liu et al. (2019b). Again, a cell-based search space is used, for both components. The search is conducted in a sequential manner (i.e., the FPN is searched first and the head afterward) as DARTS. Since the employed search strategy requires keeping the whole one-shot model in memory, it does not allow for a joint optimization in the considered setting. Zhong et al. (2020a) also employs DARTS to search for a detection head. To mitigate memory problems, they propose a more efficient scheme for sharing representations across operations with different receptive field sizes by re-using intermediate representations.

Joint Guo et al. (2020a) show that the combination of individually searched NAS backbone and multi-scale feature extractor for object detection performs worse than the combination of manually designed backbone and NAS searched multi-scale feature extractor. As a consequence, they propose searching over all the components of an object detection network. The main challenge of searching end-to-end for such a complex task is the ease with which it can become computationally infeasible. To overcome this problem, Guo et al. (2020a) proposes a hierarchical search. The first search phase is conducted on a small proxy task with a rich search space [build around FBNet (Wu et al., 2019a)] for all three components while pruning building blocks that are unlikely to be optimal. Notably, this allows starting with the same set of candidate operations for all three components. Further, by imposing a regularizer enforcing sparsity among

the architectural parameters in the one-shot model, suboptimal candidates can naturally be pruned away. a rich search space [build around FBNet (Wu et al., 2019a)] for all three components is explored with the goal of shaping the search space by pruning building blocks that are unlikely to be optimal. Notably, this allows starting with the same set of candidate operations for all three components, while in prior work the set of candidates is typically adapted to the specific component to be optimized (Xu et al., 2019a). By imposing a regularizer enforcing sparsity among the architectural parameters in the one-shot model in the first phase, suboptimal candidates can naturally be pruned away. Following, in the second search phase, the resulting pruned sub-space is used to determine an optimal architecture. Both search phases employ gradient-based optimization for efficiency. Furthermore, the authors penalize architectures with high computational costs by adding a proper regularization term.

4.2.3 Reinforcement Learning

Backbone Bender et al. (2020) propose TuNAS, inspired by ProxylessNAS (Cai et al., 2019) and ENAS (Pham et al., 2018), for image classification and also evaluate it on object detection, with only minor hyperparameter adjustments required. In contrast to most other work, Bender et al. (2020) trains a one-shot model from scratch directly on the target task rather than employing pretraining. To improve the scalability with respect to the search space, the authors propose a more aggressive weight sharing across candidate choices, e.g., by sharing filter weights across convolutions with a different number of filters. Furthermore, the memory footprint when training the one-shot model is dramatically reduced by “rematerialization”, i.e., re-computing intermediate activations rather than storing them. The authors also propose a novel hardware regularizer allowing to find models closer to the desired hardware cost. In a follow-up work (Xiong et al., 2020), the performance of TuNAS is further improved due to more powerful search space.

Multi-scale Feature and Head Search Ghiasi et al. (2019) employ a reinforcement learning-based NAS framework (Zoph et al., 2017, 2018) to search for NAS-FPN, an improved feature pyramid network (FPN) (Lin et al., 2017) yielding multi-scale features. In follow-up work, Chen et al. (2020a) extends the search space and employs Mnas-Net (Tan et al., 2019) as a search method for not only optimizing performance but also latency to find efficient networks. This is in contrast to NAS-FPN, where lightweight architectures are searched after manually adapting non-searchable components to be efficient. As both NAS-FPN and Mnas-FPN are based on expensive, black-box NAS methods that train around 10,000 architectures, they require substantial compu-

tational resources to be run, in the order of hundreds of TPU days.

While the aforementioned work focuses on the FPN module, Wang et al. (2020b) additionally optimizes the object detection head on top of the multi-scale features. This is done by using RL to the first search for an FPN-like module and afterward for a detection head. For the FPN module, similar to Ghiasi et al. (2019), the RL controller chooses feature maps from a list of candidates, an elementary operation to process, and in which way to merge it with another candidate. Once an optimal FPN is found, it is used to search for a suitable head. While typically the weights of the head are shared across all levels of the feature pyramid, Wang et al. (2020b) also searches over an index indicating from where on to share weights, while all layers of the head architecture before the index can have different weights for different pyramid levels. As the backbone architecture is not optimized, they pre-compute the output features from the backbone to make the search more efficient.

Joint Du et al. (2020) propose to search for a single network covering both backbone and multi-scale feature extractor components. This approach permutes layers of the network and searches for a better connectivity pattern between them. We highlight that for this search space consisting of layer permutations it is unclear how one-shot models could be employed and thus the authors rely on the computationally more expensive black-box optimization via RL.

4.2.4 Local Search

Backbone Jiang et al. (2020) modify an existing, well-performing, and pretrained backbone by applying network morphisms (Wei et al., 2016), which are commonly used in NAS (Cai et al., 2018a,b; Elsken et al., 2017, 2019a), to improve the backbone. Since network morphisms inherit the performance of the parent network to the child network, the child network does not need to be trained from scratch and thus the authors avoid pre-training all candidate architectures on ImageNet, which would be infeasible. In the first search phase, a purely sequential model is optimized, while a second search phase adds parallel branches to enable more powerful architectures.

4.2.5 Hill Climbing

Backbone Joint-DetNAS (Yao et al., 2021) integrates Neural Architecture Search, pruning, and knowledge distillation (KD) into a unified NAS framework. Given a base object detector, Joint-DetNAS is capable of deriving a student detector with high performance without the need for any additional training. The proposed algorithm of the unified framework comprises two processes, namely, dynamic distillation, and

student morphism. The former process finds the optimal teacher by training an elastic teacher pool with an integrated progressive shrinking strategy. The teacher detectors can be sampled from this pool without any additional cost. On the other hand, the latter process uses a weight inheritance strategy while pruning to enable flexible updating of students' architecture. This allows the full utilization of weights from predecessor architectures for efficient searching. Contrary to other mainstream NAS methods, Joint-DetNAS aims to evolve a given base detector to restrict the search space exploration. It employs a hill climbing approach for searching as it is highly compatible with the used weight inheritance strategy while being efficient in evolving student-teacher pairs. During the search, Joint-DetNAS starts with an initial student-teacher pair and then optimizes the student and the teacher alternatively. In each iteration, either the student is updated by applying an action (layer or channel pruning, adding layers, and rearranging layers) or the teacher is mutated by changing the depth or width in each backbone stage. The evaluation of the pair can be done only in a few epochs of training due to the advantages of the weight inheritance scheme.

4.2.6 Simulated Annealing

Multi-scale Feature AutoDet (Li et al., 2021) proposes a novel search space that focuses on finding the informative connections between multi-scale features for feature pyramids. It frames the search process as a combinatorial optimization problem that utilizes simulated annealing (SA) to address it. AutoDet generates a feature pyramid in two steps. First, for each output layer, several input layers are determined. Following, the best fusion operation for the determined input layers is selected. Hence, the search space consists of searching the connection topology and operation parameters. Subsequently, optimizing SA starts with a high initial temperature and then gradually cools it down as the number of iterations progresses. The high temperature enables AutoDet to accept solutions that are worse than a given current solution with more frequency. As the temperature cools down, SA narrows down towards the optimal solution of the search space. The efficiency of SA allows AutoDet to perform a search with a high input resolution of 1333×800 within a reasonable period (2.2 GPU days).

4.3 Quantitative Analysis

In Table 4, we report the quantitative results of the best-performing architecture for each of the NAS methods discussed in Sect. 4.2 that are evaluated on the MS COCO dataset. We also report a few approaches, where the search is performed for the classification task, and the discovered architecture is directly transferred to address the object detec-

tion task. It should be noted due to drastically varying model complexity and changing input image resolution sizes from one method to another, making a direct comparison is not feasible. On comparing, NASNet-A, a direct transfer approach, among the peer NAS for object detection approaches of similar input resolution (such as NAS-FCOS, FAD, NAS-FPN, AutoDet, etc.), we observe most of them outperform it. This implies using NAS directly on dense prediction approaches tends to yield relatively more optimal networks than direct transfer. It is mainly because the upstream task is considerably different than the downstream task, thus, creating an optimization gap that is very significant. Further, NAS methods that operate on higher input resolution tends to achieve better mAP score, reinforcing the importance of high-resolution images for accurate object detection. Moreover, methods utilizing low-resolution images or proxy tasks with object detection-specific search space in their search phase, are capable of discovering efficient object detectors. However, they tend to be relatively sub-optimal in terms of performance compared to the ones that operate on high-resolution and the task itself.

5 Outlook: Promising Application Domains and Future Work

Most of the NAS research on dense prediction tasks primarily focuses on the tasks of object detection and semantic segmentation tasks. However, many other dense prediction tasks can benefit from the application of NAS. Following, we briefly discuss such NAS methods. Table 5 presents the overview of the methods discussed.

For example, *disparity estimation* can be solved in an end-to-end fashion with encoder–decoder architectures (Mayer et al., 2016). The first studies in this direction have already been conducted. Saikia et al. (2019) propose AutoDispNet, which extends the typical search space from image classification consisting of a normal and a reduction cell by an upsampling cell in order to search for encoder–decoder architectures. The first order approximation of DARTS is used to allow an efficient search, followed by a hyperparameter optimization for the discovered architectures using the popular multi-fidelity Bayesian optimization method BOHB (Falkner et al., 2018). Cheng et al. (2020b) build upon AutoDispNet by also searching for a matching network on top of the feature extractor, inspired by recently manually designed networks for disparity estimation. Architectures discovered for disparity estimation (Saikia et al., 2019) or semantic segmentation (Nekrasov et al., 2019) have also been evaluated on *depth estimation*.

AutoPose (Gong et al., 2020) framework focuses on searching for multi-branch scales and network depth to achieve accurate and high-resolution 2D human pose esti-

Table 4 Performance comparison of various NAS methods for object detection

Method	Image size	Dataset (mIoU)			Params (M)	
		MS COCO				
		Val (%)	Test (%)	MACs (B)		
DetNAS + FPN	(Chen et al., 2019)	≥ 800	42.0	—	3.8	—
DetNAS + RetinaNet		≥ 800	33.3	—	—	—
NATS + CascadeRCNN	(Peng et al., 2019)	800×1200	42.0	—	—	—
NATS + RetinaNet		800×1200	37.3	—	—	—
TuNAS	(Bender et al., 2020)	320×320	—	22.5	—	—
MobileDets	(Xiong et al., 2020)	320×320	28.9	28.5	2.8	7.2
SP-NAS + CascadeRCNN	(Jiang et al., 2020)	1280×1280	—	49.1	—	—
FNA + RetinaNet	(Fang et al., 2020)	1088×800	—	33.9	133	11.7
Joint-DetNAS	(Yao et al., 2021)	1080×720	—	43.9	145.7	—
NAS-FPN + AmoebaNet	(Ghiasi et al., 2019)	1280×1280	—	48.3	2086	166.5
NAS-FPN + FPN		1280×1280	—	46.6	2633	103.9
AutoDet + SSD	(Li et al., 2021)	512×512	—	37.7	144.2	55.4
AutoDet + CascadeRCNN		1280×1280	—	47.3	—	—
OPA-FPN + CascadeRCNN	(Liang et al., 2021)	900×1200	42.8	—	225	50.6
OPA-FPN + CascadeRCNN		900×1200	—	52.2	432	80.3
OPA-FPN + RetinaNet		900×1200	38.0	—	207	36.5
MNAS-FPN	(Chen et al., 2020a)	320×320	—	25.5	0.8	3.5
Auto-FPN + RPN	(Xu et al., 2019a)	800×800	40.5	—	—	32.6
Auto-FPN + SSD		512×512	31.8	—	—	33.3
NAS-FCOS	(Wang et al., 2020b)	≥ 800	—	46.1	361	90
FAD	(Zhong et al., 2020a)	≥ 800	—	46.4	—	—
Hit-Detector + RPN	(Guo et al., 2020a)	800×1200	41.4	44.5	272	27
Hit-Detector + RetinaNet		800×1200	36.9	—	—	33
SM-NAS	(Yao et al., 2020)	800×1333	—	45.9	—	—
SpineNet + MaskRCNN	(Du et al., 2020)	1280×1280	—	49.3	520	79
SpineNet + RetinaNet		1280×1280	—	52.1	1885	163
NASNet-A (B)	(Zoph et al., 2018)	1200×1200	43.2	43.1	—	—
PC-DARTS + SSD (B)	(Xu et al., 2019b)	320×320	—	28.9	—	1.2
FairNAS-A (B)	(Chu et al., 2021)	—	—	32.4	392	—
Cream-S (B)	(Peng et al., 2020)	—	33.2	—	0.3	—

We report the best performance achieved by each method. — denotes that the metric has not been reported for the corresponding method in its respective manuscript. The image size is in $H \times W$ format and $\geq D$ denotes the shortest size has minimum D size. Additionally, B denotes methods that search on classification task while reporting results on the object detection task as well

mation. It employs a novel bi-level optimization method that employs reinforcement learning to search at network-level architecture and a gradient-based method for cell-level search. Additionally, HR-NAS (Ding et al., 2021) that prioritizes learning high-resolution representations due to its efficient fine-grained search strategy as discussed in Sect. 3 is capable of finding optimal architecture for the tasks of human pose estimation and 3D object detection.

Ulyanov et al. (2018) showed that the structure of an encoder–decoder architecture employed as a generative model is already sufficient to capture statistics of natural images without any training. Thus, such architectures can

be seen as a “deep image prior” (DIP), which can be used to parameterize images. On a variety of tasks, such as *image denoising, super-resolution or inpainting*, a natural image could successfully be generated from random noise and a randomly initialized encoder–decoder architecture. As the authors noted that the best results can be obtained by tuning the architecture for a particular task, Ho et al. (2020) and Chen et al. (2020c) employed NAS to search for deep image prior architectures via evolution and reinforcement learning, respectively. Differentiable architecture search has also been adapted for image denoising by Gou et al. (2020).

Table 5 Overview of different NAS methods for various dense prediction tasks

Task	Method	Search space Backbone	Multi-level features extractor	Search method		Performance estimation	Weight sharing	Pre-training	Search costs	Datasets
				GB	EA					
Disparity estimation	AutoDispNet (Saikia et al., 2019)	✓	✓	GB	EA	✓	✗	✗	Small	FlyingThings3D (Mayer et al., 2016)
	LEAStereo (Cheng et al., 2020b)	✓	✓	GB	EA	✓	✗	✗	Small	SceneFlow (Mayer et al., 2016)
Human pose estimation	AutoPose (Gong et al., 2020)	✓	✓	GB + RL	EA	✓	✗	✗	Small	MS COCO
	HR-NAS (Ding et al., 2021)	✓	✓	GB	EA	✓	✗	✗	Small	MS COCO
Deep image prior	(Ho et al., 2020)	✓	✓	EA	EA	✓	✗	✗	Small	DIP (Ulyanov et al., 2018)
	(Chen et al., 2020c)	✓	✓	RL	EA	✓	✗	✗	Small	DIP (Ulyanov et al., 2018)
Panoptic segmentation	Auto-panoptic (Wu et al., 2020)	✓	✓	GB + EA	EA	✓	✓	✓	Small	MS COCO, ADE20K
	3D detection and segmentations	SPVNAS (Tang et al., 2020)	✓	✓	GB + EA	EA	✓	✗	Small	SemanticKITTI (Behley et al., 2019)

Search methods are evolutionary algorithm (EA), gradient-based (GB), and reinforcement learning (RL), respectively. Weight sharing refers to weight sharing via one-shot models (Bender et al., 2018; Pham et al., 2018). Pretraining refers to ImageNet pretraining. Since the search costs depend on the hardware and are also not explicitly mentioned in each paper, we only categorize them as “small” and “high”. We assign the cost label “small” to methods that can be run within a week on a server with eight GPUs, i.e., in less than 56 GPU days. Methods with “high” search costs typically employ a large-scale, distributed infrastructure, resulting in hundreds or thousands of GPU/TPU days of compute

Other promising tasks are *panoptic segmentation* (which also covers instance segmentation) with some first work by Wu et al. (2020) and 3D detection and segmentation (Tang et al., 2020). Finally, *optical flow estimation* (Dosovitskiy et al., 2015; Ilg et al., 2017; Sun et al., 2018) is a problem that has not been considered by NAS researchers so far, and it is conceivable that NAS methods could further improve performance on this task.

6 Conclusion

In this manuscript, we discussed the application of NAS for dense prediction tasks. We presented a detailed discussion of approaches for the two core dense prediction tasks, namely, *semantic segmentation* and *object detection*. These tasks are closely related while having different degrees of object localization and inter-object distinction requirements. Consequently, we described the diverse approaches that have been proposed to address them. Further, we also discussed the application of NAS for other promising dense prediction tasks where its exploration has been limited such as *panoptic segmentation*, *depth estimation*, *image inpainting* etc. We hope that our work will serve as a good starting point for new researchers delving into these areas.

Acknowledgements This work was funded by Robert Bosch GmbH and the German Research Foundation (DFG) Emmy Noether Program grant number 468878300.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł., & Lane, N. D. (2021). Zero-cost proxies for lightweight NAS. In *International conference on learning representations*. <https://openreview.net/forum?id=0cmMMY8J5q>
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International conference on learning representations*.
- Baker, B., Gupta, O., Naik, N., & Raskar, R. (2017a). Designing neural network architectures using reinforcement learning. In *ICLR*.

- Baker, B., Gupta, O., Raskar, R., & Naik, N. (2017b). Accelerating neural architecture search using performance prediction. In *NIPS workshop on meta-learning*.
- Behley, J., Garbade, M., Millioto, A., Quenzel, J., Behnke, S., Stachniss, C., & Gall, J. (2019). Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9297–9307).
- Bender, G., Kindermans, P. J., Zoph, B., Vasudevan, V., & Le, Q. (2018). Understanding and simplifying one-shot architecture search. In *International conference on machine learning*.
- Bender, G., Liu, H., Chen, B., Chu, G., Cheng, S., Kindermans, P. J., & Le, Q. V. (2020). Can weight sharing outperform random architecture search? An investigation with tunas. In *The IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Benmeziane, H., Maghraoui, K. E., Ouarnoughi, H., Niar, S., Wistuba, M., & Wang, N. (2021). A comprehensive survey on hardware-aware neural architecture search. [arXiv:2101.09336](https://arxiv.org/abs/2101.09336)
- Brostow, G. J., Fauqueur, J., & Cipolla, R. (2008a). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2), 88–97.
- Brostow, G. J., Shotton, J., Fauqueur, J., & Cipolla, R. (2008b). Segmentation and recognition using structure from motion point clouds. In *ECCV (1)* (pp. 44–57).
- Cai, H., Chen, T., Zhang, W., Yu, Y., & Wang, J. (2018a). Efficient architecture search by network transformation. In *Association for the advancement of artificial intelligence*.
- Cai, H., Yang, J., Zhang, W., Han, S., & Yu, Y. (2018b). Path-level network transformation for efficient architecture search. In *International conference on machine learning*.
- Cai, H., Zhu, L., & Han, S. (2019). ProxylessNAS: Direct neural architecture search on target task and hardware. In *International conference on learning representations*.
- Chen, B., Ghiasi, G., Liu, H., Lin, T. Y., Kalenichenko, D., Adam, H., & Le, Q. V. (2020a). Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices. In *The IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 834–848.
- Chen, L., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. [arXiv:1706.05587](https://arxiv.org/abs/1706.05587)
- Chen, L. C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., & Shlens, J. (2018b). Searching for efficient multi-scale architectures for dense image prediction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 8699–8710). Curran Associates Inc.
- Chen, W., Gong, X., Liu, X., Zhang, Q., Li, Y., & Wang, Z. (2020b). Fasterseg: Searching for faster real-time semantic segmentation. In *International conference on learning representations*. <https://openreview.net/forum?id=BJgqQ6NYvB>
- Chen, X., & Hsieh, C. J. (2020). Stabilizing differentiable architecture search via perturbation-based regularization. In *International conference on machine learning, PMLR* (pp. 1554–1565).
- Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., & Sun, J. (2019). Detnas: Backbone search for object detection. *Advances in neural information processing systems 32* (pp. 6642–6652). San Jose: Curran Associates Inc.
- Chen, Y. C., Gao, C., Robb, E., & Huang, J. B. (2020c). NAS-DIP: Learning deep image prior with neural architecture search. In *European conference on computer vision (ECCV)*.
- Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., & Chen, L. C. (2020a). Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Li, H., Drummond, T., & Ge, Z. (2020b). Hierarchical neural architecture search for deep stereo matching. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 22158–22169). Curran Associates Inc.
- Chrabaszcz, P., Loshchilov, I., & Hutter, F. (2017). A downsampled variant of imagenet as an alternative to the CIFAR datasets. [CoRR. arXiv:1707.08819](https://arxiv.org/abs/1707.08819)
- Chu, X., Zhang, B., & Xu, R. (2021). Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12239–12248).
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.
- Ding, M., Lian, X., Yang, L., Wang, P., Jin, X., Lu, Z., & Luo, P. (2021). Hr-nas: Searching efficient high-resolution neural architectures with lightweight transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2982–2992).
- Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th international joint conference on artificial intelligence (IJCAI)*.
- Dong, X., & Yang, Y. (2019). Searching for a robust neural architecture in four GPU hours. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1761–1770).
- Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbaş, C., Golkov, V., vd Smagt, P., Cremers, D., & Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *IEEE international conference on computer vision (ICCV)*. <http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15>
- Du, X., Lin, T. Y., Jin, P., Ghiasi, G., Tan, M., Cui, Y., Le, Q. V., & Song, X. (2020). Spinenet: Learning scale-permuted backbone for recognition and localization. In *The IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Dudziak, L., Chau, T., Abdelfattah, M., Lee, R., Kim, H., & Lane, N. (2020). BRP-NAS: Prediction-based NAS using GCNs. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 10480–10490). Curran Associates Inc.
- Elsken, T., Metzen, J. H., & Hutter, F. (2017). Simple and efficient architecture search for convolutional neural networks. In *NeurIPS workshop on meta-learning*.
- Elsken, T., Metzen, J. H., & Hutter, F. (2019a). Efficient multi-objective neural architecture search via Lamarckian evolution. In *International conference on learning representations*.
- Elsken, T., Metzen, J. H., & Hutter, F. (2019b). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55), 1–21.
- Elsken, T., Staffler, B., Zela, A., Metzen, J. H., & Hutter, F. (2021). Bag of tricks for neural architecture search. In *The IEEE conference on computer vision and pattern recognition (CVPR)—Neural architecture search workshop*.
- Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1), 98–136.

- Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmaessan, Stockholm Sweden, proceedings of machine learning research* (Vol. 80, pp. 1436–1445).
- Fang, J., Sun, Y., Peng, K., Zhang, Q., Li, Y., Liu, W., & Wang, X. (2020). Fast neural network adaptation via parameter remapping and architecture search. In *International conference on learning representations*.
- Fourure, D., Emonet, R., Fromont, E., Muselet, D., Tremeau, A., & Wolf, C. (2017). Residual conv–deconv grid network for semantic segmentation. arXiv preprint [arXiv:1707.07958](https://arxiv.org/abs/1707.07958)
- Ghiasi, G., Lin, T. Y., & Le, Q. V. (2019). Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., & Keutzer, K. (2018). Squeezenext: Hardware-aware neural network design. In *The IEEE conference on computer vision and pattern recognition (CVPR) workshops*.
- Girshick, R. (2015). Fast R-CNN. In *2015 IEEE international conference on computer vision (ICCV)* (pp. 1440–1448).
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE conference on computer vision and pattern recognition* (pp. 580–587).
- Gong, X., Chen, W., Jiang, Y., Yuan, Y., Liu, X., Zhang, Q., Li, Y., & Wang, Z. (2020). Autopose: Searching multi-scale branch aggregation for pose estimation. arXiv preprint [arXiv:2008.07018](https://arxiv.org/abs/2008.07018)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 2672–2680). San Jose: Curran Associates Inc.
- Gou, Y., Li, B., Liu, Z., Yang, S., & Peng, X. (2020). Clearer: Multi-scale neural architecture search for image restoration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 17129–17140). San Jose: Curran Associates Inc.
- Gu, Y. C., Gao, S. H., Cao, X. S., Du, P., Lu, S. P., & Cheng, M. M. (2021) INAS: Integral NAS for device-aware salient object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4934–4944).
- Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X., & Xu, C. (2020a). Hit-detector: Hierarchical trinity architecture search for object detection. In *The IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., & Sun, J. (2020b). Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision* (pp. 544–560). Springer.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1904–1916.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778).
- Hinton, G., Deng, L., Yu, D., Dahl, G., Rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, 82–97.
- Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the knowledge in a neural network*. arXiv:1503.02531
- Ho, K., Gilbert, A., Jin, H., & Collomosse, J. (2020). *Neural architecture search for deep image prior*. arXiv:2001.04776
- Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for mobilenetv3. In *The IEEE international conference on computer vision* (ICCV).
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications*. In arXiv:1704.04861 [cs]
- Huang, Y., & Chen, Y. (2020). *Autonomous driving with deep learning: A survey of state-of-art technologies*. arXiv:2006.06091
- Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size*. arXiv:1602.07360 [cs]
- Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*. <http://lmb.informatik.uni-freiburg.de/Publications/2017/IMSKDB17>
- Jiang, C., Xu, H., Zhang, W., Liang, X., & Li, Z. (2020). SP-NAS: Serial-to-parallel backbone search for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., & Xing, E. P. (2018). Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in neural information processing systems* (Vol. 31).
- Klein, A., Falkner, S., Springenberg, J. T., & Hutter, F. (2017). Learning curve prediction with Bayesian neural networks. In *International conference on learning representations*.
- Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Tech. rep.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, (Vol. 25, pp. 1097–1105). Curran Associates, Inc.
- Lee, N., Ajanthan, T., & Torr, P. (2019). SNIP: Single-shot network pruning based on connection sensitivity. In *international conference on learning representations*. <https://openreview.net/forum?id=B1VZqjAcYX>
- Li, L., & Talwalkar, A. (2019). Random search and reproducibility for neural architecture search. In *Conference on uncertainty in artificial intelligence (UAI)*.
- Li, Y., Song, L., Chen, Y., Li, Z., Zhang, X., Wang, X., & Sun, J. (2020). Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Li, Z., Xi, T., Zhang, G., Liu, J., & He, R. (2021). Autodet: Pyramid network architecture search for object detection. *International Journal of Computer Vision*, 129(4), 1087–1105.
- Liang, T., Wang, Y., Tang, Z., Hu, G., & Ling, H. (2021). Opanas: One-shot path aggregation network architecture search for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10195–10203).
- Lin, P., Sun, P., Cheng, G., Xie, S., Li, X., & Shi, J. (2020). Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4203–4212).
- Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.
- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision—ECCV 2014* (pp. 740–755). Cham: Springer.

- Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A., van Ginneken, B., & Sanchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 60–88.
- Liu, C., Chen, L. C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., & Fei-Fei, L. (2019a). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Liu, H., Simonyan, K., Vinyals, O., Fernando, C., & Kavukcuoglu, K. (2018a). Hierarchical representations for efficient architecture search. In *international conference on learning representations (ICLR)*.
- Liu, H., Simonyan, K., & Yang, Y. (2019b). DARTS: Differentiable architecture search. In *International conference on learning representations*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2), 261–318. <https://doi.org/10.1007/s11263-019-01247-4>
- Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018b). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision—ECCV 2016* (pp. 21–37). Cham: Springer.
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3431–3440).
- Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., & Banzhaf, W. (2019). Nsga-net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference, association for computing machinery, New York, NY, USA, GECCO '19* (pp. 419–427).
- Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient CNN architecture design. In *The European conference on computer vision (ECCV)*.
- Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4040–4048).
- Mellor, J., Turner, J., Storkey, A., & Crowley, E. J. (2021). Neural architecture search without training. In *International conference on machine learning*.
- Mendoza, H., Klein, A., Feurer, M., Springenberg, J., & Hutter, F. (2016). Towards automatically-tuned neural networks. In *International conference on machine learning, AutoML workshop*.
- Mohan, R., & Valada, A. (2020). *Efficientps: Efficient panoptic segmentation*. arXiv preprint [arXiv:2004.02307](https://arxiv.org/abs/2004.02307)
- Nekrasov, V., Chen, H., Shen, C., & Reid, I. (2019). Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Nekrasov, V., Chen, H., Shen, C., & Reid, I. (2020). Architecture search of dynamic cells for semantic video segmentation. In *The IEEE Winter Conference on Applications of Computer Vision* (pp. 1970–1979).
- Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision—ECCV 2016* (pp. 483–499). Cham: Springer.
- Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *2015 IEEE international conference on computer vision (ICCV)* (pp. 1520–1528). <https://doi.org/10.1109/ICCV.2015.178>
- Oh, C., Tomczak, J., Gavves, E., & Welling, M. (2019). Combinatorial Bayesian optimization using the graph cartesian product. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 2914–2924). San Jose: Curran Associates Inc.
- Peng, H., Du, H., Yu, H., Li, Q., Liao, J., & Fu, J. (2020). Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. In *Advances in neural information processing systems* (Vol. 33, pp. 17955–17964).
- Peng, J., Sun, M., Zhang, Z. X., Tan, T., Yan, J. (2019). Efficient neural architecture transformation search in channel-level for object detection. In *Advances in neural information processing systems 32* (pp. 14313–14322). Curran Associates Inc. <http://papers.nips.cc/paper/9576-efficient-neural-architecture-transformation-search-in-channel-level-for-object-detection.pdf>
- Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient neural architecture search via parameter sharing. In *International conference on machine learning*.
- Pourchot, A., Ducarouge, A., & Sigaud, O. (2020). *To share or not to share: A comprehensive appraisal of weight-sharing*. arXiv preprint [arXiv:2002.04289](https://arxiv.org/abs/2002.04289)
- Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Aging evolution for image classifier architecture search. In *AAAI*.
- Real, E., Liang, C., So, D. R., & Le, Q. V. (2020). Evolving machine learning algorithms from scratch. In *International conference on machine learning*.
- Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., & Kurakin, A. (2017). Large-scale evolution of image classifiers. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning, PMLR, international convention centre, Sydney, Australia, proceedings of machine learning research* (Vol. 70, pp. 2902–2911).
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 779–788).
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 91–99). San Jose: Curran Associates Inc.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical image computing and computer-assisted intervention—MICCAI 2015* (pp. 234–241). Cham: Springer.
- Ru, B., Lyle, C., Schut, L., van der Wilk, M., & Gal, Y. (2021a). *Revisiting the train loss: An efficient performance estimator for neural architecture search*. <https://openreview.net/forum?id=XvOH0v2hsp>
- Ru, B., Wan, X., Dong, X., & Osborne, M. (2021b). Interpretable neural architecture search via Bayesian optimisation with Weisfeiler–Lehman kernels. In *International conference on learning representations*. <https://openreview.net/forum?id=j9Rv7qdXjd>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- Saikia, T., Marrakchi, Y., Zela, A., Hutter, F., & Brox, T. (2019). Autodispnet: Improving disparity estimation with automl. In *The IEEE international conference on computer vision (ICCV)*.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

- Saxena, S., & Verbeek, J. (2016). Convolutional neural fabrics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29, pp. 4053–4061). San Jose: Curran Associates Inc.
- Shaw, A., Hunter, D., Landola, F., & Sidhu, S. (2019). Squeezenas: Fast neural architecture search for faster semantic segmentation. In *The IEEE international conference on computer vision (ICCV) workshops*.
- Siems, J., Zimmer, L., Zela, A., Lukasik, J., Keuper, M., & Hutter, F. (2020). Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. In *NeurIPS 4th workshop on meta-learning*.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529, 484–503.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Y. Bengio & Y. LeCun (Eds.), *3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings*.
- Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10, 99–127.
- Sun, D., Yang, X., Liu, M. Y., & Kautz, J. (2018). Pwc-net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.
- Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., & Osborne, M. (2013). Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. In *NIPS workshop on bayesian optimization in theory and practice*.
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the thirty-first AAAI conference on artificial intelligence, AAAI'17*. AAAI Press.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on computer vision and pattern recognition (CVPR)* (pp. 2818–2826).
- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Tan, M., & Le, Q. (1907). Mixconv: Mixed depthwise convolutional kernels. [arXiv:1907.09595](https://arxiv.org/abs/1907.09595)
- Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, PMLR, Long Beach, California, USA, proceedings of machine learning research* (Vol. 97, pp. 6105–6114).
- Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781–10790).
- Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. In A. Vedaldi, H. Bischof, T. Brox, & J. M. Frahm (Eds.), *Computer vision—ECCV 2020* (pp. 685–702). Cham: Springer.
- Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.
- Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020a). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10), 3349–3364.
- Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C., & Zhang, Y. (2020b). Nas-fcos: Fast neural architecture search for object detection. In *The IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Wang, R., Cheng, M., Chen, X., Tang, X., & Hsieh, C. J. (2021). Rethinking architecture selection in differentiable NAS. In *International conference on learning representations*.
- Wei, T., Wang, C., Rui, Y., & Chen, C. W. (2016). Network morphism. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning, PMLR, New York, New York, USA, proceedings of machine learning research* (Vol. 48, pp. 564–572).
- Wen, W., Liu, H., Li, H., Chen, Y., Bender, G., & Kindermans, P. J. (2020). Neural predictor for neural architecture search. In *ECCV*.
- Weng, Y., Zhou, T., Li, Y., & Qiu, X. (2019). Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7, 44247–44257.
- White, C., Neiswanger, W., & Savani, Y. (2019). *Bananas: Bayesian optimization with neural architectures for neural architecture search*. arXiv preprint [arXiv:1910.11858](https://arxiv.org/abs/1910.11858)
- White, C., Safari, M., Sukthanker, R., Ru, B., Elsen, T., Zela, A., Dey, D., & Hutter, F. (2023). Neural architecture search: Insights from 1000 papers. *CoRR*. <https://doi.org/10.48550/ARXIV.2301.08727>
- White, C., Zela, A., Ru, B., Liu, Y., & Hutter, F. (2021). *How powerful are performance predictors in neural architecture search?* arXiv preprint
- Wistuba, M., Rawat, A., & Pedapati, T. (2019). *A survey on neural architecture search*. arXiv preprint [arXiv:1905.01392](https://arxiv.org/abs/1905.01392)
- Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., & Keutzer, K. (2019a). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Wu, H., Zhang, J., & Huang, K. (2019b). Sparsemask: Differentiable connectivity learning for dense image prediction. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*.
- Wu, Y., Zhang, G., Xu, H., Liang, X., & Lin, L. (2020). Auto-panoptic: Cooperative multi-component architecture search for panoptic segmentation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 20508–20519). San Jose: Curran Associates Inc.
- Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., Zhang, X., et al. (2021). Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing Surveys (CSUR)*, 54(9), 1–37.
- Xie, S., Zheng, H., Liu, C., & Lin, L. (2019). SNAS: Stochastic neural architecture search. In *International conference on learning representations*.
- Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Kindermans, P. J., Tan, M., Singh, V., & Chen, B. (2020). *Mobiledets: Searching for object detection architectures for mobile accelerators*. arXiv preprint.
- Xu, H., Yao, L., Zhang, W., Liang, X., & Li, Z. (2019a). Auto-FPN: Automatic network architecture adaptation for object detection beyond classification. In *The IEEE international conference on computer vision (ICCV)*.
- Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G. J., Tian, Q., & Xiong, H. (2019b). *Pc-darts: Partial channel connections for memory-efficient architecture search*. arXiv preprint [arXiv:1907.05737](https://arxiv.org/abs/1907.05737)
- Yang, A., Esperança, P. M., & Carlucci, F. M. (2020). Nas evaluation is frustratingly hard. In *International conference on learning representations*. <https://openreview.net/forum?id=HygrdpVKvr>

- Yao, L., Pi, R., Xu, H., Zhang, W., Li, Z., & Zhang, T. (2021). Joint-detnas: upgrade your detector with NAS, pruning and dynamic distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10175–10184).
- Yao, L., Xu, H., Zhang, W., Liang, X., & Li, Z. (2020). SM-NAS: structural-to-modular neural architecture search for object detection. In *The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020* (pp. 12661–12668). AAAI Press. <https://aaai.org/ojs/index.php/AAAI/article/view/6958>
- Yu, K., Ranftl, R., & Salzmann, M. (2020a). *How to train your super-net: An analysis of training heuristics in weight-sharing NAS*. arXiv preprint [arXiv:2003.04276](https://arxiv.org/abs/2003.04276)
- Yu, K., Scuto, C., Jaggi, M., Musat, C., & Salzmann, M. (2020b). Evaluating the search phase of neural architecture search. In *International conference on learning representations*, <https://openreview.net/forum?id=H1loF2NFwr>
- Yu, Q., Yang, D., Roth, H., Bai, Y., Zhang, Y., Yuille, A. L., & Xu, D. (2020c). C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., & Hutter, F. (2020a). Understanding and robustifying differentiable architecture search. In *International conference on learning representations*. <https://openreview.net/forum?id=H1gDNyrKDS>
- Zela, A., Klein, A., Falkner, S., & Hutter, F. (2018). Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. In *ICML 2018 workshop on AutoML (AutoML 2018)*.
- Zela, A., Siems, J., & Hutter, F. (2020b). Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International conference on learning representations*. <https://openreview.net/forum?id=SJx9ngStPH>
- Zhang, X., Xu, H., Mo, H., Tan, J., Yang, C., Wang, L., & Ren, W. (2021). DCNAs: Densely connected neural architecture search for semantic image segmentation. [arXiv:2003.11883](https://arxiv.org/abs/2003.11883)
- Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE conference on computer vision and pattern recognition (CVPR)*
- Zhang, Y., Lin, Z., Jiang, J., Zhang, Q., Wang, Y., Xue, H., Zhang, C., & Yang, Y. (2020). *Deeper insights into weight sharing in neural architecture search*. arXiv preprint [arXiv:2001.01431](https://arxiv.org/abs/2001.01431)
- Zhang, Y., Qiu, Z., Liu, J., Yao, T., Liu, D., & Mei, T. (2019). Customizable architecture search for semantic segmentation. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.
- Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 6230–6239). <https://doi.org/10.1109/CVPR.2017.660>
- Zhong, Y., Deng, Z., Guo, S., Scott, M. R., & Huang, W. (2020a). Representation sharing for fast object detector search and beyond. In *Proceedings of the European conference on computer vision (ECCV)*.
- Zhong, Z., Lin, Z. Q., Bidart, R., Hu, X., Daya, I. B., Li, Z., Zheng, W. S., Li, J., & Wong, A. (2020b). Squeeze-and-attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Zhong, Z., Yan, J., Wu, W., Shao, J., & Liu, C. L. (2018). Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2423–2432).
- Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A. (2016). Semantic understanding of scenes through the ade20k dataset. arXiv preprint [arXiv:1608.05442](https://arxiv.org/abs/1608.05442)
- Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Zhou, D., Zhou, X., Zhang, W., Loy, C. C., Yi, S., Zhang, X., & Ouyang, W. (2020). Econas: Finding proxies for economical neural architecture search. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Zhu, Z., Liu, C., Yang, D., Yuille, A., & Xu, D. (2019). V-nas: Neural architecture search for volumetric medical image segmentation. In *2019 International conference on 3D vision (3DV)* (pp. 240–248).
- Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *International conference on learning representations*.
- Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Conference on computer vision and pattern recognition*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.