Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation

Chenxi Liu[1]*, Liang-Chieh Chen[2], Florian Schroff[2], Hartwig Adam[2], Wei Hua[2], Alan Yuille[1], Li Fei-Fei[3]
[1]Johns Hopkins University [2]Google [3]Stanford University

Abstract

Recently, Neural Architecture Search (NAS) has successfully identified neural network architectures that exceed human designed ones on large-scale image classification. In this paper, we study NAS for semantic image segmentation. Existing works often focus on searching the repeatable cell structure, while hand-designing the outer network structure that controls the spatial resolution changes. This choice simplifies the search space, but becomes increasingly problematic for dense image prediction which exhibits a lot more network level architectural variations. Therefore, we propose to search the network level structure in addition to the cell level structure, which forms a hierarchical architecture search space. We present a network level search space that includes many popular designs, and develop a formulation that allows efficient gradient-based architecture search (3 P100 GPU days on Cityscapes images). We demonstrate the effectiveness of the proposed method on the challenging Cityscapes, PASCAL VOC 2012, and ADE20K datasets. Auto-DeepLab, our architecture searched specifically for semantic image segmentation, attains state-of-the-art performance without any ImageNet pretraining.[1]

1. Introduction

Deep neural networks have been proved successful across a large variety of artificial intelligence tasks, including image recognition [38, 25], speech recognition [27], machine translation [73, 81] *etc.* While better optimizers [36] and better normalization techniques [32, 80] certainly played an important role, a lot of the progress comes from the design of neural network architectures. In computer vision, this holds true for both image classification [38, 72, 75, 76, 74, 25, 85, 31, 30] and dense image prediction [16, 51, 7, 64, 56, 55].

More recently, in the spirit of AutoML and democratizing AI, there has been significant interest in designing neural network architectures *automatically*, instead of relying heavily on expert experience and knowledge. Importantly, in the past year, Neural Architecture Search (NAS) has successfully identified architectures that exceed human-designed architectures on large-scale image classification problems [93, 47, 62].

Model	Auto Search Cell	Network	Dataset	Days	Task
ResNet [25]	✗	✗	-	-	Cls
DenseNet [31]	✗	✗	-	-	Cls
DeepLabv3+ [11]	✗	✗	-	-	Seg
NASNet [93]	✓	✗	CIFAR-10	2000	Cls
AmoebaNet [62]	✓	✗	CIFAR-10	2000	Cls
PNASNet [47]	✓	✗	CIFAR-10	150	Cls
DARTS [49]	✓	✗	CIFAR-10	4	Cls
DPC [6]	✓	✗	Cityscapes	2600	Seg
Auto-DeepLab	✓	✓	Cityscapes	3	Seg

Table 1: Comparing our work against other CNN architectures with two-level hierarchy. The main differences include: (1) we directly search CNN architecture for semantic segmentation, (2) we search the network level architecture as well as the cell level one, and (3) our efficient search only requires 3 P100 GPU days.

Image classification is a good starting point for NAS, because it is the most fundamental and well-studied high-level recognition task. In addition, there exists benchmark datasets (*e.g.*, CIFAR-10) with relatively small images, resulting in less computation and faster training. However, image classification should not be the end point for NAS, and the current success shows promise to extend into more demanding domains. In this paper, we study Neural Architecture Search for semantic image segmentation, an important computer vision task that assigns a label like "person" or "bicycle" to each pixel in the input image.

Naively porting ideas from image classification would not suffice for semantic segmentation. In image classification, NAS typically applies transfer learning from low res-

*Work done while an intern at Google.
[1]Code for Auto-DeepLab released at https://github.com/tensorflow/models/tree/master/research/deeplab.

olution images to high resolution images [93], whereas optimal architectures for semantic segmentation must inherently operate on high resolution imagery. This suggests the need for: (1) a more relaxed and general search space to capture the architectural variations brought by the higher resolution, and (2) a more efficient architecture search technique as higher resolution requires heavier computation.

We notice that modern CNN designs [25, 85, 31] usually follow a two-level hierarchy, where the outer network level controls the spatial resolution changes, and the inner cell level governs the specific layer-wise computations. The vast majority of current works on NAS [93, 47, 62, 59, 49] follow this two-level hierarchical design, but only automatically search the inner cell level while hand-designing the outer network level. This limited search space becomes problematic for dense image prediction, which is sensitive to the spatial resolution changes. Therefore in our work, we propose a trellis-like network level search space that augments the commonly-used cell level search space first proposed in [93] to form a *hierarchical* architecture search space. Our goal is to jointly learn a good combination of repeatable cell structure and network structure specifically for semantic image segmentation.

In terms of the architecture search method, reinforcement learning [92, 93] and evolutionary algorithms [63, 62] tend to be computationally intensive even on the low resolution CIFAR-10 dataset, therefore probably not suitable for semantic image segmentation. We draw inspiration from the differentiable formulation of NAS [69, 49], and develop a continuous relaxation of the discrete architectures that exactly matches the hierarchical architecture search space. The hierarchical architecture search is conducted via stochastic gradient descent. When the search terminates, the best cell architecture is decoded greedily, and the best network architecture is decoded efficiently using the Viterbi algorithm. We directly search architecture on 321×321 image crops from Cityscapes [13]. The search is very efficient and only takes about 3 days on one P100 GPU.

We report experimental results on multiple semantic segmentation benchmarks, including Cityscapes [13], PASCAL VOC 2012 [15], and ADE20K [90]. Without ImageNet [65] pretraining, our best model significantly outperforms FRRN-B [60] by 8.6% and GridNet [17] by 10.9% on Cityscapes test set, and performs comparably with other ImageNet-pretrained state-of-the-art models [82, 88, 4, 11, 6] when also exploiting the coarse annotations on Cityscapes. Notably, our best model (without pretraining) attains the same performance as DeepLabv3+ [11] (with pretraining) while being 2.23 times faster in Multi-Adds. Additionally, our light-weight model attains the performance only 1.2% lower than DeepLabv3+ [11], while requiring 76.7% fewer parameters and being 4.65 times faster in Multi-Adds. Finally, on PASCAL VOC 2012 and ADE20K, our best model outperforms several state-of-the-art models [90, 44, 82, 88, 83] while using strictly less data for pretraining.

To summarize, the contribution of our paper is four-fold:

- Ours is one of the first attempts to extend NAS beyond image classification to dense image prediction.
- We propose a network level architecture search space that augments and complements the much-studied cell level one, and consider the more challenging joint search of network level and cell level architectures.
- We develop a differentiable, continuous formulation that conducts the two-level hierarchical architecture search efficiently in 3 GPU days.
- Without ImageNet pretraining, our model significantly outperforms FRRN-B and GridNet, and attains comparable performance with other ImageNet-pretrained state-of-the-art models on Cityscapes. On PASCAL VOC 2012 and ADE20K, our best model also outperforms several state-of-the-art models.

2. Related Work

Semantic Image Segmentation Convolutional neural networks [42] deployed in a fully convolutional manner (FCNs [68, 51]) have achieved remarkable performance on several semantic segmentation benchmarks. Within the state-of-the-art systems, there are two essential components: multi-scale context module and neural network design. It has been known that context information is crucial for pixel labeling tasks [26, 70, 37, 39, 16, 54, 14, 10]. Therefore, PSPNet [88] performs spatial pyramid pooling [21, 41, 24] at several grid scales (including image-level pooling [50]), while DeepLab [8, 9] applies several parallel atrous convolution [28, 20, 68, 57, 7] with different rates. On the other hand, the improvement of neural network design has significantly driven the performance from AlexNet [38], VGG [72], Inception [32, 76, 74], ResNet [25] to more recent architectures, such as Wide ResNet [86], ResNeXt [85], DenseNet [31] and Xception [12, 61]. In addition to adopting those networks as backbones for semantic segmentation, one could employ the encoder-decoder structures [64, 2, 55, 44, 60, 58, 33, 79, 18, 11, 87, 83] which efficiently captures the long-range context information while keeping the detailed object boundaries. Nevertheless, most of the models require initialization from the ImageNet [65] pretrained checkpoints except FRRN [60] and GridNet [17] for the task of semantic segmentation. Specifically, FRRN [60] employs a two-stream system, where full-resolution information is carried in one stream and context information in the other pooling stream. GridNet, building on top of a similar idea, contains multiple streams with different resolutions. In this work, we apply neural architecture search for network backbones specific for semantic segmentation. We further show state-of-the-art performance without ImageNet pretraining, and significantly outperforms FRRN [60] and GridNet [17] on Cityscapes [13].

Neural Architecture Search Method Neural Architecture Search aims at automatically designing neural network architectures, hence minimizing human hours and efforts. While some works [22, 34, 92, 49] search RNN cells for language tasks, more works search good CNN architectures for image classification.

Several papers used reinforcement learning (either policy gradients [92, 93, 5, 77] or Q-learning [3, 89]) to train a recurrent neural network that represents a policy to generate a sequence of symbols specifying the CNN architecture. An alternative to RL is to use evolutionary algorithms (EA), that "evolves" architectures by mutating the best architectures found so far [63, 84, 53, 48, 62]. However, these RL and EA methods tend to require massive computation during the search, usually thousands of GPU days. PNAS [47] proposed a progressive search strategy that markedly reduced the search cost while maintaining the quality of the searched architecture. NAO [52] embedded architectures into a latent space and performed optimization before decoding. Additionally, several works [59, 49, 1] utilized architectural sharing among sampled models instead of training each of them individually, thereby further reduced the search cost. Our work follows the differentiable NAS formulation [69, 49] and extends it into the more general hierarchical setting.

Neural Architecture Search Space Earlier papers, *e.g.*, [92, 63], tried to directly construct the entire network. However, more recent papers [93, 47, 62, 59, 49] have shifted to searching the repeatable cell structure, while keeping the outer network level structure fixed by hand. First proposed in [93], this strategy is likely inspired by the two-level hierarchy commonly used in modern CNNs.

Our work still uses this cell level search space to keep consistent with previous works. Yet one of our contributions is to propose a new, general-purpose network level search space, since we wish to jointly search across this two-level hierarchy. Our network level search space shares a similar outlook as [67], but the important difference is that [67] kept the entire "fabrics" with no intention to alter the architecture, whereas we associate an explicit weight for each connection and focus on decoding a *single* discrete structure. In addition, [67] was evaluated on segmenting face images into 3 classes [35], whereas our models are evaluated on large-scale segmentation datasets such as Cityscapes [13], PASCAL VOC 2012 [15], and ADE20K [90].

The most similar work to ours is [6], which also studied NAS for semantic image segmentation. However, [6] focused on searching the much smaller Atrous Spatial Pyramid Pooling (ASPP) module using random search, whereas we focus on searching the much more fundamental network backbone architecture using more advanced and more efficient search methods.

3. Architecture Search Space

This section describes our two-level hierarchical architecture search space. For the inner cell level (Sec. 3.1), we reuse the one adopted in [93, 47, 62, 49] to keep consistent with previous works. For the outer network level (Sec. 3.2), we propose a novel search space based on observation and summarization of many popular designs.

3.1. Cell Level Search Space

We define a *cell* to be a small fully convolutional module, typically repeated multiple times to form the entire neural network. More specifically, a cell is a directed acyclic graph consisting of B blocks.

Each *block* is a two-branch structure, mapping from 2 input tensors to 1 output tensor. Block i in cell l may be specified using a 5-tuple (I_1, I_2, O_1, O_2, C), where $I_1, I_2 \in \mathcal{I}_i^l$ are selections of input tensors, $O_1, O_2 \in \mathcal{O}$ are selections of layer types applied to the corresponding input tensor, and $C \in \mathcal{C}$ is the method used to combine the individual outputs of the two branches to form this block's output tensor, H_i^l. The cell's output tensor H^l is simply the concatenation of the blocks' output tensors $H_1^l, \ldots, H_B^l$ in this order.

The set of possible input tensors, $\mathcal{I}_i^l$, consists of the output of the previous cell H^{l-1}, the output of the previous-previous cell H^{l-2}, and previous blocks' output in the current cell $\{H_1^l, \ldots, H_i^l\}$. Therefore, as we add more blocks in the cell, the next block has more choices as potential source of input.

The set of possible layer types, $\mathcal{O}$, consists of the following 8 operators, all prevalent in modern CNNs:

- 3×3 depthwise-separable conv
- 5×5 depthwise-separable conv
- 3×3 atrous conv with rate 2
- 5×5 atrous conv with rate 2
- 3×3 average pooling
- 3×3 max pooling
- skip connection
- no connection (zero)

For the set of possible combination operators $\mathcal{C}$, we simply let element-wise addition to be the only choice.

3.2. Network Level Search Space

In the image classification NAS framework pioneered by [93], once a cell structure is found, the entire network is constructed using a pre-defined pattern. Therefore the network level was not part of the architecture search, hence its search space has never been proposed nor designed.

This pre-defined pattern is simple and straightforward: a number of "normal cells" (cells that keep the spatial resolution of the feature tensor) are separated equally by inserting "reduction cells" (cells that divide the spatial resolution

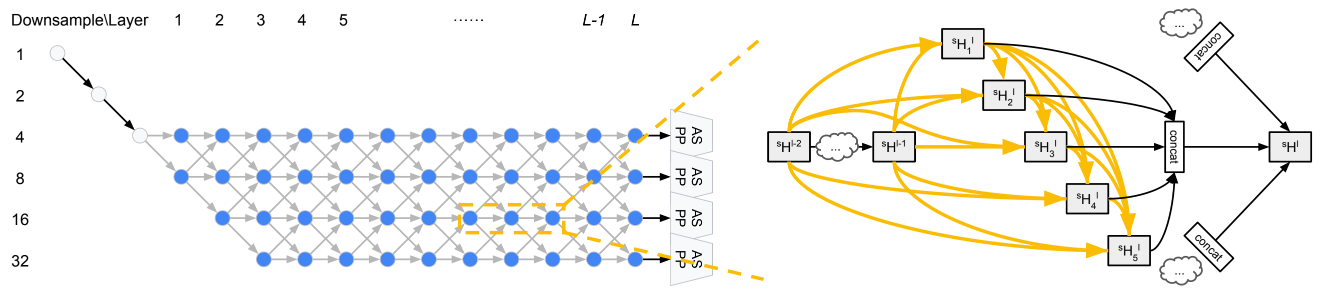
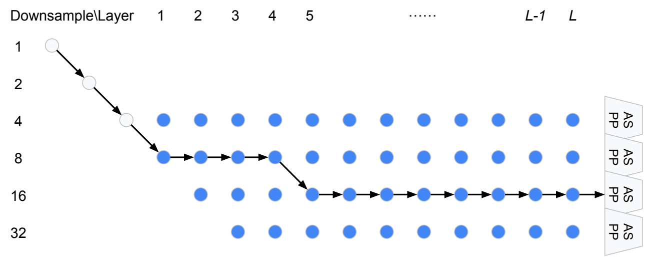
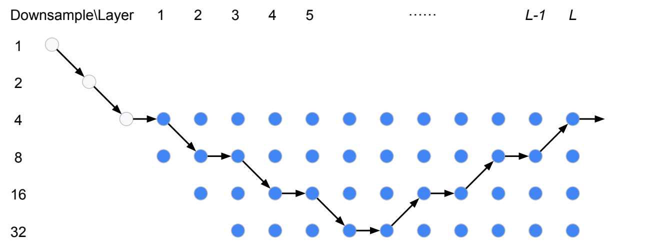


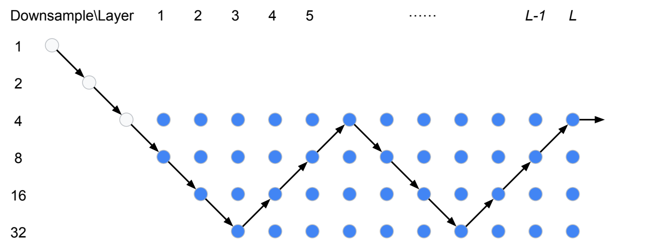
Figure 1: *Left:* Our network level search space with $L = 12$. Gray nodes represent the fixed "stem" layers, and a path along the blue nodes represents a candidate network level architecture. *Right:* During the search, each cell is a densely connected structure as described in Sec. 4.1.1. Every yellow arrow is associated with the set of values $\alpha_{j\to i}$. The three arrows after `concat` are associated with $\beta^l_{\frac{s}{2}\to s}, \beta^l_{s\to s}, \beta^l_{2s\to s}$ respectively, as described in Sec. 4.1.2. Best viewed in color.



(a) Network level architecture used in DeepLabv3 [9].



(b) Network level architecture used in Conv-Deconv [56].



(c) Network level architecture used in Stacked Hourglass [55].

Figure 2: Our network level search space is general and includes various existing designs.

by 2 and multiply the number of filters by 2). This keep-downsampling strategy is reasonable in the image classification case, but in dense image prediction it is also important to keep high spatial resolution, and as a result there are more network level variations [9, 56, 55].

Among the various network architectures for dense image prediction, we notice two principles that are consistent:

- The spatial resolution of the next layer is either twice as large, or twice as small, or remains the same.
- The smallest spatial resolution is downsampled by 32.

Following these common practices, we propose the following network level search space. The beginning of the network is a two-layer "stem" structure that each reduces the spatial resolution by a factor of 2. After that, there are a total of L layers with unknown spatial resolutions, with the maximum being downsampled by 4 and the minimum being downsampled by 32. Since each layer may differ in spatial resolution by at most 2, the first layer after the stem could only be either downsampled by 4 or 8. We illustrate our network level search space in Fig. 1. Our goal is then to find a good path in this L-layer trellis.

In Fig. 2 we show that our search space is general enough to cover many popular designs. In the future, we have plans to relax this search space even further to include U-net architectures [64, 45, 71], where layer l may receive input from one more layer preceding l in addition to $l-1$.

We reiterate that our work searches the network level architecture *in addition to* the cell level architecture. Therefore our search space is strictly more challenging and general-purpose than previous works.

4. Methods

We begin by introducing a continuous relaxation of the (exponentially many) discrete architectures that exactly matches the hierarchical architecture search described above. We then discuss how to perform architecture search via optimization, and how to decode back a discrete architecture after the search terminates.

4.1. Continuous Relaxation of Architectures

4.1.1 Cell Architecture

We reuse the continuous relaxation described in [49]. Every block's output tensor H_i^l is connected to all hidden states in $\mathcal{I}_i^l$:

$$H_i^l = \sum_{H_j^l \in \mathcal{I}_i^l} O_{j\to i}(H_j^l) \tag{1}$$

In addition, we approximate each $O_{j\to i}$ with its continuous relaxation $\bar{O}_{j\to i}$, defined as:

$$\bar{O}_{j\to i}(H_j^l) = \sum_{O^k \in \mathcal{O}} \alpha_{j\to i}^k O^k(H_j^l) \tag{2}$$

where

$$\sum_{k=1}^{|\mathcal{O}|} \alpha_{j\to i}^k = 1 \qquad \forall i, j \tag{3}$$

$$\alpha_{j\to i}^k \geq 0 \qquad \forall i, j, k \tag{4}$$

In other words, $\alpha_{j\to i}^k$ are normalized scalars associated with each operator $O^k \in \mathcal{O}$, easily implemented as softmax.

Recall from Sec. 3.1 that H^{l-1} and H^{l-2} are always included in $\mathcal{I}_i^l$, and that H^l is the concatenation of $H_1^l, \ldots, H_B^l$. Together with Eq. (1) and Eq. (2), the cell level update may be summarized as:

$$H^l = \text{Cell}(H^{l-1}, H^{l-2}; \alpha) \tag{5}$$

4.1.2 Network Architecture

Within a cell, all tensors are of the same spatial size, which enables the (weighted) sum in Eq. (1) and Eq. (2). However, as clearly illustrated in Fig. 1, tensors may take different sizes in the network level. Therefore in order to set up the continuous relaxation, each layer l will have at most 4 hidden states $\{{}^4H^l, {}^8H^l, {}^{16}H^l, {}^{32}H^l\}$, with the upper left superscript indicating the spatial resolution.

We design the network level continuous relaxation to exactly match the search space described in Sec. 3.2. We associated a scalar with each gray arrow in Fig. 1, and the network level update is:

$$\begin{aligned} {}^sH^l = &\beta_{\frac{s}{2}\to s}^l \text{Cell}({}^{\frac{s}{2}}H^{l-1}, {}^sH^{l-2}; \alpha) \\ &+ \beta_{s\to s}^l \text{Cell}({}^sH^{l-1}, {}^sH^{l-2}; \alpha) \\ &+ \beta_{2s\to s}^l \text{Cell}({}^{2s}H^{l-1}, {}^sH^{l-2}; \alpha) \end{aligned} \tag{6}$$

where $s = 4, 8, 16, 32$ and $l = 1, 2, \ldots, L$. The scalars β are normalized such that

$$\beta_{s\to\frac{s}{2}}^l + \beta_{s\to s}^l + \beta_{s\to 2s}^l = 1 \qquad \forall s, l \tag{7}$$

$$\beta_{s\to\frac{s}{2}}^l \geq 0 \quad \beta_{s\to s}^l \geq 0 \quad \beta_{s\to 2s}^l \geq 0 \qquad \forall s, l \tag{8}$$

also implemented as softmax.

Eq. (6) shows how the continuous relaxations of the two-level hierarchy are weaved together. In particular, β controls the outer network level, hence depends on the spatial size and layer index. Each scalar in β governs an entire set of α, yet α specifies the same architecture that depends on neither spatial size nor layer index.

As illustrated in Fig. 1, Atrous Spatial Pyramid Pooling (ASPP) modules are attached to each spatial resolution at the L-th layer (atrous rates are adjusted accordingly). Their outputs are bilinear upsampled to the original resolution before summed to produce the prediction.

4.2. Optimization

The advantage of introducing this continuous relaxation is that the scalars controlling the connection strength between different hidden states are now part of the differentiable computation graph. Therefore they can be optimized efficiently using gradient descent. We adopt the first-order approximation in [49], and partition the training data into two disjoint sets *trainA* and *trainB*. The optimization alternates between:

1. Update network weights w by $\nabla_w \mathcal{L}_{trainA}(w, \alpha, \beta)$
2. Update architecture α, β by $\nabla_{\alpha,\beta} \mathcal{L}_{trainB}(w, \alpha, \beta)$

where the loss function $\mathcal{L}$ is the cross entropy calculated on the semantic segmentation mini-batch. The disjoint set partition is to prevent the architecture from overfitting the training data.

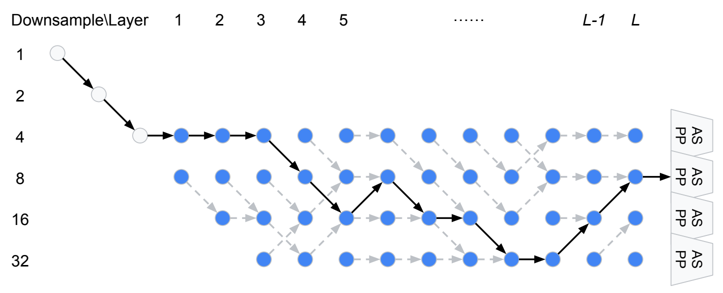
4.3. Decoding Discrete Architectures

Cell Architecture Following [49], we decode the discrete cell architecture by first retaining the 2 strongest predecessors for each block (with the strength from hidden state j to hidden state i being $\max_{k, O^k \neq zero} \alpha_{j\to i}^k$; recall from Sec. 3.1 that "zero" means "no connection"), and then choose the most likely operator by taking the argmax.

Network Architecture Eq. (7) essentially states that the "outgoing probability" at each of the blue nodes in Fig. 1 sums to 1. In fact, the β values can be interpreted as the "transition probability" between different "states" (spatial resolution) across different "time steps" (layer number). Quite intuitively, our goal is to find the path with the "maximum probability" from start to end. This path can be decoded efficiently using the classic Viterbi algorithm, as in our implementation.

5. Experimental Results

Herein, we report our architecture search implementation details as well as the search results. We then report semantic segmentation results on benchmark datasets with our best found architecture.



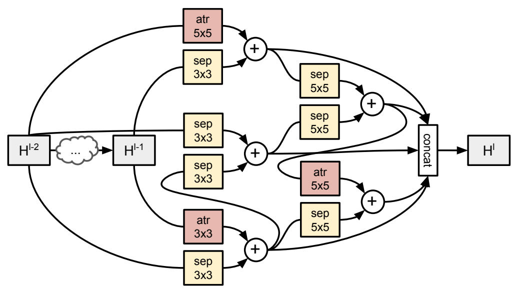


Figure 3: The Auto-DeepLab architecture found by our Hierarchical Neural Architecture Search on Cityscapes. Gray dashed arrows show the connection with maximum β at each node. **atr:** atrous convolution. **sep:** depthwise-separable convolution.

5.1. Architecture Search Implementation Details

We consider a total of $L = 12$ layers in the network, and $B = 5$ blocks in a cell. The network level search space has 2.9×10^4 unique paths, and the number of cell structures is 5.6×10^{14}. So the size of the joint, hierarchical search space is in the order of 10^{19}.

We follow the common practice of doubling the number of filters when halving the height and width of feature tensor. Every blue node in Fig. 1 with downsample rate s has $B \times F \times \frac{s}{4}$ output filters, where F is the filter multiplier controlling the model capacity. We set $F = 8$ during the architecture search. A stride 2 convolution is used for all $\frac{s}{2} \to s$ connections, both to reduce spatial size and double the number of filters. Bilinear upsampling followed by 1×1 convolution is used for all $2s \to s$ connections, both to increase spatial size and halve the number of filters.

The Atrous Spatial Pyramid Pooling module used in [9] has 5 branches: one 1×1 convolution, three 3×3 convolution with various atrous rates, and pooled image feature. During the search, we simplify ASPP to have 3 branches instead of 5 by only using one 3×3 convolution with atrous rate $\frac{96}{s}$. The number of filters produced by each ASPP branch is still $B \times F \times \frac{s}{4}$.

We conduct architecture search on the Cityscapes dataset [13] for semantic image segmentation. More specifically, we use 321×321 random image crops from half-resolution (512×1024) images in the *train_fine* set. We randomly select half of the images in *train_fine* as *trainA*, and the other half as *trainB* (see Sec. 4.2).

The architecture search optimization is conducted for a total of 40 epochs. The batch size is 2 due to GPU memory constraint. When learning network weights w, we use SGD optimizer with momentum 0.9, cosine learning rate that decays from 0.025 to 0.001, and weight decay 0.0003. The initial values of α, β before softmax are sampled from a standard Gaussian times 0.001. They are optimized using Adam optimizer [36] with learning rate 0.003 and weight decay 0.001. We empirically found that if α, β are optimized from the beginning when w are not well trained, the architecture tends to fall into bad local optima. Therefore we start optimizing α, β after 20 epochs. The entire architecture search optimization takes about 3 days on one P100 GPU. Fig. 4 shows that the validation accuracy steadily improves throughout this process. We also tried searching for longer epochs (60, 80, 100), but did not observe benefit.

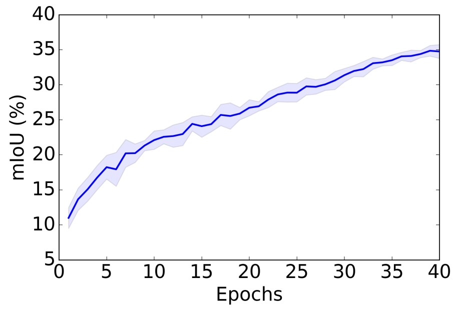


Figure 4: Validation accuracy during 40 epochs of architecture search optimization across 10 random trials.

Fig. 3 visualizes the best architecture found. In terms of network level architecture, higher resolution is preferred at both beginning (stays at downsample by 4 for longer) and end (ends at downsample by 8). We also show the strongest outgoing connection at each node using gray dashed arrows. We observe a general tendency to downsample in the first 3/4 layers and upsample in the last 1/4 layers. In terms of cell level architecture, the conjunction of atrous convolution and depthwise-separable convolution is often used, suggesting that the importance of context has been learned. Note that atrous convolution is rarely found to be useful in cells for image classification[2].

5.2. Semantic Segmentation Results

We evaluate the performance of our found best architecture (Fig. 3) on Cityscapes [13], PASCAL VOC 2012 [15], and ADE20K [90] datasets.

[2]Among NASNet-{A, B, C}, PNASNet-{1, 2, 3, 4, 5}, AmoebaNet-{A, B, C}, ENAS, DARTS, atrous convolution was used only once in AmoebaNet-B reduction cell.

Method	ImageNet	F	Multi-Adds	Params	mIOU (%)
Auto-DeepLab-S		20	333.25B	10.15M	79.74
Auto-DeepLab-M		32	460.93B	21.62M	80.04
Auto-DeepLab-L		48	695.03B	44.42M	80.33
FRRN-A [60]		-	-	17.76M	65.7
FRRN-B [60]		-	-	24.78M	-
DeepLabv3+ [11]	✓	-	1551.05B	43.48M	79.55

Table 2: Cityscapes validation set results with different Auto-DeepLab model variants. F: the filter multiplier controlling the model capacity. All our models are trained from *scratch* and with *single-scale* input during inference.

Method	itr-500K	itr-1M	itr-1.5M	**SDP**	mIOU (%)
Auto-DeepLab-S	✓				75.20
Auto-DeepLab-S		✓			77.09
Auto-DeepLab-S			✓		78.00
Auto-DeepLab-S			✓	✓	79.74

Table 3: Cityscapes validation set results. We experiment with the effect of adopting different training iterations (500K, 1M, and 1.5M iterations) and the Scheduled Drop Path method (**SDP**). All models are trained from scratch.

We follow the same training protocol in [9, 11]. In brief, during training we adopt a polynomial learning rate schedule [50] with initial learning rate 0.05, and large crop size (*e.g.*, 769 $\times$ 769 on Cityscapes, and 513 $\times$ 513 on PASCAL VOC 2012 and resized ADE20K images). Batch normalization parameters [32] are fine-tuned during training. The models are trained from scratch with 1.5M iterations on Cityscapes, 1.5M iterations on PASCAL VOC 2012, and 4M iterations on ADE20K, respectively.

We adopt the simple encoder-decoder structure similar to DeepLabv3+ [11]. Specifically, our encoder consists of our found best network architecture augmented with the ASPP module [8, 9], and our decoder is the same as the one in DeepLabv3+ which recovers the boundary information by exploiting the low-level features that have downsample rate 4. Additionally, we redesign the "stem" structure with three 3 $\times$ 3 convolutions (with stride 2 in the first and third convolutions). The first two convolutions have 64 filters while the third convolution has 128 filters. This "stem" has been shown to be effective for segmentation in [88, 78].

5.2.1 Cityscapes

Cityscapes [13] contains high quality pixel-level annotations of 5000 images with size 1024 $\times$ 2048 (2975, 500, and 1525 for the training, validation, and test sets respectively) and about 20000 coarsely annotated training images. Following the evaluation protocol [13], 19 semantic labels are used for evaluation without considering the void label.

In Tab. 2, we report the Cityscapes validation set results. Similar to MobileNets [29, 66], we adjust the model capacity by changing the filter multiplier F. As shown in the table, higher model capacity leads to better performance at the cost of slower speed (indicated by larger Multi-Adds).

Method	**ImageNet**	**Coarse**	mIOU (%)
FRRN-A [60]			63.0
GridNet [17]			69.5
FRRN-B [60]			71.8
Auto-DeepLab-S			79.9
Auto-DeepLab-L			80.4
Auto-DeepLab-S		✓	80.9
Auto-DeepLab-L		✓	82.1
ResNet-38 [82]	✓	✓	80.6
PSPNet [88]	✓	✓	81.2
Mapillary [4]	✓	✓	82.0
DeepLabv3+ [11]	✓	✓	82.1
DPC [6]	✓	✓	82.7
DRN_CRL_Coarse [91]	✓	✓	82.8

Table 4: Cityscapes test set results with *multi-scale* inputs during inference. **ImageNet:** Models pretrained on ImageNet. **Coarse:** Models exploit coarse annotations.

In Tab. 3, we show that increasing the training iterations from 500K to 1.5M iterations improves the performance by 2.8%, when employing our light-weight model variant, Auto-DeepLab-S. Additionally, adopting the Scheduled Drop Path [40, 93] further improves the performance by 1.74%, reaching 79.74% on Cityscapes validation set.

We then report the test set results in Tab. 4. Without any pretraining, our best model (Auto-DeepLab-L) significantly outperforms FRNN-B [60] by 8.6% and GridNet [17] by 10.9%. With extra coarse annotations, our model Auto-DeepLab-L, without pretraining on ImageNet [65], achieves the test set performance of 82.1%, outperforming PSPNet [88] and Mapillary [4], and attains the same performance as DeepLabv3+ [11] while requiring 55.2% fewer Mutli-Adds computations. Notably, our light-weight model variant, Auto-DeepLab-S, attains 80.9% on the test set, comparable to PSPNet, while using merely 10.15M parameters and 333.25B Multi-Adds.

5.2.2 PASCAL VOC 2012

PASCAL VOC 2012 [15] contains 20 foreground object classes and one background class. We augment the original dataset with the extra annotations provided by [23], resulting in 10582 (*train_aug*) training images.

In Tab. 5, we report our validation set results. Our best model, Auto-DeepLab-L, with single scale inference significantly outperforms [19] by 20.36%. Additionally, for all our model variants, adopting multi-scale inference improves the performance by about 1%. Further pretraining our models on COCO [46] for 4M iterations improves the

Method	**MS**	**COCO**	mIOU (%)
DropBlock [19]			53.4
Auto-DeepLab-S			71.68
Auto-DeepLab-S	✓		72.54
Auto-DeepLab-M			72.78
Auto-DeepLab-M	✓		73.69
Auto-DeepLab-L			73.76
Auto-DeepLab-L	✓		75.26
Auto-DeepLab-S		✓	78.31
Auto-DeepLab-S	✓	✓	80.27
Auto-DeepLab-M		✓	79.78
Auto-DeepLab-M	✓	✓	80.73
Auto-DeepLab-L		✓	80.75
Auto-DeepLab-L	✓	✓	82.04

Table 5: PASCAL VOC 2012 validation set results. We experiment with the effect of adopting *multi-scale* inference (**MS**) and COCO-pretrained checkpoints (**COCO**). Without any pretraining, our best model (Auto-DeepLab-L) outperforms DropBlock by 20.36%. All our models are not pretrained with ImageNet images.

Method	**ImageNet**	**COCO**	mIOU (%)
Auto-DeepLab-S		✓	82.5
Auto-DeepLab-M		✓	84.1
Auto-DeepLab-L		✓	85.6
RefineNet [44]	✓	✓	84.2
ResNet-38 [82]	✓	✓	84.9
PSPNet [88]	✓	✓	85.4
DeepLabv3+ [11]	✓	✓	87.8
MSCI [43]	✓	✓	88.0

Table 6: PASCAL VOC 2012 test set results. Our Auto-DeepLab-L attains comparable performance with many state-of-the-art models which are pretrained on both **ImageNet** and **COCO** datasets. We refer readers to the official leader-board for other state-of-the-art models.

performance significantly.

Finally, we report the PASCAL VOC 2012 test set result with our COCO-pretrained model variants in Tab. 6. As shown in the table, our best model attains the performance of 85.6% on the test set, outperforming RefineNet [44] and PSPNet [88]. Our model is lagged behind the top-performing DeepLabv3+ [11] with Xception-65 as network backbone by 2.2%. We think that PASCAL VOC 2012 dataset is too small to train models from scratch and pretraining on ImageNet is still beneficial in this case.

Method	**ImageNet**	mIOU (%)	Pixel-Acc (%)	Avg (%)
Auto-DeepLab-S		40.69	80.60	60.65
Auto-DeepLab-M		42.19	81.09	61.64
Auto-DeepLab-L		43.98	81.72	62.85
CascadeNet (VGG-16) [90]	✓	34.90	74.52	54.71
RefineNet (ResNet-152) [44]	✓	40.70	-	-
UPerNet (ResNet-101) [83] †	✓	42.66	81.01	61.84
PSPNet (ResNet-152) [88]	✓	43.51	81.38	62.45
PSPNet (ResNet-269) [88]	✓	44.94	81.69	63.32
DeepLabv3+ (Xception-65) [11] †	✓	45.65	82.52	64.09

Table 7: ADE20K validation set results. We employ *multi-scale* inputs during inference. †: Results are obtained from their up-to-date model zoo websites respectively. **ImageNet:** Models pretrained on ImageNet. Avg: Average of mIOU and Pixel-Accuracy.

5.2.3 ADE20K

ADE20K [90] has 150 semantic classes and high quality annotations of 20000 training images and 2000 validation images. In our experiments, the images are all resized so that the longer side is 513 during training.

In Tab. 7, we report our validation set results. Our models outperform some state-of-the-art models, including RefineNet [44], UPerNet [83], and PSPNet (ResNet-152) [88]; however, without any ImageNet [65] pretraining, our performance is lagged behind the latest work of [11].

6. Conclusion

In this paper, we present one of the first attempts to extend Neural Architecture Search beyond image classification to dense image prediction problems. Instead of fixating on the cell level, we acknowledge the importance of spatial resolution changes, and embrace the architectural variations by incorporating the network level into the search space. We also develop a differentiable formulation that allows efficient (about $1000\times$ faster than DPC [6]) architecture search over our two-level hierarchical search space. The result of the search, Auto-DeepLab, is evaluated by training on benchmark semantic segmentation datasets from scratch. On Cityscapes, Auto-DeepLab significantly outperforms the previous state-of-the-art by 8.6%, and performs comparably with ImageNet-pretrained top models when exploiting the coarse annotations. On PASCAL VOC 2012 and ADE20K, Auto-DeepLab also outperforms several ImageNet-pretrained state-of-the-art models.

For future work, within the current framework, related applications such as object detection should be plausible; we could also try untying the cell architecture α across different layers (*cf*. [77]) with little computation overhead. Beyond the current framework, a more general network level search space should be beneficial (*cf*. Sec. 3.2).

Acknowledgments We thank Sergey Ioffe for valuable feedback; Cloud AI and Mobile Vision team for support. CL and AY acknowledge a gift from YiTu.

References

[1] K. Ahmed and L. Torresani. Maskconnect: Connectivity learning by gradient descent. In *ECCV*, 2018. 3

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*, 2015. 2

[3] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017. 3

[4] S. R. Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, 2018. 2, 7

[5] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Efficient architecture search by network transformation. In *AAAI*, 2018. 3

[6] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*, 2018. 1, 2, 3, 7, 8

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1, 2

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 2, 7

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 2, 4, 6, 7

[10] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 2

[11] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 1, 2, 7, 8

[12] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 3, 6, 7

[14] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2

[15] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge a retrospective. *IJCV*, 2014. 2, 3, 6, 7

[16] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013. 1, 2

[17] D. Fourure, R. Emonet, E. Fromont, D. Muselet, A. Tremeau, and C. Wolf. Residual conv-deconv grid network for semantic segmentation. In *BMVC*, 2017. 2, 3, 7

[18] J. Fu, J. Liu, Y. Wang, and H. Lu. Stacked deconvolutional network for semantic segmentation. *arXiv:1708.04943*, 2017. 2

[19] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *NIPS*, 2018. 7, 8

[20] A. Giusti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *ICIP*, 2013. 2

[21] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005. 2

[22] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *arXiv:1503.04069*, 2015. 3

[23] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 7

[24] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[26] X. He, R. S. Zemel, and M. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *CVPR*, 2004. 2

[27] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 1

[28] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets: Time-Frequency Methods and Phase Space*, pages 289–297. Springer, 1989. 2

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017. 7

[30] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2

[32] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 2, 7

[33] M. A. Islam, M. Rochan, N. D. Bruce, and Y. Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, 2017. 2

[34] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, 2015. 3

[35] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting crfs with boltzmann machine shape priors for image labeling. In *CVPR*, 2013. 3

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1, 6

[37] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009. 2

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[39] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009. 2

[40] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 7

[41] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2

[42] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 2

[43] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang. Multi-scale context intertwining for semantic segmentation. In *ECCV*, 2018. 8

[44] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *CVPR*, 2017. 2, 8

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4

[46] T.-Y. Lin et al. Microsoft coco: Common objects in context. In *ECCV*, 2014. 7

[47] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *ECCV*, 2018. 1, 2, 3

[48] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018. 3

[49] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv:1806.09055*, 2018. 1, 2, 3, 5

[50] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015. 2, 7

[51] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[52] R. Luo, F. Tian, T. Qin, and T.-Y. Liu. Neural architecture optimization. In *NIPS*, 2018. 3

[53] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. *arXiv:1703.00548*, 2017. 3

[54] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feed-forward semantic segmentation with zoom-out features. In *CVPR*, 2015. 2

[55] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 1, 2, 4

[56] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1, 4

[57] G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *CVPR*, 2015. 2

[58] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *CVPR*, 2017. 2

[59] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 2, 3

[60] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017. 2, 3, 7

[61] H. Qi, Z. Zhang, B. Xiao, H. Hu, B. Cheng, Y. Wei, and J. Dai. Deformable convolutional networks – coco detection and segmentation challenge 2017 entry. *ICCV COCO Challenge Workshop*, 2017. 2

[62] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. *arXiv:1802.01548*, 2018. 1, 2, 3

[63] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017. 2, 3

[64] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 2, 4

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2, 7, 8

[66] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 7

[67] S. Saxena and J. Verbeek. Convolutional neural fabrics. In *NIPS*, 2016. 3

[68] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[69] R. Shin, C. Packer, and D. Song. Differentiable neural network architecture search. In *ICLR Workshop*, 2018. 2, 3

[70] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009. 2

[71] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv:1612.06851*, 2016. 4

[72] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[73] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 1

[74] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1, 2

[75] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[76] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1, 2

[77] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv:1807.11626*, 2018. 3, 8

[78] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018. 7

[79] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi, and J. Uijlings. The devil is in the decoder. In *BMVC*, 2017. 2

[80] Y. Wu and K. He. Group normalization. In *ECCV*, 2018. 1

[81] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016. 1

[82] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv:1611.10080*, 2016. 2, 7, 8

[83] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 2, 8

[84] L. Xie and A. Yuille. Genetic cnn. In *ICCV*, 2017. 3

[85] S. Xie, R. Girshick, P. Dollr, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 2

[86] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 2

[87] Z. Zhang, X. Zhang, C. Peng, D. Cheng, and J. Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018. 2

[88] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 7, 8

[89] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018. 3

[90] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 2, 3, 6, 8

[91] Y. Zhuang, F. Yang, L. Tao, C. Ma, Z. Zhang, Y. Li, H. Jia, X. Xie, and W. Gao. Dense relation network: Learning consistent and context-aware representation for semantic image segmentation. In *ICIP*, 2018. 7

[92] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2, 3

[93] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 2, 3, 7