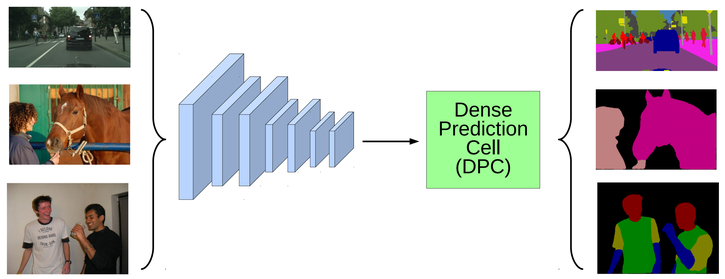
Searching for Efficient Multi-Scale Architectures for Dense Image Prediction

Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, Jonathon Shlens
Google Inc.

Motivation



- Meta-learning approaches may exceed human-invented architectures on scalable image classification (Zoph et al, 2017).
- The real promise of meta-learning is our ability to extend these techniques to other domains (Saxena and Verbeek, 2016; Zoph and Le, 2017).
- We present a first effort towards applying meta-learning to dense image prediction.

Overview

1. Challenges: Naively porting ideas from image classification (Russakovsky et al., 2015) would not suffice:

- Network motifs and operations notably differ.
- Architecture search must inherently operate on high resolution imagery to capture multi-scale information.

2. Overview of proposed methods:

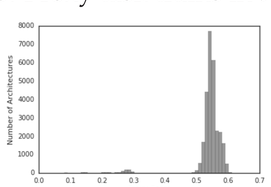
- A computationally tractable and simple proxy task.
- A recursive search space for a dense prediction cell (DPC).

Methods- Proxy task

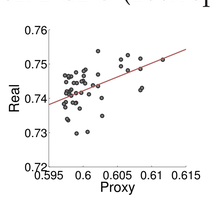
1. Goal of proxy tasks:
- Fast to compute, and must correlate with large-scale task.

2. Our proposed proxy tasks:
- A DPC is directly built on the cached feature maps.

3. Proxy task trains in 90 minutes on 1 GPU (100x speed-up).



(a) Score distribution



(b) Predictive of large-scale task ($\rho = 0.46$)

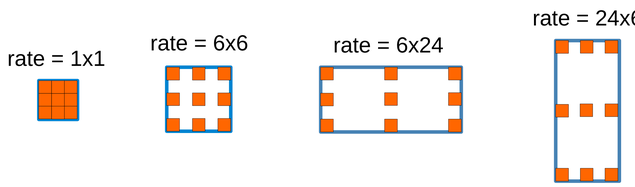
Methods- Search Space

1. Recursive search space built from operations gleaned from dense prediction literature.

- DPC is a DAG graph with B branches.
- Branch b_i is represented by a 3-tuple, (X_i, OP_i, Y_i), where $X_i \in \mathcal{X}_i$ is the input tensor, $OP_i \in \mathcal{OP}$ is the operation on input X_i, and Y_i is the output tensor.
- The final output, $Y = concat(Y_1, Y_2, \ldots, Y_{\mathcal{B}})$
- $\mathcal{X}_i = \{\mathcal{F}, Y_1, \ldots, Y_{i-1}\}$ (i.e., last network backbone feature maps, $\mathcal{F}$, plus all outputs obtained by previous branches).
- $\mathcal{X}_1 = \{\mathcal{F}\}$, i.e., the first branch can only take $\mathcal{F}$ as input.

2. The operator space, $\mathcal{OP}$, is defined as:

- Convolution with a 1×1 kernel.
- 3×3 atrous separable convolution (Howard et al., 2017; Chen et al., 2018) with rate $r_h \times r_w$, where r_h and $r_w \in \{1, 3, 6, 9, \ldots, 21\}$.
- Average spatial pyramid pooling (Zhao et al., 2017) with grid size $g_h \times g_w$, where g_h and $g_w \in \{1, 2, 4, 8\}$.



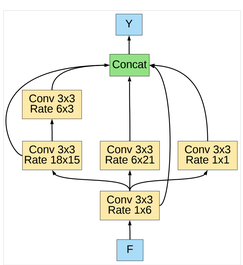
3. Our proposed search space encodes leading architectures:

- Each branch of the cell could be built in parallel or in cascade.
- For $\mathcal{B} = 5$ branches, the search space contains $\mathcal{B}! \times 81^{\mathcal{B}} \approx 4.2 \times 10^{11}$ configurations.

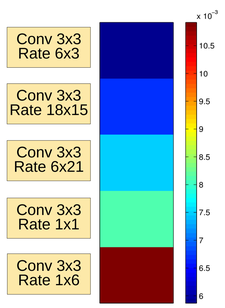
4. We build on top of an efficient random search algorithm (Golovin et al., 2017).

- Sampling points uniformly at random as well as sampling some points near the currently best observed architectures (in total 28K architectures and 2600 GPU hours).

5. Found DPC architecture:



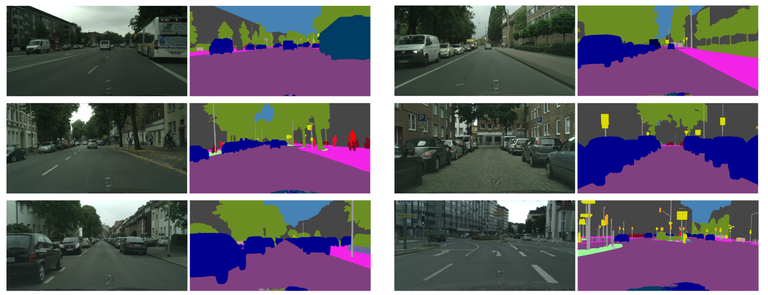
(a) Top-1 DPC



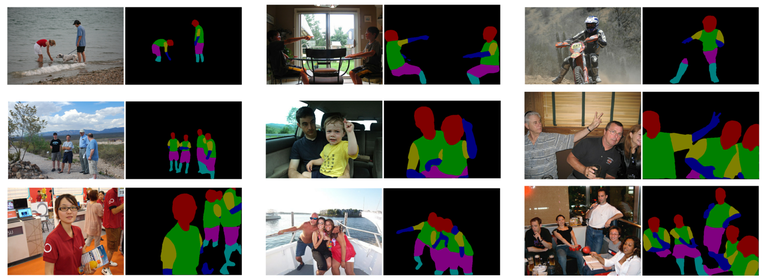
(b) Filter weights ($L1$ norm)

Experimental Results

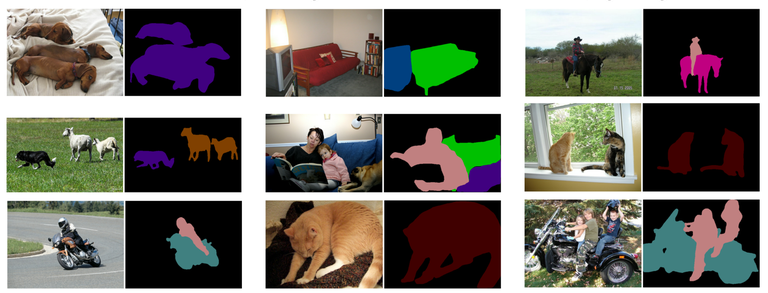
1. Cityscapes (Cordts et al., 2016): 19 classes (e.g., road, pedestrian).



2. PASCAL-Person-Part (Chen et al., 2014): 7 classes (e.g., head, torso).



3. PASCAL VOC 2012 (Everingham et al., 2014): 21 classes (e.g., dog, sofa).



Method	mIOU
PSPNet	81.2
Mapillary Research	82.0
DeepLabv3+	82.1
DPC (ours)	**82.7**

(a) Cityscapes test set

Method	mIOU
Liang *et al.*	63.57
Xia *et al.*	64.39
Fang *et al.*	67.60
DPC (ours)	71.34

(b) PASCAL-Person-Part val set

Method	mIOU
EncNet	85.9
DFN	86.2
DeepLabv3+	87.8
ExFuse	87.9
MSCI	88.0
DPC (ours)	87.9

(c) PASCAL VOC 2012 test set

Conclusions / Future Directions

- This work demonstrates that meta-learning achieves state-of-the-art results on 3 dense image prediction tasks.
- Employing more sophisticated meta-learning outside (e.g. reinforcement learning, SMBO) may lead to further gains.
- A version of the dense prediction cell may be downloaded from: `https://github.com/tensorflow/models/tree/master/research/deeplab`