Pixel-Adaptive Convolutional Neural Networks

Hang Su[1], Varun Jampani[2], Deqing Sun[2], Orazio Gallo[2], Erik Learned-Miller[1], Jan Kautz[2]

[1] UMass Amherst [2] NVIDIA

video intro

CVPR LONG BEACH • JUNE 16-20, 2019

PAC is a content-adaptive operation that generalizes spatial convolutions

https://suhangpro.github.io/pac (code available)

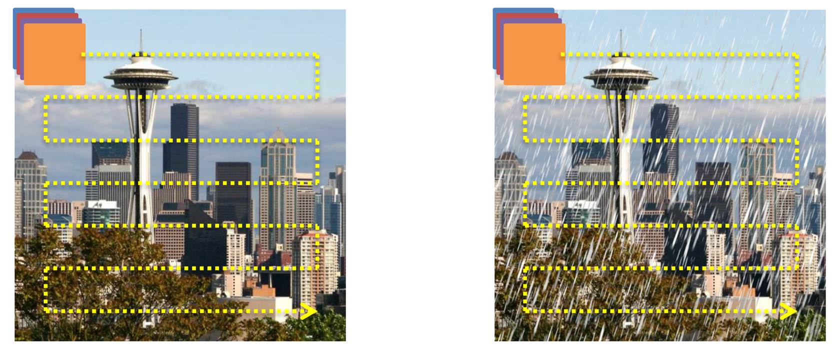
1 Motivation

$$\mathbf{v}'_i = \sum_{j \in \Omega(i)} \mathbf{W}[\mathbf{p}_i - \mathbf{p}_j]\mathbf{v}_j + \mathbf{b}$$

Spatial convolution
- Fundamental building block of modern neural networks
- Provides an efficient and effective way to propagate and integrate features across image pixels

It has advantages …
- Simple formulation → efficient parallelization
- Spatial sharing → capturing invariance
- Allows hierarchical features learning

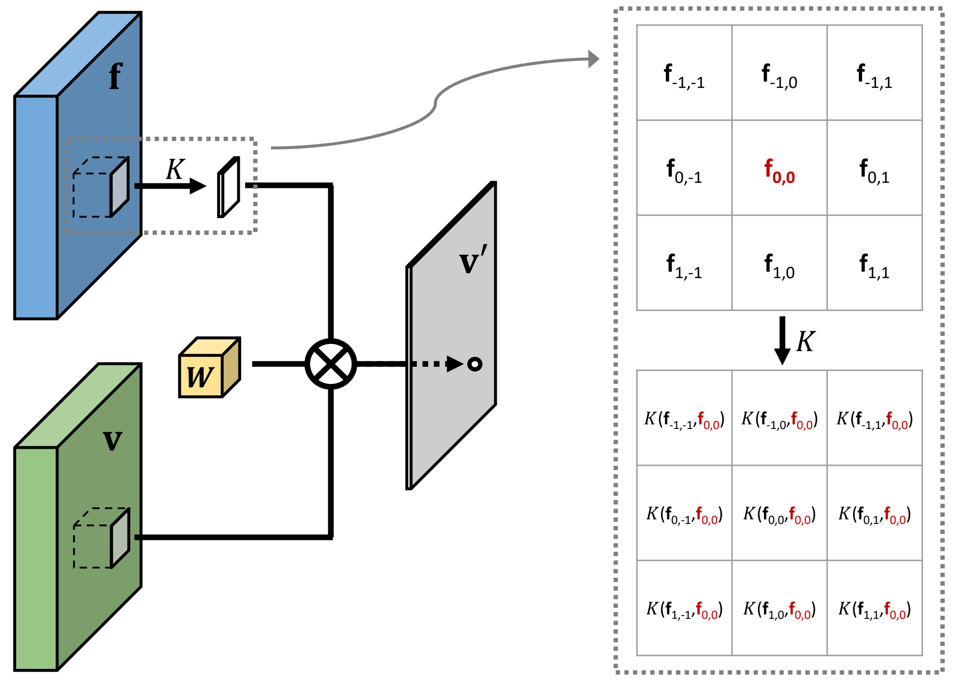
But the filters are content-agnostic …
- Loss gradient are pooled across pixel locations.
- Once learned, same filter banks are used across all images and pixels locations.



Goal: Make convolutions adaptive to image content

2 Pixel-Adaptive Convolution (PAC)

$$\mathbf{v}'_i = \sum_{j \in \Omega(i)} K(\mathbf{f}_i, \mathbf{f}_j)\mathbf{W}[\mathbf{p}_i - \mathbf{p}_j]\mathbf{v}_j$$

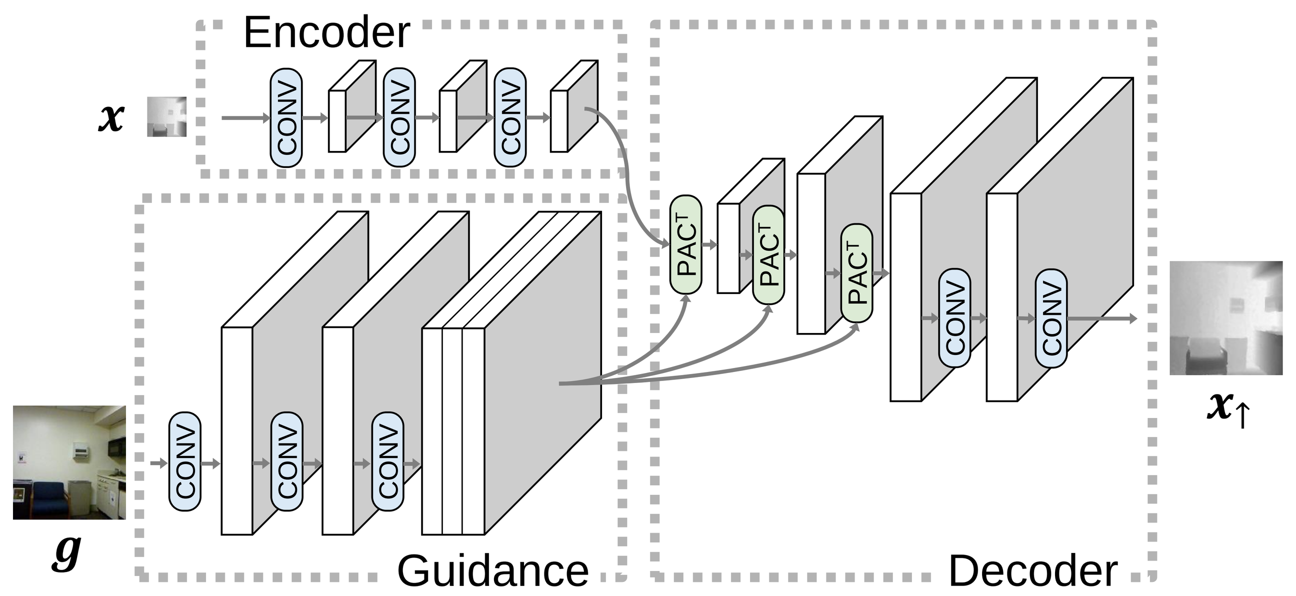


$\mathbf{v}$	input features	$\mathbf{W}$	filter weights
$\mathbf{v}'$	output features	$\mathbf{f}$	adapting features
$\mathbf{p}$	(x,y) coordinates	K	adapting kernel

Making convolution content-adaptive
- Kernel function K *modifies* filters according to $\mathbf{f}$.
- In contrast, kernel-prediction networks [1,2,3] generate filters directly: $\mathbf{v}'_i = \sum_{j\in\Omega(i)} \mathbf{W}_i[\mathbf{p}_i - \mathbf{p}_j]\mathbf{v}_j$.
- We use $K(\mathbf{f}_i, \mathbf{f}_j) = \exp(-\frac{1}{2}\|\mathbf{f}_i - \mathbf{f}_j\|^2)$ for our experiments.
- $\mathbf{f}$ and $\mathbf{v}$ can both be learned through backpropagation.

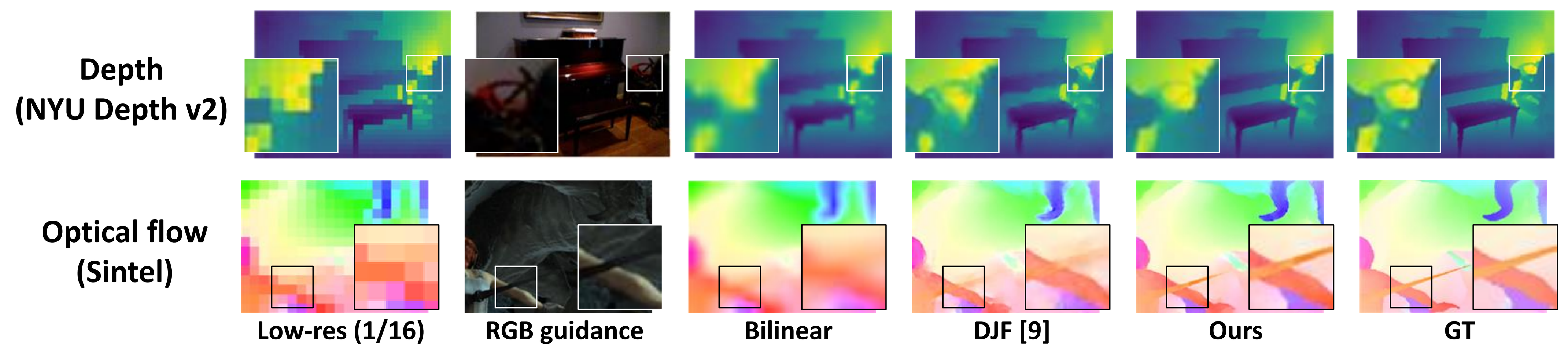
PAC generalizes many existing operations
- Spatial convolution: $K(\mathbf{f}_i, \mathbf{f}_j) = 1$
- Bilateral filter: $\mathbf{f} = (r, g, b)$, $K(\mathbf{f}_i, \mathbf{f}_j) = \exp(-\frac{1}{2\alpha_1}\|\mathbf{f}_i - \mathbf{f}_j\|^2)$, $\mathbf{W}[\mathbf{p}_i - \mathbf{p}_j] = \exp(-\frac{1}{2\alpha_2}\|\mathbf{p}_i - \mathbf{p}_j\|^2)$
- Average pooling: $K(\mathbf{f}_i, \mathbf{f}_j) = 1, \mathbf{W}[\mathbf{p}_i - \mathbf{p}_j] = \frac{1}{F^2}$
- Detail-preserving pooling [4]: $K(\mathbf{f}_i, \mathbf{f}_j) = \alpha + (\|\mathbf{f}_i - \mathbf{f}_j\|^2 + \epsilon^2)^\lambda$

3 Deep Joint Upsampling Networks



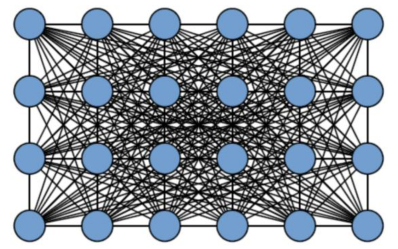
Method	4×	8×	16×
Bicubic	8.16	14.22	22.32
MRF	7.84	13.98	22.20
GF [5]	7.32	13.62	22.03
JBU [6]	4.07	8.29	13.35
DMSG [7]	3.78	6.37	11.16
FBS [8]	4.29	8.94	14.59
DJF [9]	3.38	5.86	10.11
Ours	2.39	4.59	8.09

Results on NYU Depth v2



4 Efficient CRF with PAC

Fully Connected CRF [10]



$$p(\mathbf{l}|I) \propto \exp\left\{-\sum_i \psi_u(l_i|I) - \sum_{i<j}\psi_p(l_i, l_j|I)\right\}$$

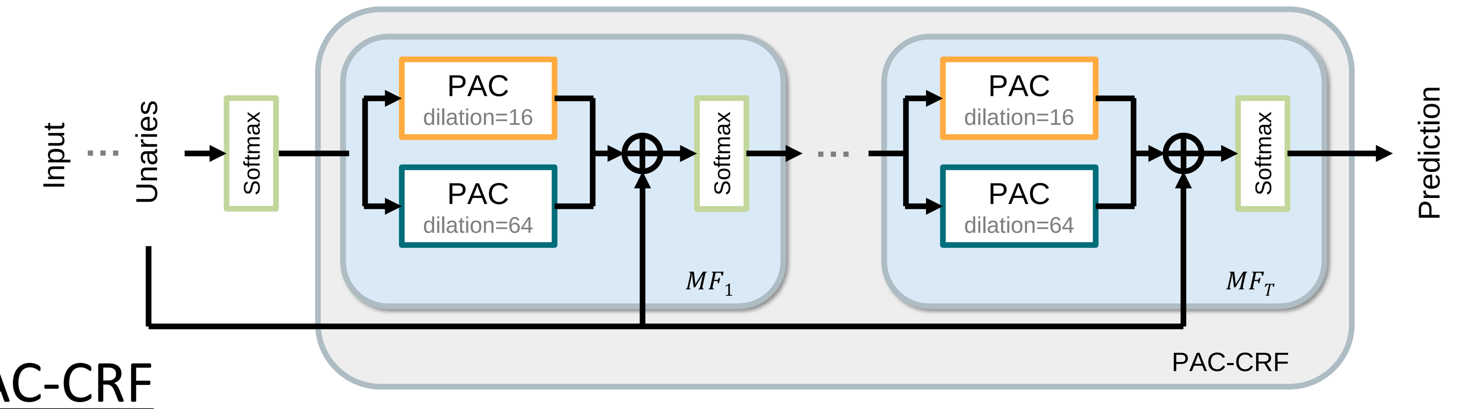
- Pairwise term:

$$\psi_p(l_i, l_j|I) = \mu(l_i, l_j)K(\mathbf{f}_i, \mathbf{f}_j)$$

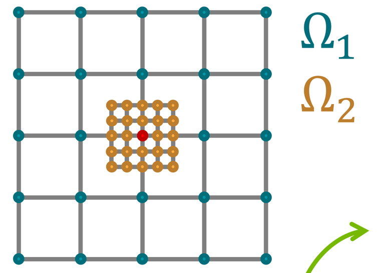
$$K(\mathbf{f}_i, \mathbf{f}_j) = w_1 \exp\left\{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\theta_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\theta_\beta^2}\right\} + w_2 \exp\left\{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\theta_\gamma^2}\right\}$$

- Mean-field update rule:

$$Q_i^{(t+1)}(l) \leftarrow \frac{1}{Z_i}\exp\left\{-\psi_u(l) - \sum_{l'\in\mathcal{L}}\mu(l, l')\sum_{j\neq i}K(\mathbf{f}_i, \mathbf{f}_j)Q_j^{(t)}(l')\right\}$$



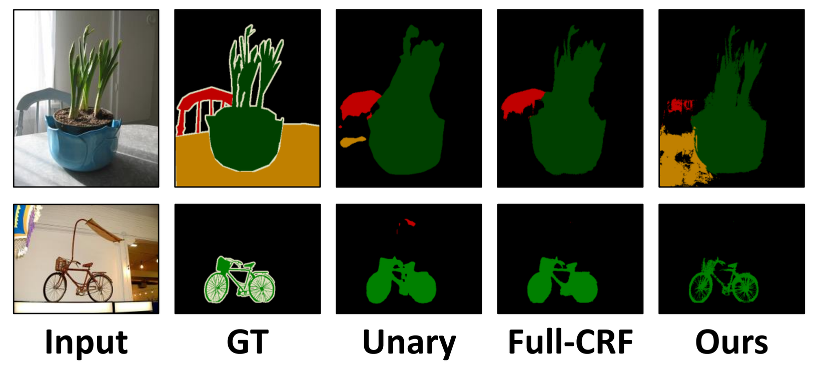
PAC-CRF
- A more general pairwise term: $\psi_p(l_i, l_j|I) = W(\mathbf{p}_j - \mathbf{p}_i, l_i, l_j)K(\mathbf{f}_i, \mathbf{f}_j)$
- *Learned* features in K: $K(\mathbf{f}_i, \mathbf{f}_j) = \exp\left\{-\frac{1}{2}\|\mathbf{f}_i - \mathbf{f}_j\|^2\right\}$
- Local connections with multiple *dilation factors*



- Update rule: $Q_i^{(t+1)}(l) \leftarrow \frac{1}{Z_i}\exp\left\{-\psi_u(l) - \underbrace{\sum_{l'\in\mathcal{L}}\sum_{j\in\Omega(i)} W(\mathbf{p}_j - \mathbf{p}_i, l, l')K(\mathbf{f}_i, \mathbf{f}_j)Q_j^{(t)}(l')}_{\text{PAC!}}\right\}$

Method	mIoU	CRF Runtime
Unary only (FCN)	67.20	-
Full-CRF [10]	+2.45	629ms
BCL-CRF [11]	+2.33	2.6s
Conv-CRF [12]	+1.57	38ms
PAC-CRF, 32	+2.21	39ms
PAC-CRF, 16-64	+2.62	78ms

Results on Pascal VOC2012 (test)



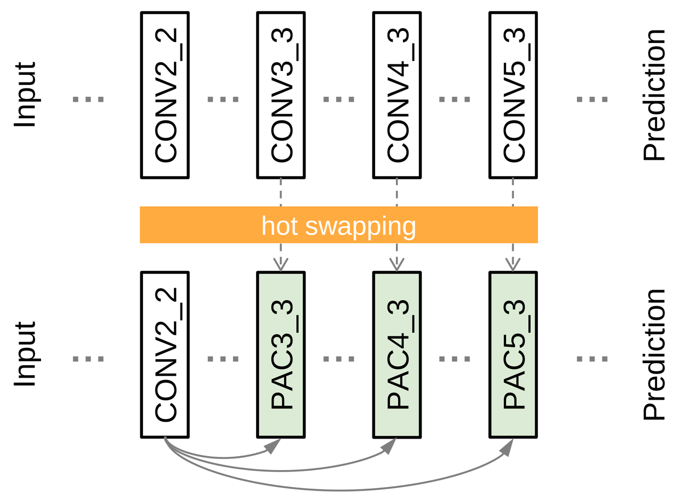
5 Layer Hot-Swapping with PAC

A modification to pre-trained networks that:
- Replace certain Conv layers with PAC counterparts
- Retain the pre-trained weights
- Reuse earlier layers for adapting features $\mathbf{f}$

Experiments
- ~2% improvement w/ minimal runtime overhead
- Complementary with PAC-CRF improvement

Method	mIoU	Runtime
FCN	67.20	39ms
PAC-FCN	69.18	41ms
PAC-FCN + PAC-CRF	71.34	118ms

Results on Pascal VOC2012 (test)



- ✓ Drop-in replacement
- ✓ Zero added parameters
- ✓ Minimal overhead

References

[1] Brabandere et al. Dynamic filter networks. NIPS '16.
[2] Bako et al. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. ToG '17.
[3] Wu et al. Dynamic sampling convolutional neural networks. ECCV '18.
[4] Saeedan et al. Detail preserving pooling in deep networks. CVPR '18.
[5] He et al. Guided image filtering. ECCV '10.
[6] Kopf et al. Joint bilateral upsampling. ToG '07.
[7] Hui et al. Depth map super-resolution by deep multi-scale guidance. ECCV '16.
[8] Barron and Poole. Fast bilateral solver. ECCV '16.
[9] Li et al. Joing image filtering with deep convolutional networks. TPAMI '18.
[10] Krähenbühl and Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. NIPS '11.
[11] Jampani et al. Learning sparse high-dimensional filters: image filtering, dense CRFs and bilateral neural networks. CVPR '16.
[12] Teichmann and Cipolla. Convolutional CRFs for semantic segmentation. arXiv:1805.04777.

Contact: hsu@cs.umass.edu; vjampani@nvidia.com