Fully Convolutional Networks

Jon Long and Evan Shelhamer
CVPR15 Caffe Tutorial
pixels in, pixels out

monocular depth estimation (Liu et al. 2015)

boundary prediction (Xie & Tu 2015)
end-to-end learning

< 1/5 second
a classification network
becoming fully convolutional

convolution

227 × 227  55 × 55  27 × 27  13 × 13  1 × 1
becoming fully convolutional

convolution

H × W  H/4 × W/4  H/8 × W/8  H/16 × W/16  H/32 × W/32
upsampling output

convolution

H × W  H/4 × W/4  H/8 × W/8  H/16 × W/16  H/32 × W/32  H × W
end-to-end, pixels-to-pixels network

convolution

H × W  H/4 × W/4  H/8 × W/8  H/16 × W/16  H/32 × W/32  H × W
end-to-end, pixels-to-pixels network

conv, pool, nonlinearity

upsampling

pixelwise output + loss
spectrum of deep features

combine *where* (local, shallow) with *what* (global, deep)

image

intermediate layers

fuse features into deep jet

(cf. Hariharan et al. CVPR15 “hypercolumn”)

skip layers

end-to-end, joint learning of semantics and location
skip layer refinement

input image

<table>
<thead>
<tr>
<th>stride</th>
<th>no skips</th>
<th>1 skip</th>
<th>2 skips</th>
<th>ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

skip layer refinement
training + testing

- train full image at a time *without patch sampling*
- reshape network to take input of any size
- forward time is \(~150\text{ms}\) for \(500 \times 500 \times 21\) output
Relative to prior state-of-the-art SDS:

- 20% improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation
Hariharan et al. ECCV14
models + code

fully convolutional networks are fast, end-to-end models for pixelwise problems

- **code** in Caffe branch (merged soon)
- **models** for PASCAL VOC, NYUDv2, SIFT Flow, PASCAL-Context in Model Zoo

fcn.berkeleyvision.org

caffe.berkeleyvision.org

github.com/BVLC/caffe
models

- **PASCAL VOC** standard for object segmentation
- **NYUDv2** multi-modal rgb + depth scene segmentation
- **SIFT Flow** multi-task for semantic + geometric segmentation
- **PASCAL-Context** object + scene segmentation
```python
import numpy as np
from PIL import Image

import caffe

# load image, switch to BGR, subtract mean, and make dims C x H x W for Caffe
in_ = np.array(im, dtype=np.float32)
in_ = in_[:, :, ::-1]
in_ -= np.array([104.00698793, 116.6876762, 122.67891434])
in_ = in_.transpose((2, 0, 1))

# load net
net = caffe.Net('deploy.prototxt', 'fcn-32s-pascalcontext.caffemodel', caffe.TEST)
# shape for input (data blob is N x C x H x W), set data
net.blobs['data'].reshape(1, *in_.shape)
net.blobs['data'].data[:] = in_

# run net and take argmax for prediction
net.forward()
out = net.blobs['score'].data[0].argmax(axis=0)
```

inference script (gist)
solving

```python
# base net -- follow the editing model parameters example to make
# a fully convolutional VGG16 net.
# http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/net_surgery.ipynb
base_weights = 'vgg16fc.caffemodel'

# init
caffe.set_mode_gpu()
caffe.set_device(0)

solver = caffe.SGDSolver('solver.prototxt')

# do net surgery to set the deconvolution weights for bilinear interpolation
interp_layers = [k for k in solver.net.params.keys() if 'up' in k]
interp_surgery(solver.net, interp_layers)

# copy base weights for fine-tuning
solver.net.copy_from(base_weights)

# solve straight through -- a better approach is to define a solving loop to
# 1. take SGD steps
# 2. score the model by the test net `solver.test_nets[0]`
# 3. repeat until satisfied
solver.step(80000)
```
solving script (gist)
Reshape

- Decide shape on-the-fly in C++ / Python / MATLAB
- DataLayer automatically reshapes for batch size == 1
- Essentially free (only reallocates when necessary)
Helpful Layers

- Losses can take spatial predictions + truths
- Deconvolution / “backward convolution” can compute interpolation
- Crop: maps coordinates between layers
FCN for Pose Estimation

Georgia Gkioxari
UC Berkeley
FCN for Pose Estimation

Input data:

Image
FCN for Pose Estimation

Input data:

Image

Keypoints
FCN for Pose Estimation

Input data:

- **Image**
- **Keypoints**

Define an area around the keypoint as its positive neighborhood with radius $r$. 
FCN for Pose Estimation

Input data:

- Image
- Keypoints
- Labels
FCN for Pose Estimation

Input data:
Heat Map Predictions from FCN

Test Image

Right Ankle
Right Knee
Right Hip
Right Wrist
Right Elbow
Right Shoulder
Heat Map Predictions from FCN

Test Image

Right Ankle  Right Knee  Right Hip  Right Wrist  Right Elbow  Right Shoulder

Two modes because there are two Right Shoulders in the image!
# Heat Maps to Keypoints

<table>
<thead>
<tr>
<th>PCK @ 0.2</th>
<th>LSP test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ankle</td>
<td>56.5</td>
</tr>
<tr>
<td>Knee</td>
<td>60.0</td>
</tr>
<tr>
<td>Hip</td>
<td>56.6</td>
</tr>
<tr>
<td>Wrist</td>
<td>62.9</td>
</tr>
<tr>
<td>Elbow</td>
<td>71.8</td>
</tr>
<tr>
<td>Shoulder</td>
<td>78.8</td>
</tr>
<tr>
<td>Head</td>
<td>93.6</td>
</tr>
</tbody>
</table>

- **FCN baseline PCK == ~69%**
- **State-of-the-art == ~72%**
Details

Architecture:

- FCN - 32 stride. **No** data augmentation.
- radius = 0.1*im.shape[0] (**no** cross validation)

Runtime on a K40:

- 0.7 sec/iteration for training (15hrs for 80K iterations)
- 0.25 sec/image for inference for all keypoints
conclusion

fully convolutional networks are fast, end-to-end models for pixelwise problems

- **code** in Caffe branch (merged soon)
- **models** for PASCAL VOC, NYUDv2, SIFT Flow, PASCAL-Context in Model Zoo

[fcn.berkeleyvision.org](http://fcn.berkeleyvision.org)
[caffe.berkeleyvision.org](http://caffe.berkeleyvision.org)
[github.com/BVLC/caffe](http://github.com/BVLC/caffe)