Pixel Prediction Tasks

Semantic segmentation

Colorization

Depth / Surface Normal Estimation

Many Slides from L. Lazebnik.
Outline

• Semantic segmentation
  • Architectures
    • “Convolutionalization”
    • Dilated convolutions
    • Hyper-columns / skip-connections
    • Learned up-sampling architectures

• Other dense prediction problems
From Image Classifiers to Semantic Segmentation

**Have:** an pre-trained image classification network

**Want:** pixel-wise predictions on arbitrary sized images

Sparse, Low-resolution Output

Sparse, Low-resolution Output

Fix 1: A trous Conv., Dilated Conv.

Fix 1: A trous Conv., Dilated Conv.

- $16 \times 16$
  - $ks = 3$, stride = 2, dilation = 1
  - $8 \times 8$
  - $ks = 3$, stride = 2, dilation = 1
  - $4 \times 4$
  - $ks = 3$, stride = 2, dilation = 1
  - $2 \times 2$
  - $3 \times 3$
  - $ks = 3$, stride = 1, dilation = 1
  - $5 \times 5$
  - $9 \times 9$
  - $16 \times 16$
  - $ks = 3$, stride = 1, dilation = 1
  - $16 \times 16$
  - $ks = 3$, stride = 1, dilation = 2
  - $16 \times 16$
  - $ks = 3$, stride = 1, dilation = 4
Fix 1: A trous Conv., Dilated Conv.

Same as running the CNN on shifted versions of the image and stitching.

A. 3x3 conv, stride 2

B. 3x3 conv, stride 1
Fix 1: A trous Conv., Dilated Conv.

Dilation factor 1

Dilation factor 2

Dilation factor 3
Fix 1: A trous Conv., Dilated Conv.

- Use in FCN to remove downsampling: change stride of max pooling layer from 2 to 1, dilate subsequent convolutions by factor of 2 (possibly without re-training any parameters)
- Instead of reducing spatial resolution of feature maps, use a large sparse filter

Fix 1: A trous Conv., Dilated Conv.

F. Yu and V. Koltun, Multi-scale context aggregation by dilated convolutions, ICLR 2016
Fix 2: Hyper-columns/Skip Connections

• Even though with dilation we can predict each pixel, fine-grained information needs to be propagated through the network.

• Idea: Additionally use features from within the network.

Fix 2: Hyper-columns/Skip Connections

- Predictions by 1x1 conv layers, bilinear upsampling

- Predictions by 1x1 conv layers, \textit{learned} 2x upsampling, fusion by summing

Fix 2: Hyper-columns/Skip Connections

FCN-32s  FCN-16s  FCN-8s  Ground truth

---

Fix 2b: Learned Upsampling

• Predictions by 1x1 conv layers, bilinear upsampling

• Predictions by 1x1 conv layers, \textit{learned} 2x upsampling, fusion by summing

U-Net

- Like FCN, fuse upsampled higher-level feature maps with higher-res, lower-level feature maps
- Unlike FCN, fuse by concatenation, predict at the end

O. Ronneberger, P. Fischer, T. Brox  *U-Net: Convolutional Networks for Biomedical Image Segmentation*, MICCAI 2015
Learned Upsampling (Transposed convolution)

• Use the filter to “paint” in the output: place copies of the filter on the output, multiply by corresponding value in the input, sum where copies of the filter overlap

• 1D example:

![Diagram showing input and output with a filter applied]

Input: \( x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \)

Filter: \([w_1 \quad w_2 \quad w_3]\)

Output: \( w_1 x_1 \)

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

• Use the filter to “paint” in the output: place copies of the filter on the output, multiply by corresponding value in the input, sum where copies of the filter overlap

• 1D example:

\[
\begin{align*}
\text{input} & : x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 \\
\text{output} & : w_2 x_1 + w_1 x_2 \\
\text{filter:} & \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}
\end{align*}
\]

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

- Use the filter to “paint” in the output: place copies of the filter on the output, multiply by corresponding value in the input, sum where copies of the filter overlap

- 1D example:

\[ w_1 x_1 + w_2 x_2 + w_3 x_3 \]

Same as convolution with a flipped filter!

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

- Use the filter to “paint” in the output: place copies of the filter on the output, multiply by corresponding value in the input, sum where copies of the filter overlap
- 1D example:

\[
\begin{align*}
    w_3 x_2 + w_2 x_3 + w_1 x_4
\end{align*}
\]

Same as convolution with a flipped filter!

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use output stride $> 1$

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- Backwards-strided convolution: to increase resolution, use output stride > 1

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use *output stride* $> 1$

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use *output stride* $> 1$

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use *output stride* $> 1$

**Animation:** [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use *output stride* > 1

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use *output stride > 1*
  - For stride 2, dilate the input by inserting rows and columns of zeros between adjacent entries, convolve with flipped filter
  - Sometimes called convolution with *fractional input stride 1/2*

Q: What 3x3 filter would correspond to bilinear upsampling?

<table>
<thead>
<tr>
<th>1/4</th>
<th>1/2</th>
<th>1/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>1/4</td>
<td>1/2</td>
<td>1/4</td>
</tr>
</tbody>
</table>

V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, arXiv 2018
Upsampling by unpooling

- Alternative to transposed convolution: max unpooling

Max pooling

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Max unpooling

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Remember pooling indices (which element was max)

Output is sparse, so unpooling is typically followed by a transposed convolution layer
Fix 3: Use local edge information (CRFs)

\[ P(y|x) = \frac{1}{Z} e^{-E(y,x)} \]

\[ y^* = \arg \max_y P(y|x) \]

\[ = \arg \min_y E(y, x) \]

\[ E(y, x) = \sum_i E_{data}(y_i, x) + \sum_{i,j \in \mathcal{N}} E_{smooth}(y_i, y_j, x) \]

Source: B. Hariharan
Fix 3: Use local edge information (CRFs)

Idea: take convolutional network prediction and sharpen using classic techniques

**Conditional Random Field**

\[
y^* = \arg \min_y \sum_i E_{data}(y_i, x) + \sum_{i,j \in \mathcal{N}} E_{smooth}(y_i, y_j, x)
\]

\[
E_{smooth}(y_i, y_j, x) = \mu(y_i, y_j) w_{ij}(x)
\]

- **Label compatibility**
- **Pixel similarity**

Source: B. Hariharan
Fix 3: Use local edge information (CRFs)

Largely unnecessary given modern networks

Source: B. Hariharan
The model on the coarse annotations in order to compete with other state-of-art performance of 79.55% on the validation set.

Of the deeper network backbone (denoted as X-71 in the table), attains the best what \[ tive to increase more layers in the entry flow in the Xception \[
\]

VOC 2012 dataset. We also discover that on the Cityscapes dataset, it is e\[\]

in DeepLab model, the image-level features are more e\[\]

mented image-level feature improves the performance to 79.14%, showing that \[\]

validation set. Adding the proposed decoder module significantly improves the \[\]

module and image-level features \[\]

as feature extractor and \[\]

the naive bilinear upsampling (denoted as \[\]

pling. (b) Qualitative e\[\]

when employing \[\]

Fig. 5.

After finding the best model variant on \[\]

Table 6. (a) mIOU as a function of trimap band width around the object boundaries \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]

31 \[\]
Outline

• Semantic segmentation
  • Architectures
    • “Convolutionalization”
    • Dilated convolutions
    • Hyper-columns / skip-connections
    • Learned up-sampling architectures

• Other dense prediction problems
Other dense prediction tasks

• Depth estimation
• Surface normal estimation
• Colorization
• ....
Depth and normal estimation

D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015
Depth and normal estimation

D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015
Colorization