Learning for vision V architectures

Karel Zimmermann
http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Outline

• Architectures of classification networks
• Architectures of segmentation networks
• Architectures of regression networks
• Architectures of detection networks
• Architectures of feature matching networks
Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: **Steel drum**
Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: **Steel drum**
Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: **Steel drum**
Classification results

http://image-net.org/challenges/LSVRC/2017/index

Label: **Steel drum**
Classification results

AlexNet
8 layers

Classification Error

0.3
0.25
0.2
0.15
0.1
0.05
0.05
0

2010
2011
2012

0.28
0.26
??
AlexNet on ImageNet 2012 (over 27k citations !!!)

- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?

AlexNet on ImageNet 2012 (over 27k citations !!!)

- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp, 3x3 filters, stride=2, pad=0)?

AlexNet on ImageNet 2012 (over 27k citations !!!)

- Param in layer1 (conv, 96 11x11 filters, stride=4, pad=0)?
- Param in layer2 (maxp, 3x3 filters, stride=2, pad=0)?
- Param in layer3 (conv, 256 5x5 filters, stride=1, pad=2)?
- Parameters in total: 60M, Depth: 8 layers

Classification results

AlexNet
8 layers

Classification Error

<table>
<thead>
<tr>
<th>Year</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>0.28</td>
</tr>
<tr>
<td>2011</td>
<td>0.26</td>
</tr>
<tr>
<td>2012</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Classification results

AlexNet
8 layers
VGGnet
19 layers

Classification Error

<table>
<thead>
<tr>
<th>Year</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0.28</td>
<td>0.26</td>
<td>0.16</td>
<td>?</td>
</tr>
</tbody>
</table>

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
• Parameters in total: 138M, Depth: 19 layers
Simonyan and Zissermann, Very Deep Convolutional Networks for Large Scale Image Recognition, 2014
https://arxiv.org/abs/1409.1556
Simonyan and Zissermann, *Very Deep Convolutional Networks for Large Scale Image Recognition*, 2014

https://arxiv.org/abs/1409.1556

- **VGGNet vs AlexNet**
  - **AlexNet**: large filters, shallow (8 layers)
  - **VGGNet**: small filters, deeper (19 layers)

- Parameters in total: 138M, Depth: 19 layers
VGGNet vs AlexNet

- AlexNet: one 7x7 filter (49+1 params)

Image from: https://mc.ai/cnn-architectures-vggnet/
Simonyan and Zissermann, Very Deep Convolutional Networks for Large Scale Image Recognition, 2014
https://arxiv.org/abs/1409.1556
VGGLNet vs AlexNet

- VGGLNet: three 3x3 filters (3x9+3 params) has the same reception filed
- AlexNet: one 7x7 filter (49+1 params)

Image from: https://mc.ai/cnn-architectures-vggnet/
Simonyan and Zissermann, Very Deep Convolutional Networks for Large Scale Image Recognition, 2014
https://arxiv.org/abs/1409.1556
Classification results

AlexNet
8 layers

VGGnet
19 layers

Classification Error

<table>
<thead>
<tr>
<th>Year</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>0.28</td>
</tr>
<tr>
<td>2011</td>
<td>0.26</td>
</tr>
<tr>
<td>2012</td>
<td>0.16</td>
</tr>
<tr>
<td>2013</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Classification results

- **AlexNet**: 8 layers
  - 2010: 0.28
  - 2011: 0.26
  - 2012: 0.16
  - 2013: 0.12
  - 2014: ?

- **VGGnet**: 19 layers
  - 2010: 0.28
  - 2011: 0.26
  - 2012: 0.16
  - 2013: 0.12
  - 2014: ?

- **GoogLeNet**: 22 layers
  - 2010: 0.28
  - 2011: 0.26
  - 2012: 0.16
  - 2013: 0.12
  - 2014: ?
GoogLeNet: concatenation of inception modules:

28x28x192

256 conv 3x3

28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet: concatenation of inception modules:

28x28x192 -> 256 conv 5x5 -> 28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet: concatenation of inception modules:

28x28x192 → ??? → 28x28x256

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet: concatenation of inception modules:

64 conv 1x1

128 conv 3x3

32 conv 5x5

max-pool 3x3

28x28x256

28x28x192

Too many operations! => simplification using 1x1 conv

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
GoogLeNet

Additional loss layer which injects the gradient inside

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842

image source: http://joelouismarino.github.io/images/blog_images/blog_googlenet_keras/googlenet_diagram.png
GoogLeNet

- 12x fewer parameters than AlexNet
- depth 22 layers
- training: few high-end GPU about a week

Szegedy et al. Going Deeper with Convolutions, CVPR, 2014
https://arxiv.org/abs/1409.4842
Classification results

AlexNet
8 layers
VGGnet
19 layers
GoogLeNet
22 layers

<table>
<thead>
<tr>
<th>Year</th>
<th>Classification Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>0.28</td>
</tr>
<tr>
<td>2011</td>
<td>0.26</td>
</tr>
<tr>
<td>2012</td>
<td>0.16</td>
</tr>
<tr>
<td>2013</td>
<td>0.12</td>
</tr>
<tr>
<td>2014</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Classification results

- **AlexNet**: 8 layers, 2010, Error: 0.28
- **VGGnet**: 19 layers, 2011, Error: 0.26
- **GoogLeNet**: 22 layers, 2012, Error: 0.16
- **ResNet**: 152 layers, 2013, Error: 0.12
- **?**: 2014, Error: 0.07
- **?**: 2015

**Czech Technical University in Prague**

**Faculty of Electrical Engineering, Department of Cybernetics**

30
ResNet

The main idea is as follows:

- deeper ConvNet architectures yielded higher errors.
- error was higher even in training => no overfitting
- problem stems from the optimization (vanishing gradient)

He et al. Going Deeper with Convolutions, CVPR, 2015
• Gradient in deep nets vanishes quickly
• In straightforward conv architecture the weights from the beginning of the net has minor influence on the output !!!
• In backward-pass the gradient of weights in the first layer is computed by multiplication of the all following gradients => prone to diminish!
ResNet: skip connections layer preserve gradient

\[ y = x + f(x) \quad \text{output} \]

He et al. Going Deeper with Convolutions, CVPR, 2015

forward pass

\[ y = x + f(x) \]

He et al. Going Deeper with Convolutions, CVPR, 2015
forward pass

\[ \begin{align*}
  f(x) &\quad \text{input} \\
  \text{conv} &\quad \text{ReLu} \\
  \text{conv} &\quad + \\
  y = x + f(x) &\quad \text{output}
\end{align*} \]

He et al. Going Deeper with Convolutions, CVPR, 2015
He et al. Going Deeper with Convolutions, CVPR, 2015
He et al. Going Deeper with Convolutions, CVPR, 2015
Skip connections partially avoids diminishing gradient
The weights from the beginning of the net has strong influence on the output!
ResNet: deep ConvNet with skip connections

- Competition time about 152 layers ResNet,
- Recently they are able to train 1k layers ResNet
- Initialization with zero weights is meaningful
- Better gradient flow

https://www.kaggle.com/keras/resnet50/home
He et al. Going Deeper with Convolutions, CVPR, 2015
ResNet => DenseNet

Start with multilayer ResNet architecture

• Directly propagate each feature map to all following layers

DenseNet

- Directly propagate each feature map to all following layers
- Improves gradient flow in backward pass

• There exists many “almost independent” paths
• Unravelling of ResNet architecture allows to understand robustness wrt noise and layer removal
Classification results

AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

Human error around 5%
Squeeze and Excitation Networks [Hu et al, CVPR oral, 2017]  

- Winner of ILSVRC 2017
- Enhancement of ResNet, InceptionNet and DenseNet architectures by SE blocks consistently decrease error on ImageNet, COCO, …
Classification results

AlexNet
8 layers

VGGnet
19 layers

GoogLeNet
22 layers

ResNet
152 layers

SE-nets

Classification Error

2010: 0.28
2011: 0.26
2012: 0.16
2013: 0.12
2014: 0.07
2015: 0.036
2016: 0.03
2017: 0.023

16.7% ↓ 23.3% ↓
Summary classification architectures

- It seems that the deeper the better
- ResNet is easy, well-studied architecture=> consider as a starting point
- You should be careful about combining DropOut with BN
  https://arxiv.org/abs/1801.05134
- Capsule networks
Outline

• Architectures of classification networks
• Architectures of segmentation networks
• Architectures of regression networks
• Architectures of detection networks
• Architectures of regression networks
• Architectures of feature matching networks
Semantic segmentation

- road
- sidewalk
- pedestrian
- traffic sign
- trees
- sky
Semantic segmentation

RGB image (HxWx3)

CNN

labels (HxWxN)

- road
- sidewalk
- pedestrian
- traffic sign
- trees
- sky
Semantic segmentation

RGB image (HxWx3)

pixel-wise probability of being road

channel 1

road
sidewalk
pedestrian
traffic sign
trees
sky
Semantic segmentation

RGB image (HxWx3)

CNN

pixel-wise probability of being **sideway**

channel 2

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Semantic segmentation

RGB image (HxWx3)

CNN

pixel-wise probability of being pedestrian

channel 3
Semantic segmentation

RGB image (HxWx3)

CNN

as many output channels as semantic labels

road
sideway
pedestrian
traffic sign
trees
sky
Semantic segmentation

RGB image (HxWx3)

ground truth (0-1 values)
Semantic segmentation

RGB image
(HxWx3)

ground truth (0-1 values)

\[- \log \left( \right) \]
Semantic segmentation

RGB image (HxWx3)

cross-entropy loss

ground truth (0-1 values)

\[ \sum \log \left( \frac{1}{p \cdot (1-p)} \right) \]
Semantic segmentation

- Loss: cross entropy loss summed over all pixels
- Convolution layers:
  - decrease spatial resolution
  - increase number of channels
- Deconvolution layers: exactly opposite

Deconvolution

deconv \[
\begin{bmatrix}
1 & 3 & 0 \\
2 & 0 & 1 \\
0 & 3 & 1 \\
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 1 \\
2 & 0 \\
\end{bmatrix}
\] =

image (3x3) \quad kernel (2x2) \quad output (6x6)
Deconvolution

\[
deconv((1 \ 3 \ 0), (1 \ 1 \ 2 \ 0)) = \frac{1 \times}{1x}
\]

image (3x3) \quad \text{kernel} (2x2) \quad \text{output} (6x6)
Deconvolution

decov \((3x3)\), \(1x\) \(11\) \(1\) \(2\) \(0\) \(0\) \(3\) \(1\) \(2\) \(0\) = \(6x6\)

image \((3x3)\)  kernel \((2x2)\)  output \((6x6)\)
Deconvolution

\[ (\begin{array}{ccc} 1 & 3 & 0 \\ 2 & 0 & 1 \\ 0 & 3 & 1 \end{array}) \text{ deconv } (\begin{array}{cc} 1 & 1 \\ 2 & 0 \end{array}) = (\begin{array}{cc} 1 & 1 \\ 2 & 0 \end{array}) \]

image (3x3) \hspace{1cm} kernel (2x2) \hspace{1cm} output (6x6)
Deconvolution

deconv \begin{pmatrix} 1 & 3 & 0 \\ 2 & 0 & 1 \\ 0 & 3 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}

image (3x3) \hspace{1cm} kernel (2x2) \hspace{1cm} output (6x6)
Deconvolution

\[
\begin{bmatrix}
1 & 3 & 0 \\
2 & 0 & 1 \\
0 & 3 & 1 \\
\end{bmatrix}
\] 
\( \times \)

\[
\begin{bmatrix}
1 & 1 \\
2 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
11 & 3 & 3 \\
20 & 6 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 3 & 3 \\
2 & 0 & 6 & 0 \\
\end{bmatrix}
\]

image (3x3)  kernel (2x2)  output (6x6)
Deconvolution

deconv (  \begin{array}{ccc}
1 & 3 & 0 \\
2 & 0 & 1 \\
0 & 3 & 1 \\
\end{array} , \begin{array}{cc}
1 & 1 \\
2 & 0 \\
\end{array}) = \begin{array}{cccc}
1 & 1 & 3 & 3 \\
2 & 0 & 6 & 0 \\
\end{array}

image (3x3) \hspace{1cm} kernel (2x2) \hspace{1cm} output (6x6)
Deconvolution

deconv (1 3 0
2 0 1
0 3 1), (1 1
2 0) = (1 1 3 3 0 0
2 0 6 0 0 0

image (3x3) kernel (2x2) output (6x6)

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
unpooling

unpool \( \begin{pmatrix} 1 & 3 \\ 2 & 0 \end{pmatrix} \) = \[
\begin{pmatrix}
1 & 1 & 3 & 3 \\
1 & 1 & 3 & 3 \\
2 & 2 & 0 & 0 \\
2 & 2 & 0 & 0
\end{pmatrix}
\]

image (2x2) output (4x4)
max-unpooling

\[
\text{max-unpool}( \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix} ) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\]

image (2x2) output (4x4)

remember position of the maximum from max-pooling layer
• Replace maxpooling by Atrous Convolution
• Replace deconvolutions by bi-linear interp+CRF

DeepLab v3 - results

(a) Image  
(b) G.T.  
(c) Before CRF  
(d) After CRF
DeepLab v3 - results

(a) Image

(b) G.T.

(c) Before CRF

(d) After CRF
DeepLab v3 - results

(a) Image
(b) G.T.
(c) Before CRF
(d) After CRF
DeepLab v3 - results

(a) Image    (b) G.T.    (c) Before CRF    (d) After CRF

CRF failure cases
DeepLab v3 - summary

- significantly outperforms state-of-the-art on several datasets
- CRF improves mIOU about 2%
- ASPP improves mIOU about 3%
- codes available: https://github.com/tensorflow/models/tree/master/research/deeplab
Outline

• Architectures of classification networks
• Architectures of segmentation networks
• Architectures of regression networks
• Architectures of detection networks
• Architectures of regression networks
• Architectures of feature matching networks
Pose regression baseline

ConvNet directly estimates joint positions (2xN real numbers)

Straightforward learning directly minimize L2 loss over all joint positions (2D/3D).

$J_k$: Joint

Integral Human Pose Regression [Sun ECCV 2018]

Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- ConvNet first estimates N joint’s heat maps $H_k$, $k = 1 \ldots N$ (i.e. N 2D-images or N 3D-arrays)
- Learning minimizes segmentation loss over the N images

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as position of heatmap maximum

\[ H_k: \text{Heatmap} \]

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as position of heatmap maximum

\[ H_k: \text{Heatmap} \]

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research [https://arxiv.org/abs/1711.08229]
Pose detection baseline

• estimate joint position as position of heatmap maximum

\[ J_k = \arg \max_p H_k(p) \]

Integral Human Pose Regression [Sun ECCV 2018]  
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as position of heatmap maximum

\[ J_k = \arg \max_p H_k(p) \]

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as expected value in heatmap $H_k$

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as expected value in heatmap

\[ J_k = \int_{p \in \Omega} p \cdot \hat{H}_k(p) \]

Integral Human Pose Regression [Sun ECCV 2018]  
Microsoft Research https://arxiv.org/abs/1711.08229
Pose detection baseline

- estimate joint position as expected value in heatmap

\[ J_k = \int_{p \in \Omega} p \cdot \tilde{H}_k(p) \]

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research https://arxiv.org/abs/1711.08229

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Pose detection+regression baseline

- ConvNet first estimates N joint’s heat maps $H_k$, $k = 1 \ldots N$ (i.e. N 2D-images or N 3D-arrays)
- Learning minimizes segmentation loss over the N images
- Joints positions $(p)$ estimated as expected value in $H_k$

Integral Human Pose Regression [Sun ECCV 2018]
Microsoft Research [https://arxiv.org/abs/1711.08229]
PoseTrack challenge (ICCV 2017/ECCV 2018)
https://posetrack.net
Pose regression references

• PoseTrack benchmark a datasets
  https://posetrack.net

• Guler et al. (Facebook Research), DensePose
  https://arxiv.org/abs/1802.00434
  https://github.com/facebookresearch/Densepose

• Realtime Multi-Person 2D Human Pose Estimation using Part Affinity Fields, CVPR 2017 Oral
  https://www.youtube.com/watch?v=pW6nZxeWlGM

• Integral Human Pose Regression [Sun ECCV 2018]
  Microsoft Research
  https://arxiv.org/abs/1711.08229
Outline

• Architectures of classification networks
• Architectures of segmentation networks
• Architectures of regression networks
• Architectures of detection networks
• Architectures of feature matching networks
Object detection
Object detection

car

person

car

car
Object detection
Object detection

class: person

CNN

0.7
0.1
0.2
0.0
Object detection
Object detection

class: car

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Object detection

CNN

0.0

0.9

0.1

0.0
Object detection

class: background

CNN

[0.0, 0.1, 0.0, 0.9]
Object detection

classify all rectangles
Object detection

• Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  \[ H \times W \times \text{Aspect\_Ratio} \times \text{Scales} \times 0.001 \text{ sec} = \text{months} \]
Object detection

CNN

classify all rectangles
Object detection

 classify + align only 2k region proposals

https://arxiv.org/abs/1504.08083
Object detection

• Approach works but it takes extremely long to compute response on all rectangular sub-windows:
H x W x Aspect_Ratio x Scales x 0.001 sec = months

• Instead we can use elementary signal processing method to extract only 2k viable candidates:
[Girschick ICCV 2015], Fast-RCNN
https://arxiv.org/abs/1504.08083
(find 2k cand.) + (2k cand. x 0.001 sec) = 47+2 sec = 49 sec
Object detection

The search for region proposals is computational bottleneck!!!

https://arxiv.org/abs/1504.08083
Object detection

region proposal net
(output: 2k proposals)

Region Proposal Net (RPN)

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Region Proposal Net (RPN)

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Faster-RCNN https://arxiv.org/abs/1506.01497
Region Proposal Net (RPN)

- generate bounding which corresponds to discrete positions in low resolution feature maps and measure IoU

Region Proposal Net (RPN)

- Generate bounding boxes that correspond to discrete positions in low resolution feature maps and measure IoU.
- Bbs with IoU > 0.7 are objects, bbs with IoU < 0.3 are not objects.

Region Proposal Net (RPN)

- for each discrete bb RPN predicts:
  - its “alignment with gt” (regression loss)
  - its “objectness” (classification loss)

Region Proposal Net (RPN)

• for each discrete bb RPN predicts:
  • its “alignment with gt” (regression loss)
  • its “objectness” (classification loss)


Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Object detection

region proposal net
(output: 2k proposals)

Save computational power by reusing RPN feature maps

Faster-RCNN https://arxiv.org/abs/1506.01497
Object detection

region proposal net
(output: 2k proposals)

Save computational power by reusing RPN feature maps

Object detection

- Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  \[ H \times W \times \text{Aspect Ratio} \times \text{Scales} \times 0.001 \text{ sec} = \text{months} \]
- Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083
  \[ \text{find 2k cand.} + (2k \text{ cand. } \times 0.001 \text{ sec}) = 47+2 \text{ sec} = 49 \text{ sec} \]
- Do region proposal by CNN => \( 0.3 + 2 = 2.3 \text{ sec} \)
Object detection

region proposal net
(output: 2k proposals)

[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870

classification
+ alignment
+ segmentation mask
+ pose regression
[He et al CVPR 2017] Mask-RCNN
https://arxiv.org/abs/1703.06870
Mask RCNN - applications

Playground detection for OpenStreetMaps

https://github.com/jremillard/images-to-osm
Mask RCNN - applications
Mapping challenge

https://github.com/crowdAI/crowdai-mapping-challenge-mask-rcnn
Mask RCNN - applications

Detection and segmentation for surgery robot
Mask RCNN - applications

Distinguishing nuclei in microscopy images

https://github.com/matterport/Mask_RCNN/tree/master/samples/nucleus
Mask RCNN - applications

https://github.com/huuuuusy/Mask-RCNN-Shiny

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Object detection

• Approach works but it takes extremely long to compute response on all rectangular sub-windows:
  \[ H \times W \times \text{Aspect Ratio} \times \text{Scales} \times 0.001 \text{ sec} = \text{months} \]

• Instead we can use elementary signal processing method to extract only 2k viable candidates:
  [Girschick ICCV 2015], Fast-RCNN
  https://arxiv.org/abs/1504.08083
  \[(\text{find 2k cand.}) + (2k \text{ cand.} \times 0.001 \text{ sec}) = 47+2 \text{ sec} = 49 \text{ sec}\]

• Do region proposal by CNN => \[0.3 + 2 = 2.3 \text{ sec}\]

• Similar idea but more efficient implementation YOLO/SSD:
  about \[0.2 \text{ sec}\]
Deep convolutional - object detection

YOLO v2

http://pureddie.com/yolo
Outline

• Architectures of classification networks
• Architectures of segmentation networks
• Architectures of regression networks
• Architectures of detection networks
• Architectures of feature matching networks
Input: RGB image
Output: set of detected feature points with descriptors

Descriptor is vector which is:
• similar for corresponding points
• and dissimilar for not corresponding points.
Segmentation CNN for pixel-wise two-class labelling

- class 1: “suitable feature point”
- class 2: “unsuitable feature point”
LIFT: Learnable Invariant Feature Descriptors

Score map
Probability map

Expected value of x and y under the prob. distr.
LIFT: Learnable Invariant Feature Descriptors

\[U\]

\[
\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array}
\]

\[V\]

\[
\begin{array}{cccc}
3 & 3 & 2 & 2 \\
3 & 3 & 2 & 2 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

\[m = 1, n = 1\]
convolution with $\kappa(m, n) =>$ differentiable!

$$\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array}$$

$$\begin{array}{cccc}
\kappa(m, n) & & & \\
\otimes & & & \\
\begin{array}{cccc}
& & & \\
& & & \\
& & & \\
& & & \\
\end{array} \\
\end{array}$$

$$\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array} \times \begin{array}{cccc}
& & & \\
& & & \\
& & & \\
& & & \\
\end{array} = \begin{array}{cccc}
& & & \\
& & & \\
& & & \\
& & & \\
\end{array}$$

$LIFT: \text{Learnable Invariant Feature Descriptors}$

[Yi et al ECCV 2016] [https://arxiv.org/abs/1603.09114]
LIFT: Learnable Invariant Feature Descriptors

$U = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 2 & 3 & 2 & 1 \\ 0 & 1 & 0 & 3 \\ 1 & 3 & 1 & 2 \end{bmatrix}$

\[ \kappa(m, n) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \]

$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$V = \begin{bmatrix} \color{red}3 \\ 1 \end{bmatrix}$
LIFT: Learnable Invariant Feature Descriptors

\[ U \times \kappa(m, n) = V \]

\[ n = 1 \]

\[ m = 2 \]
LIFT: Learnable Invariant Feature Descriptors

\[
\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array} \otimes \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} = \begin{array}{cccc}
3 & 3 & 3 & 1 \\
\end{array}
\]

\[m = 3, \quad n = 1\]
LIFT: Learnable Invariant Feature Descriptors

\[ \kappa(m, n) \]

\[ U \]

\[ \begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array} \]

\[ V \]

\[ \begin{array}{cccc}
3 & 3 & 3 & 3 \\
1 & 1 & 1 & 1 \\
\end{array} \]

\[ n = 1 \]

\[ m = 4 \]
LIFT: Learnable Invariant Feature Descriptors

\[
\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} \times
\begin{array}{cccc}
3 & 3 \\
3 & 3 \\
1 & 1 \\
1 & 1 \\
\end{array}
= \begin{array}{cccc}
m = 4 \\
3 & 3 \\
3 & 3 \\
1 & 1 \\
1 & 1 \\
\end{array}
\]

n = 2
LIFT: Learnable Invariant Feature Descriptors

\[ U \]:

\[
\begin{array}{cccc}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\] \times \kappa(m, n)

\[
\begin{array}{cccc}
3 & 3 & 0 & 0 \\
3 & 3 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

\[ n = 2 \]

\[ m = 4 \]
LIFT: Learnable Invariant Feature Descriptors

\[\begin{bmatrix} 1 & 0 & 0 & 2 \\ 2 & 3 & 2 & 1 \\ 0 & 1 & 0 & 3 \\ 1 & 3 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 2 & 2 \\ 3 & 3 & 2 & 2 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \]

\( U \times \kappa(m, n) = V \)
LIFT: Learnable Invariant Feature Descriptors

Can we translate image $U$ by 1/2 pixel?

$$
\begin{bmatrix}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{bmatrix}
\times
\begin{bmatrix}
\kappa(m, n) \\
\end{bmatrix}
= 
\begin{bmatrix}
\end{bmatrix}$$

$m = 1$

$n = 1$

$V$
LIFT: Learnable Invariant Feature Descriptors

Can we translate image U by 1/2 pixel?

\[
U \begin{bmatrix}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{bmatrix} \quad \times \quad \begin{bmatrix}
0.5 & 0.5 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.5 & 0.5 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
V \begin{bmatrix}
0.5 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]
LIFT: Learnable Invariant Feature Descriptors

What about rotation?

\[
\begin{pmatrix}
1 & 0 & 0 & 2 \\
2 & 3 & 2 & 1 \\
0 & 1 & 0 & 3 \\
1 & 3 & 1 & 2 \\
\end{pmatrix} \otimes \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0.3 & 0.3 & 0 & 0 \\
0.2 & 0.2 & 0 & 0 \\
\end{pmatrix} = \begin{pmatrix}
& & & \\
& & & \\
& & & \\
1.1 & & & \\
& & & \\
& & & \\
& & & \\
\end{pmatrix}
\]

\[
U \Rightarrow \text{LIFT pipeline} \Rightarrow \begin{pmatrix}
\text{DET} & \rightarrow & \text{softargmax} & \rightarrow & \text{Crop} & \rightarrow & \text{ORI} & \rightarrow & \text{Rot} & \rightarrow & \text{DESC} & \rightarrow & \text{description vector}
\end{pmatrix}
\]
LIFT: Learnable Invariant Feature Descriptors

Spatial Transformer Networks

\[ V_i = \sum_n \sum_m U_{nm} \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \kappa(m, n) \]

Bilinear approximation of affine transformation is differentiable!

LIFT: Learnable Invariant Feature Descriptors

- Trained in end-to-end manner
- Ground truth correspondences for training obtained from SfM and webcams
- Training set consists of four-touples:

  Two corresponding patches on distinctive points
  One not corresponding patch on a distinctive point
  One patch on a not distinctive point

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
LIFT: Learnable Invariant Feature Descriptors

- All patches are fed into the network and differentiable loss
- Loss makes:
  - \(d_1\) and \(d_2\) as close as possible,
  - \(d_3\) as far as possible (from \(d_1\) and \(d_2\))
  - DET to have high response on \(p_1, p_2, p_3\) and small on \(p_4\)
Summary architectures

• Deeper architectures, with many small kernels with skip-connections (e.g. ResNet, DenseNet) seems reasonable
• Decreasing the spatial resolution while increasing spatial resolution allows to exploit context.
• Atrous spatial pyramid seems to be viable replacement for max-pooling
• Argmax is not differentiable, but it can be replaced by expected value.
• Any affine transformation can be tackled by Spatial Transform Layer
• Divide and Conquer strategy with as many as possible auxiliary losses seems to work well on many problems
• A lot of dark-magic needed for successful training