Basic CNN Architecture Review
AlexNet
Inception
ResNet
(a) Image classification

(b) Object localization

(c) Semantic segmentation

(d) Instance segmentation
Segmentation
<table>
<thead>
<tr>
<th>Name and Reference</th>
<th>Purpose</th>
<th>Year</th>
<th>Classes</th>
<th>Data</th>
<th>Resolution</th>
<th>Sequence</th>
<th>Synthetic/Real</th>
<th>Samples (training)</th>
<th>Samples (validation)</th>
<th>Samples (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASCAL VOC 2012 Segmentation [27]</td>
<td>Generic</td>
<td>2012</td>
<td>21</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>1464</td>
<td>1449</td>
<td>Private</td>
</tr>
<tr>
<td>PASCAL-Context [28]</td>
<td>Generic</td>
<td>2014</td>
<td>540 (59)</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>10103</td>
<td>N/A</td>
<td>9637</td>
</tr>
<tr>
<td>PASCAL-Part [29]</td>
<td>Generic-Part</td>
<td>2014</td>
<td>20</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>10103</td>
<td>N/A</td>
<td>9637</td>
</tr>
<tr>
<td>SSD [30]</td>
<td>Generic</td>
<td>2011</td>
<td>21</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>8498</td>
<td>2857</td>
<td>N/A</td>
</tr>
<tr>
<td>Microsoft COCO [31]</td>
<td>Generic</td>
<td>2014</td>
<td>80</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>82783</td>
<td>40504</td>
<td>81434</td>
</tr>
<tr>
<td>SYNTHIA [32]</td>
<td>Urban (Driving)</td>
<td>2016</td>
<td>11</td>
<td>2D</td>
<td>960 × 720</td>
<td>✓</td>
<td>S</td>
<td>13407</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Cityscapes (fine) [33]</td>
<td>Urban</td>
<td>2015</td>
<td>30 (8)</td>
<td>2D</td>
<td>2048 × 1024</td>
<td>✓</td>
<td>R</td>
<td>2975</td>
<td>500</td>
<td>1525</td>
</tr>
<tr>
<td>Cityscapes (coarse) [33]</td>
<td>Urban</td>
<td>2015</td>
<td>30 (8)</td>
<td>2D</td>
<td>2048 × 1024</td>
<td>✓</td>
<td>R</td>
<td>22973</td>
<td>500</td>
<td>N/A</td>
</tr>
<tr>
<td>CamVid [34]</td>
<td>Urban (Driving)</td>
<td>2009</td>
<td>32</td>
<td>2D</td>
<td>960 × 720</td>
<td>✓</td>
<td>R</td>
<td>701</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Urban (Driving)</td>
<td>2009</td>
<td>11</td>
<td>2D</td>
<td>960 × 720</td>
<td>✓</td>
<td>R</td>
<td>367</td>
<td>100</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>KITTI-Layout [36] [37]</td>
<td>Urban</td>
<td>2012</td>
<td>3</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>323</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>KITTI-Ros [38]</td>
<td>Urban</td>
<td>2015</td>
<td>11</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>170</td>
<td>N/A</td>
<td>46</td>
</tr>
<tr>
<td>Stanford background [40]</td>
<td>Outdoor</td>
<td>2009</td>
<td>8</td>
<td>2D</td>
<td>320 × 240</td>
<td>✓</td>
<td>R</td>
<td>725</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SfMFlow [41]</td>
<td>Outdoor</td>
<td>2011</td>
<td>33</td>
<td>2D</td>
<td>256 × 256</td>
<td>✓</td>
<td>R</td>
<td>2688</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Youtube-Objects-Jain [42]</td>
<td>Objects</td>
<td>2014</td>
<td>10</td>
<td>2D</td>
<td>480 × 360</td>
<td>✓</td>
<td>R</td>
<td>10167</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Adobe’s Portrait Segmentation [26]</td>
<td>Portrait</td>
<td>2016</td>
<td>2</td>
<td>2D</td>
<td>600 × 800</td>
<td>✓</td>
<td>R</td>
<td>1500</td>
<td>300</td>
<td>N/A</td>
</tr>
<tr>
<td>MNC [43]</td>
<td>Materials</td>
<td>2015</td>
<td>23</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>7061</td>
<td>5000</td>
<td>N/A</td>
</tr>
<tr>
<td>DAVIS [44] [45]</td>
<td>Indoor</td>
<td>2012</td>
<td>40</td>
<td>2D</td>
<td>480 × 640</td>
<td>✓</td>
<td>R</td>
<td>4219</td>
<td>2023</td>
<td>2180</td>
</tr>
<tr>
<td>NYUv2 [46]</td>
<td>Indoor</td>
<td>2013</td>
<td>–</td>
<td>2D</td>
<td>640 × 480</td>
<td>✓</td>
<td>R</td>
<td>795</td>
<td>654</td>
<td>N/A</td>
</tr>
<tr>
<td>SUN3D [47]</td>
<td>Indoor</td>
<td>2015</td>
<td>37</td>
<td>2D</td>
<td>Variable</td>
<td>✓</td>
<td>R</td>
<td>19640</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SUNRGBD [48]</td>
<td>Indoor</td>
<td>2011</td>
<td>51</td>
<td>2D</td>
<td>640 × 480</td>
<td>✓</td>
<td>R</td>
<td>207920</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>RGB-D Object Dataset [49]</td>
<td>Household objects</td>
<td>2016</td>
<td>16/20</td>
<td>3D</td>
<td>N/A</td>
<td>✓</td>
<td>R</td>
<td>31,963</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Stanford 2D-3D-S [51]</td>
<td>Indoor</td>
<td>2017</td>
<td>13</td>
<td>2D/2.3D/3D</td>
<td>1080 × 1080</td>
<td>✓</td>
<td>R</td>
<td>70469</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3D Mesh [52]</td>
<td>Object/Part</td>
<td>2009</td>
<td>19</td>
<td>3D</td>
<td>N/A</td>
<td>✓</td>
<td>S</td>
<td>380</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Sydney Urban Objects Dataset [53]</td>
<td>Urban (Objects)</td>
<td>2013</td>
<td>26</td>
<td>3D</td>
<td>N/A</td>
<td>✓</td>
<td>R</td>
<td>41</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Large-Scale Point Cloud Classification Benchmark [54]</td>
<td>Urban/Nature</td>
<td>2016</td>
<td>8</td>
<td>3D</td>
<td>N/A</td>
<td>✓</td>
<td>R</td>
<td>15</td>
<td>N/A</td>
<td>15</td>
</tr>
</tbody>
</table>
FCN
SegNet

Input
RGB Image

Convolutional Encoder-Decoder
Pooling Indices

Output
Segmentation

Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
DILATED CONVOLUTIONS

(a)  
(b)  
(c)
Multi-scale CNN
ReSeg net.
Instance Segmentation
Deep Watershed Transform

\[
l_{\text{direction}} = \sum_{p \in P_{\text{obj}}} w_p \left( \frac{1}{2} \cos^{-1} \langle \bar{u}_p^{\text{GT}}, \bar{u}_p^{\text{pred}} \rangle \right)^2
\]

\[
l_{\text{watershed}} = \sum_{p \in P_{\text{obj}}} \sum_{k=1}^{K} w_p c_k (t_{p,k} \log \bar{y}_{p,k} + t_{p,k} \log y_{p,k})
\]
Detection
RCNN

R-CNN: Regions with CNN features

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions
\[
\lambda_{\text{coord}} \sum_{i=0}^{B} \sum_{j=0}^{B} \left( \sqrt{\hat{x}_i - x_i} \right)^2 + \left( \sqrt{\hat{y}_i - y_i} \right)^2 \\
+ \lambda_{\text{coord}} \sum_{i=0}^{B} \sum_{j=0}^{B} \left( \sqrt{\hat{w}_i - w_i} \right)^2 + \left( \sqrt{\hat{h}_i - h_i} \right)^2 \\
+ \sum_{i=0}^{B} \sum_{j=0}^{B} \left( \hat{c}_i - c_i \right)^2 \\
+ \lambda_{\text{noobj}} \sum_{i=0}^{B} \sum_{j=0}^{B} \left( \hat{c}_i - c_i \right)^2 \\
+ \sum_{i=0}^{B} \sum_{c \in \text{classes}} \left( \hat{p}_i(c) - p_i(c) \right)^2 \quad (3)
\]
Region Proposal Networks

2k scores

4k coordinates

cls layer

reg layer

256-d

intermediate layer

sliding window

conv feature map

k anchor boxes
Faster RCNN

R-CNN: Regions with CNN features

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Object is a cat
Bounding-box regression loss
Classification loss

Object or not object
BB proposal
Classification loss
Bounding-box regression loss

Region Proposal Network
feature map
Last conv layer
Rol pooling
pre-train image-net
CNN
VGG

focal Systems
Spatial Pyramid Pooling

fully-connected layers (fc₆, fc₇)

fixed-length representation

16×256-d

4×256-d

256-d

spatial pyramid pooling layer

feature maps of conv₅ (arbitrary size)

convolutional layers

input image
ROI Pooling

Hi-res input image:
3 x 800 x 600
with region proposal

Hi-res conv features:
C x H x W
with region proposal

Divide projected region into h x w grid
FPNs

(a) Featurized image pyramid
(b) Single feature map
(c) Pyramidal feature hierarchy
(d) Feature Pyramid Network

Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.
RetinaNet
Mask R-CNN

Figure 1. The **Mask R-CNN** framework for instance segmentation.

Faster R-CNN
Caffe: https://github.com/rbgirshick/py-faster-rcnn
PyTorch: https://github.com/longcw/faster_rcnn_pytorch
MatLab: https://github.com/ShaoqingRen/faster_rcnn

Mask R-CNN
PyTorch: https://github.com/felixgwu/mask_rcnn_pytorch
TensorFlow: https://github.com/CharlesShang/FastMaskRCNN
IF WE HAVE TIME:

Discussion on evaluation
Intersection over Union

- **Pixel Accuracy (PA):** It is the simplest metric, simply computing a ratio between the amount of properly classified pixels and the total number of them.

\[
PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}}
\]

- **Mean Pixel Accuracy (MPA):** A slightly improved PA in which the ratio of correct pixels is computed in a per-class basis and then averaged over the total number of classes.

\[
MPA = \frac{1}{k+1} \sum_{i=0}^{k} p_{ii} \frac{1}{\sum_{j=0}^{k} p_{ij}}
\]

\[
MIOU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}
\]
<table>
<thead>
<tr>
<th></th>
<th>Y_hat = True</th>
<th>Y_hat = False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = True</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Y = False</td>
<td>450</td>
<td>450</td>
</tr>
</tbody>
</table>
Atrous Models in Semantic Segmentation
Why Atrous?
Let’s remove the fully connected layers and do segmentation.

Problem?
Output Size is too small!
So let's remove all max pools!

What's the problem?
Receptive Field is too small!

So let’s stride the convolutions by 2!
Size Problem again!

Solution: Atrous
Atrous VGG

Can still classify =>
Size invariance w/ Atrous: Atrous-Ception
F1 Loss
F1 Score:

\[
PRE = \frac{TP}{TP + FP}
\]

\[
REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}
\]

\[
F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}
\]
Binary Crossentropy Loss

\[ L(w) = \frac{1}{N} \sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N} \sum_{n=1}^{N} \left[ y_n \log \hat{y}_n + (1 - y_n) \log (1 - \hat{y}_n) \right], \]
Let’s Optimize for the right thing!
y = labels, y’ = predictions

TP = \text{sum}(y(*)y')

TP + FP = \text{sum}(y')

TP + FN = \text{sum}(y)

Loss = -F1
Results:

Cross entropy

- F1

GT