Fast scene understanding and prediction for autonomous platforms

Bert De Brabandere, KU Leuven, October 2017
Who am I?

- MSc in Electrical Engineering at KU Leuven, Belgium
  - One year at ETH Zurich

- Last year PhD student with Luc Van Gool at KU Leuven
  - Fast Scene Understanding for Autonomous Platforms
  - Collaboration with Toyota

- Currently intern at Deepmind
  - Working on meta-learning
  - Until end of November
Outline

- Dynamic filter networks for video prediction
- Semantic instance segmentation with a discriminative loss function
- Real-time scene understanding: integrating components and ongoing work
Dynamic Filter Networks

Bert De Brabandere*, Xu Jia*, Tinne Tuytelaars, Luc Van Gool
Motivation
Dynamic Filter Networks

- **Standard network:**
  - filters (layer weights) are updated during training, stay fixed during inference
Dynamic Filter Networks

- **Standard network:**
  - filters (layer weights) are updated during training, stay fixed during inference

- **Dynamic Filter Network:**
  - filters are generated dynamically by another network during inference
Standard Convolution
Dynamic Convolution

Input A

Filter-generating network

Input B

Output
Proof-of-concept: learning steerable filters

$\theta = 45^\circ$
Standard Local Filtering

= “convolution with unshared weights”
Dynamic Local Filtering

Input A ➔ Filter-generating network ➔ Output

Input ➔ Input B ➔ Output
Video Prediction
Video pixel networks: 87.6
Stereo Prediction
Visualizing the filters

video prediction

input

filters

prediction

ground truth

stereo prediction
Related work

- Evolving modular fast-weight networks for control (Gomez & Schmidhuber)
- Spatial Transformer Networks (Jaderberg et al.)
- Spatio-temporal video autoencoder with differentiable memory (Patraucean et al.)
- Image question answering using convnet with dynamic parameter prediction (Noh et al.)
- A dynamic convolutional layer for short range weather prediction (Klein et al.)
- HyperNetworks (Ha et al.)
- Video Pixel Networks (Kalchbrenner et al.)
- Learning feed-forward one-shot learners (Bertinetto et al., NIPS 2016)
- Unsupervised learning for physical interaction through video prediction (Finn et al., NIPS 2016)
- Dynamic Steerable Blocks in Deep Residual Networks (Jacobsen et al.)
Summary

- **Traditional network**: filters stay fixed during inference

- **Dynamic Filter Network**: filters dynamically generated by another network
  - Dynamic Convolution
  - Dynamic Local Filtering (= “convolution with unshared weights”)

- **Experiments**
  - Learning steerable filters
  - Video prediction
  - Stereo prediction
  - Visualizing the filters

code available at github.com/dbbert/dfn
Semantic Instance Segmentation with a Discriminative Loss Function

Bert De Brabandere*, Davy Neven*, Luc Van Gool
Semantic Instance Segmentation

= Generate segmentation mask and label for each individual object in the scene
Most popular approach: detect-and-segment

Mask R-CNN (He et al., 2017)
Instance-aware semantic segmentation via multi-task network cascades (Dai et al., 2016)
Shape-aware instance segmentation (Hayder et al., 2016)
... and many more
But how to do these?
Our method

1. Embed each pixel with a deep network such that:
   ○ Pixel embeddings of same instance lie close together
   ○ Pixel embeddings of different instances lie far apart
Our method

1. Embed each pixel with a deep network such that:
   - Pixel embeddings of same instance lie close together
   - Pixel embeddings of different instances lie far apart

2. Cluster the pixel embeddings
The formulas

\[ L_{var} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} \left[ \left\| \mu_c - x_i \right\| - \delta_v \right]_+^2 \quad (1) \]

\[ L_{dist} = \frac{1}{C(C-1)} \sum_{c_A=1}^{C} \sum_{c_B=1}^{C} \sum_{c_A \neq c_B} \left[ 2\delta_d - \| \mu_{c_A} - \mu_{c_B} \| \right]_+^2 \quad (2) \]

\[ L_{reg} = \frac{1}{C} \sum_{c=1}^{C} \| \mu_c \| \quad (3) \]

\[ L = \alpha \cdot L_{var} + \beta \cdot L_{dist} + \gamma \cdot L_{reg} \quad (4) \]
Training visualization
Clustering at test time

1. Send input image through network to generate pixel embeddings

2. Pick a random pixel embedding

3. Select all pixel embeddings that lie within a margin $\delta_v$ from it
   - = fast thresholding operation
   - These pixels belong to one instance
   - We can do mean-shift iterations in this step for robustness

4. Go back to 2, but only pick from the pixels that are not yet labeled
Results on CVPPP plant leaf dataset
Results on CVPPP plant leaf dataset

| Method               | SBD  | $|D_iC|$ |
|----------------------|------|-------|
| RIS+CRF [31]         | 66.6 | 1.1   |
| MSU [32]             | 66.7 | 2.3   |
| Nottingham [32]      | 68.3 | 3.8   |
| Wageningen [44]      | 71.1 | 2.2   |
| IPK [25]             | 74.4 | 2.6   |
| PRIAn [15]           | -    | 1.3   |
| End-to-end [30]      | 84.9 | 0.8   |
| Ours                 | 84.2 | 1.0   |

Table 1. Segmentation and counting performance on the test set of the CVPPP leaf segmentation challenge.
Results on Cityscapes
## Results on Cityscapes

<table>
<thead>
<tr>
<th>name</th>
<th>fine</th>
<th>coarse</th>
<th>16-bit</th>
<th>depth</th>
<th>video</th>
<th>sub</th>
<th>AP &gt; 50k</th>
<th>AP &gt; 100m</th>
<th>AP &gt; 50m</th>
<th>Runtime [s]</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask R-CNN [COCO]</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>32.0</td>
<td>58.1</td>
<td>45.8</td>
<td>49.5</td>
<td>n/a</td>
</tr>
<tr>
<td>Mask R-CNN [fine-only]</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>26.2</td>
<td>49.9</td>
<td>37.6</td>
<td>40.1</td>
<td>n/a</td>
</tr>
<tr>
<td>SGN</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>25.0</td>
<td>44.9</td>
<td>38.9</td>
<td>44.5</td>
<td>n/a</td>
</tr>
<tr>
<td>Pixelwise Instance Segmentation with a Dynamically Instantiated Network</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>20.0</td>
<td>38.8</td>
<td>32.6</td>
<td>37.6</td>
<td>n/a</td>
</tr>
<tr>
<td>Deep Watershed Transformation</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>19.4</td>
<td>35.3</td>
<td>31.4</td>
<td>36.8</td>
<td>n/a</td>
</tr>
<tr>
<td>Semantic Instance Segmentation with a Discriminative Loss Function</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>17.5</td>
<td>35.9</td>
<td>27.8</td>
<td>31.0</td>
<td>yes</td>
</tr>
<tr>
<td>Boundary-aware Instance Segmentation</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>17.4</td>
<td>36.7</td>
<td>29.3</td>
<td>34.0</td>
<td>n/a</td>
</tr>
<tr>
<td>InstanceCut</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>13.0</td>
<td>27.9</td>
<td>22.1</td>
<td>26.1</td>
<td>n/a</td>
</tr>
<tr>
<td>Joint Graph Decomposition and Node Labeling</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>9.8</td>
<td>23.2</td>
<td>16.8</td>
<td>20.3</td>
<td>n/a</td>
</tr>
<tr>
<td>RecAttend</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>9.5</td>
<td>18.9</td>
<td>16.8</td>
<td>20.9</td>
<td>n/a</td>
</tr>
<tr>
<td>Pixel-level Encoding for Instance Segmentation</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>8.9</td>
<td>21.1</td>
<td>15.3</td>
<td>16.7</td>
<td>n/a</td>
</tr>
<tr>
<td>R-CNN + MCG convex hull</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>4.6</td>
<td>12.9</td>
<td>7.7</td>
<td>10.3</td>
<td>60.0</td>
</tr>
<tr>
<td>Instance-level Segmentation of Vehicles by Deep Contours</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>2.3</td>
<td>3.7</td>
<td>3.9</td>
<td>4.9</td>
<td>0.2</td>
</tr>
</tbody>
</table>
But how to do these?
Scattered sticks toy dataset
### Speed-accuracy trade-off by plugging in different network architectures

<table>
<thead>
<tr>
<th></th>
<th>Dim</th>
<th>AP</th>
<th>AP&lt;sub&gt;gt&lt;/sub&gt;</th>
<th>fps</th>
<th>#p</th>
<th>mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENet [27]</td>
<td>512 x 256</td>
<td>0.19</td>
<td>0.21</td>
<td>145</td>
<td>0.36</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>768 x 384</td>
<td>0.21</td>
<td>0.25</td>
<td>94</td>
<td>1.03</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>1024 x 512</td>
<td>0.20</td>
<td>0.26</td>
<td>61</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td>Segnet [6]</td>
<td>512 x 256</td>
<td>0.20</td>
<td>0.22</td>
<td>27</td>
<td>1.22</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>768 x 384</td>
<td>0.22</td>
<td>0.26</td>
<td>14</td>
<td>29.4</td>
<td>1.47</td>
</tr>
<tr>
<td></td>
<td>1024 x 512</td>
<td>0.18</td>
<td>0.24</td>
<td>8</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>Dilation [45]</td>
<td>512 x 256</td>
<td>0.21</td>
<td>0.24</td>
<td>15</td>
<td>2.20</td>
<td>2.64</td>
</tr>
<tr>
<td></td>
<td>768 x 384</td>
<td>0.24</td>
<td>0.29</td>
<td>6</td>
<td>2.20</td>
<td>3.27</td>
</tr>
<tr>
<td></td>
<td>1024 x 512</td>
<td>0.23</td>
<td>0.30</td>
<td>4</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>Resnet38 [40]</td>
<td>512 x 256</td>
<td>0.24</td>
<td>0.27</td>
<td>12</td>
<td>4.45</td>
<td>8.83</td>
</tr>
<tr>
<td></td>
<td>768 x 384</td>
<td>0.29</td>
<td>0.34</td>
<td>5</td>
<td>8.83</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Average Precision (AP), AP using gt segmentation labels (AP<sub>gt</sub>), speed of forward pass (fps), number of parameters ($\times 10^6$) and memory usage (GB) for different models evaluated on the car class of the Cityscapes validation set.
The bad news

- Doesn’t seem to work well on datasets where diverse objects can appear in random constellations, e.g. Pascal VOC and MS COCO
  - Detection-based approach with non-max suppression more suited than holistic approach?
The bad news

- Doesn't seem to work well on datasets where diverse objects can appear in random constellations, e.g. Pascal VOC and MS COCO
  - Detection-based approach with non-max suppression more suited than holistic approach?

- Problem with field-of-view in networks that are not deep enough
Some videos
Real-time scene understanding: integrating components
Integrating different components

- Parts of the network can be shared
  - Increases speed
  - Reduces GPU memory
  - Increases performance of individual tasks

- Can process 1024x512 images at ~20fps on a single GPU
Depth estimation

- **Goal**: predict a dense depth map of the scene

- **From stereo**:
  - Semi-global matching (SGM)
  - For the front stereo camera

- **From a single image**:
  - Deep network
  - For the side mono camera’s
Sharing the network between tasks or not?

<table>
<thead>
<tr>
<th></th>
<th>semantic</th>
<th>instance</th>
<th>depth</th>
<th>mem</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained separately</td>
<td>58.3%</td>
<td>0.20%</td>
<td>9.2m</td>
<td>2.6 GB</td>
<td>12 fps</td>
</tr>
<tr>
<td>Trained together</td>
<td>59.3%</td>
<td>0.21%</td>
<td>7.5m</td>
<td>1.2 GB</td>
<td>21 fps</td>
</tr>
</tbody>
</table>

Fast scene understanding for autonomous driving. B. De Brabandere, D. Neven, M. Proesmans, L.Van Gool.
Thank you!