Structured Prediction using Convolutional Neural Networks

Bohyung Han
bhhan@postech.ac.kr
Computer Vision Lab.

Overview

- Convolutional Neural Networks (CNNs)
- Structured predictions for low-level computer vision
  - Image denoising
  - Super-resolution
- Deconvolutions for structured predictions
  - Object generation
  - Semantic segmentation
- Summary

Convolutional Neural Networks

- Feed-forward network
  - Convolutions
  - Non-linearity: Sigmoid or Rectified Linear Unit (ReLU)
  - Pooling: (typically) local maximum
- End-to-end supervised learning
- Representation learning

LeNet [LeCun89]

Convolutional Neural Network (CNN)

CNN had not shown impressive performance.
- Reasons for failure
  - Insufficient training data
  - Slow convergence
    - Bad activation function: Sigmoid function
    - Too many parameters
    - Limited computing resources
  - Lack of theory: needed to rely on trial-and-error

CNN recently draws a lot of attention due to its great success.
- Reasons for recent success
  - Availability of larger training datasets, e.g., ImageNet
  - Powerful GPUs
  - Better model regularization strategy such as dropout
  - Simple activation function: ReLU

Other CNNs for Classification

• Very Deep ConvNet by VGG [Simonyan15]
  - Smaller filters: 3x3
    - More non-linearity
    - Less parameters to learn: ~140 millions
  - A significant performance improvement with 16–19 layers
  - Generalization to other datasets
  - The first place for localization and the second place for classification in ILSVRC 2014

Other CNNs for Classification

• GoogLeNet [Szegedy15]
  - Network in network: inception modules
  - Auxiliary classifiers to facilitate training
  - 22 layer network: 27 layers if pooling layers are counted
  - The winner of ILSVRC 2014 classification task

AlexNet [Krizhevsky12]
Low-Level Structured Prediction using Convolutional Neural Networks

Denoising Network
- A simple CNN to reconstruct noise-free images

- Input and output are both images in a same dimension.
- Activation function: sigmoid function
- Loss function: 2-norm reconstruction error
- 24 filters in each layers but 8 inter-layer connections
- 5x5 filters and 624 convolutions (15,697 parameters)

V. Jain and S. Seung, *Natural Image Denoising with Convolutional Networks*. NIPS 2008

Super-Resolution CNN

- Network architecture
  - 3 convolution layers with ReLu for the first two layers
  - Conceptually similar to sparse coding
  - Good performance with a small number of training images


Super-Resolution CNN

- Training
  - 2-norm loss function to favor a high PSNR between input and output
  - Preprocessing: bicubic interpolation to the desired size
  - Stochastic gradient descent
  - Learned filters
Structured Prediction using CNNs

By Prof. Bohyung Han

Super-Resolution CNN

Test convergence

Results

Original / PSNR  Bicubic / 24.04 dB  SC / 25.58 dB  SRCNN / 27.58 dB

Structured Prediction in Low-Level Vision

• CNN as a collection of non-linear filters
  ▪ Low-level processing
  ▪ Learning data-driven filters
  ▪ No domain shift between input and output

Deconvolution Networks

• Generative convolutional neural network
• Domain shift between input and output
• Advantages
  ▪ Capable of structural prediction
    • Segmentation
    • Matching
    • Object generation
  ▪ More general than classification: extending applicability of CNNs
• Challenges
  ▪ More parameters
    • Difficult to train
    • Requires more training data, which may need heavy human efforts
  ▪ Task specific network: typically not transferrable
Deconvolution for Structured Prediction

- Object generation
- Semantic segmentation

Object Generation

Discriminative vs. Generative CNN

- Discriminative CNN
  - Generate an object based on high-level inputs such as
    - Class
    - Orientation with respect to camera
    - Additional parameters
      - Rotation, translation, zoom
      - Stretching horizontally or vertically
      - Hue, saturation, brightness

- Generative CNN

Goal

**Contribution**

- Knowledge transfer
  - Given limited number of viewpoints of an object, the network can use the knowledge learned from other similar objects to infer remaining viewpoints.
- Interpolation between different objects
  - Generative CNN learns the manifold of chairs.

**Data**

- Using 3D chair model dataset\textsuperscript{[Aubry14]}
  - Original dataset: 1393 chair models, 62 viewpoints, 31 azimuth angles, 2 elevation angles
  - Sanitized version: 809 models, tight cropping, resizing to 128x128
- Notation
  - \( D = \{(c^1, v^1, \theta^1), (c^2, v^2, \theta^2), \ldots, (c^N, v^N, \theta^N)\} \)
    - \( c \): class label
    - \( v \): viewpoint
    - \( \theta \): additional parameters
  - \( O = \{(x^1, s^1), (x^2, s^2), \ldots, (x^N, s^N)\} \)
    - \( x \): target RGB output image
    - \( s \): segmentation mask


**Network Architecture**

- \( g = u \circ h \)
- 32M parameters altogether

**Operations**

- Unpooling: 2x2
- Deconvolution: 5x5
- ReLU
Training

- **Objective function**
  - Minimizing the Euclidean error in 2D of reconstructing the segmented-out chair image and the segmentation mask
  \[
  \min_{W} \sum_{i=1}^{N} \lambda \left\| u_{\text{RGB}} \left( h(x^i, v^i, \theta^i) \right) - T_{\theta^i}(x^i \cdot s^i) \right\|_2^2 + \left\| u_{\text{seg}} \left( h(x^i, v^i, \theta^i) \right) - T_{\theta^i}s^i \right\|_2^2
  \]

- **Optimization**
  - Stochastic gradient descent with momentum of 0.9
  - Learning rate
    - 0.0002 for the first 500 epochs
    - Dividing by 2 after every 100 epoch
  - Orthogonal matrix initialization\[^{[Saxe14]}\]


Learned Filters

- Visualization of uconv-3 layer filters in 128x128 network

  **RGB stream**

  **Segmentation stream**

- Facts and observations
  - The final output at each position is generated from a linear combination of these filters.
  - They include edges and blobs.

Network Capacity

- **Translation**
- **Rotation**
- **Zoom**
- **Stretch**
- **Saturation**
- **Brightness**
- **Color**

Single Unit Activation

- **Images generated from single unit activations**
  - FC-1 (class)
  - FC-2 (class)
  - FC-3
  - FC-4

All neurons are set to 0’s.
Hidden Layer Analysis

• **Zoom neuron**
  - Increasing activation of the “zoom neuron” found in FC-4 feature map

• **Spatial mask**
  - Chairs generated from spatially masked 8x8 FC-5 feature map

Interpolation between Angles

- With knowledge transfer
- Without knowledge transfer

Summary

• Supervised training of CNN can also be used to generate images.
• Generative network does not merely memorize, but also generalizes well.
• The proposed network is capable of processing very different inputs using the same standard layers.
Deconvolution Network for Semantic Segmentation

Semantic Segmentation
- Segmenting image based on its semantic notion

FCN for Semantic Segmentation
- Segmentation by Fully Convolutional Network (FCN)\cite{Long15}
  - End-to-End CNN architecture for semantic segmentation
  - Convert fully connected layers to convolutional layers

Deconvolution Filter
- Bilinear interpolation filter
  - Same filter for every class
  - There is no learning!
  - Not a real deconvolution
- How does this deconvolution work?
  - Deconvolution filter is fixed.
  - Fining-tuning convolution layers of the network with segmentation ground-truth.

Limitations of FCN-based Semantic Segmentation

- Coarse output score map
  - A single bilinear filter should handle the variations in all kinds of object classes.
  - Difficult to capture detailed structure of objects in image
- Fixed size receptive field
  - Unable to handle multiple scales
  - Difficult to delineate too small or large objects compared to the size of receptive field
- Noisy predictions due to skip architecture
  - Trade off between details and noises
  - Minor quantitative performance improvement

Learning Deconvolution Network

- Instance-wise training and prediction
  - Easy data augmentation
  - Reducing solution space
  - Inference on object proposals, then aggregation
  - Labeling objects in multiple scales

Operations in Deconvolution Network

- Unpooling
  - Place activations to pooled location
  - Preserve structure of activations
- Deconvolution
  - Densify sparse activations
  - Bases to reconstruct shape
- ReLU
  - Same with convolution network
How Deconvolution Network Works?

- Visualization of activations

1. Input image
2. Object proposals
3. Prediction and aggregation
4. Results

DeconvNet

Inference

- Instance-wise prediction
  - Inference on object proposals
    - Each class corresponds to one of the channel in the output layer.
    - Label of a pixel is given by max operation of all channels.
  - Aggregation of object proposals
    - Max operation with all proposals overlapping on each pixel
    - Number of proposals: not sensitive to accuracy
    - 50 proposals for evaluation

Results

- Handling multi-scale objects naturally

Number of proposals

Input image
Ground-truth
FCN
DecovNet
DecovNet+CRF
Summary

• Confirmation of some conjectures
  ▪ Deconvolution network is conceptually reasonable.
  ▪ Learning a deep deconvolution network is a feasible option for semantic segmentation.
• Presenting a few critical training strategies
  ▪ Data augmentation
  ▪ Multi-stage training
  ▪ Batch normalization
• Good performance
  ▪ Best in all algorithms trained on PASCAL VOC dataset
  ▪ The 3rd overall
• Code available
  ▪ http://cvlab.postech.ac.kr/research/deconvnet

Motivation

• Challenges in existing supervised learning approaches
  ▪ Heavy labeling efforts in semantic segmentation
  ▪ Much more expensive to obtain pixel-wise segmentation labels than other kinds of labels
  ▪ Difficult to extend to other classes and handle more classes

Problem Setting

• Semi-supervised learning with hybrid annotations
  ▪ Many weak annotations: image-level object class labels
  ▪ Few strong annotations: full segmentation labels

DecoupledNet for Semi-Supervised Semantic Segmentation
Structured Prediction using CNNs

**Architecture**
- Classification network
- Segmentation network
- Bridging layers

**Classification Network**
- **Specification**
  - Input: image $x_i$
  - Output: 20-dimensional class label vector $S(x_i; \theta_c) \in R^L$
- **Construction**
  - Fine-tuning from VGG 16-layer net
  - Transferrable from any other existing classification networks

\[
\min_{\theta_c} \sum_i e_c(y_i; S(x_i; \theta_c)), \text{ where } y_i \in \{0,1\}^L \text{ is GT.}
\]

**Segmentation Network**
- **Specification**
  - Input: class-specific activation map $g^l_i$ of input image $x_i$
  - Output: two-channel class-specific segmentation map $M(g^l_i; \theta_s)$
- **Construction**
  - Adopting DeconvNet
  - Customized for binary segmentation

\[
\min_{\theta_s} \sum_i e_s(z_i; M(g^l_i; \theta_s)), \text{ where } z_i \text{ is binary GT.}
\]

**Bridging Layers**
- **Specification**
  - Input: concatenation of $f_{\text{spat}}$ and $f_{\text{cls}}^l$ in the channel direction
  - Output: class-specific activation map $g^l_i$
- **Construction**
  - Fully connected layers
  - $f_{\text{spat}}$: pool5
  - $f_{\text{cls}}$: backpropagating class-specific information until pool5
Class-Specific Information

- Class-specific saliency map\cite{Simonyan12}
  - Given an image, pixels related to specific class can be identified by computing gradient of class score w.r.t image by
  \[
  f_{\text{cls}}^t = \frac{\partial S_t}{\partial f^{(k)}} = \frac{\partial f^{(M)}}{\partial f^{(M-1)}} \cdot \frac{\partial f^{(M-1)}}{\partial f^{(M-2)}} \cdots \frac{\partial f^{(k+1)}}{\partial f^{(k)}}
  \]


Class-Specific Activation Maps

- Search space reduction

<table>
<thead>
<tr>
<th>Class</th>
<th>Image</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>aeroplane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bicycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bird</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Segmentation Maps

Segmentation Maps
Structured Prediction using CNNs
By Prof. Bohyung Han

Inference

• Need iterations
  ▪ Computing segmentation map for each identified label
  ▪ Using the same segmentation network with different class-specific information

\[
M(\mathbf{g}_i; \theta_s) = \max \left( M_f(\mathbf{g}^\text{person}_i; \theta_s), M_f(\mathbf{g}^\text{motorbike}_i; \theta_s), M_b(\mathbf{g}_i; \theta_s) \right)
\]

Qualitative Results

Comparison to other algorithms in PASCAL VOC 2012 validation set

<table>
<thead>
<tr>
<th># of classes</th>
<th>DecoupledNet</th>
<th>WSSL-Small-FoV</th>
<th>WSSL-Large-FoV</th>
<th>DecoupledNet-Sir</th>
<th>DecomNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>67.5</td>
<td>63.9</td>
<td>67.6</td>
<td>67.5</td>
<td>67.1</td>
</tr>
<tr>
<td>25 (x 20 classes)</td>
<td>62.1</td>
<td>56.9</td>
<td>54.2</td>
<td>50.3</td>
<td>38.6</td>
</tr>
<tr>
<td>10 (x 20 classes)</td>
<td>57.4</td>
<td>47.6</td>
<td>38.9</td>
<td>41.7</td>
<td>21.5</td>
</tr>
<tr>
<td>5 (x 20 classes)</td>
<td>53.1</td>
<td>-</td>
<td>-</td>
<td>32.7</td>
<td>15.3</td>
</tr>
</tbody>
</table>

Per-class accuracy in PASCAL VOC 2012 test set

<table>
<thead>
<tr>
<th>Models</th>
<th>bkg</th>
<th>area</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbk</th>
<th>psst</th>
<th>plat</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>DecoupledNet-Full</td>
<td>91.5</td>
<td>78.8</td>
<td>39.9</td>
<td>78.1</td>
<td>53.8</td>
<td>68.1</td>
<td>83.2</td>
<td>78.2</td>
<td>80.6</td>
<td>25.8</td>
<td>62.6</td>
<td>55.5</td>
<td>75.1</td>
<td>77.1</td>
<td>77.1</td>
<td>76.0</td>
<td>47.8</td>
<td>74.1</td>
<td>47.5</td>
<td>66.4</td>
<td>60.4</td>
</tr>
<tr>
<td>DecoupledNet-25</td>
<td>90.1</td>
<td>75.8</td>
<td>41.7</td>
<td>70.4</td>
<td>46.6</td>
<td>62.2</td>
<td>83.0</td>
<td>69.9</td>
<td>76.7</td>
<td>23.1</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>87.4</td>
</tr>
<tr>
<td>DecoupledNet-10</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>87.4</td>
</tr>
</tbody>
</table>

Quantitative Results

Comparison to other algorithms in PASCAL VOC 2012 validation set

<table>
<thead>
<tr>
<th># of classes</th>
<th>DecoupledNet</th>
<th>WSSL-Small-FoV</th>
<th>WSSL-Large-FoV</th>
<th>DecoupledNet-Sir</th>
<th>DecomNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>67.5</td>
<td>63.9</td>
<td>67.6</td>
<td>67.5</td>
<td>67.1</td>
</tr>
<tr>
<td>25 (x 20 classes)</td>
<td>62.1</td>
<td>56.9</td>
<td>54.2</td>
<td>50.3</td>
<td>38.6</td>
</tr>
<tr>
<td>10 (x 20 classes)</td>
<td>57.4</td>
<td>47.6</td>
<td>38.9</td>
<td>41.7</td>
<td>21.5</td>
</tr>
<tr>
<td>5 (x 20 classes)</td>
<td>53.1</td>
<td>-</td>
<td>-</td>
<td>32.7</td>
<td>15.3</td>
</tr>
</tbody>
</table>

Per-class accuracy in PASCAL VOC 2012 test set

<table>
<thead>
<tr>
<th>Models</th>
<th>bkg</th>
<th>area</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbk</th>
<th>psst</th>
<th>plat</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>DecoupledNet-Full</td>
<td>91.5</td>
<td>78.8</td>
<td>39.9</td>
<td>78.1</td>
<td>53.8</td>
<td>68.1</td>
<td>83.2</td>
<td>78.2</td>
<td>80.6</td>
<td>25.8</td>
<td>62.6</td>
<td>55.5</td>
<td>75.1</td>
<td>77.1</td>
<td>77.1</td>
<td>76.0</td>
<td>47.8</td>
<td>74.1</td>
<td>47.5</td>
<td>66.4</td>
<td>60.4</td>
</tr>
<tr>
<td>DecoupledNet-25</td>
<td>90.1</td>
<td>75.8</td>
<td>41.7</td>
<td>70.4</td>
<td>46.6</td>
<td>62.2</td>
<td>83.0</td>
<td>69.9</td>
<td>76.7</td>
<td>23.1</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>87.4</td>
</tr>
<tr>
<td>DecoupledNet-10</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>88.5</td>
<td>73.8</td>
<td>40.1</td>
<td>68.1</td>
<td>45.5</td>
<td>59.5</td>
<td>76.4</td>
<td>62.7</td>
<td>71.4</td>
<td>17.7</td>
<td>87.4</td>
</tr>
</tbody>
</table>

Summary

• Novel deep neural network architecture for semi-supervised learning with hybrid annotations
• Outstanding performance
• Easy training
  ▪ Free from iterative procedure for label inference
• More flexible approach
  ▪ Extensible to other classes by fine-tuning existing classification networks
  ▪ Capable of handling many classes without parameter explosion
• Code available
  ▪ http://cvlab.postech.ac.kr/research/decouplednet
Conclusion

- CNN: useful for structured predictions
  - 2D/3D object generation
  - Semantic segmentation
  - Human pose estimation
  - Visual tracking
  - Image enhancement
  - ...

- More parameters but trainable
- Unpooling is approximate but effective.

Concluding Remark