OverFeat

Integrated Recognition, Localization and Detection using Convolutional Networks

Sermanet et. al

Presentation by Eric Holmdahl
Roadmap

1. Goal
2. Background
3. Related Work
4. Algorithm Overview
5. Breakdown By Task
   1. Classification
   2. Localization
   3. Detection
Goal

Perform classification, localization, and detection on the ImageNet Dataset
Classification

Determining what is the main object in an image
Localization

- Determining where an object is located in an image
Detection

• Performing localization for all objects present in an image
Background: Feed Forward Neural Networks
Background: Convolutional Nets

- Alternating convolution and max pooling layers feed into fully connected neural net
- Max pooling: with window size k\times k, outputs highest intensity value in window size
- Convolution: Scanning window, shared weights within window

\[
x_{ij}^k = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{k-1}
\]
Related Work
Krizhevsky et. Al: *ImageNet Classification With Deep Convolutional Neural Networks*
Review: Krizhevsky Architecture

• Large CNNs used to densely process images with overlapping windows
• ReLU Nonlinear neuron output
• DropOut
Krizhevsky Results

- Brought CNNs to forefront of classification/localization/detection problem

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 (val)</th>
<th>Top-5 (val)</th>
<th>Top-5 (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT + FVs [7]</td>
<td>—</td>
<td>—</td>
<td>26.2%</td>
</tr>
<tr>
<td>1 CNN</td>
<td>40.7%</td>
<td>18.2%</td>
<td>—</td>
</tr>
<tr>
<td>5 CNNs</td>
<td>38.1%</td>
<td>16.4%</td>
<td>16.4%</td>
</tr>
<tr>
<td>1 CNN*</td>
<td>39.0%</td>
<td>16.6%</td>
<td>—</td>
</tr>
<tr>
<td>7 CNNs*</td>
<td>36.7%</td>
<td>15.4%</td>
<td>15.3%</td>
</tr>
</tbody>
</table>
Giusti et. al: *Fast Image Scanning With Deep Convolutional Networks*
Giusti Fast Scanning

• Problem: CNNs perform a great deal of redundant computing of convolutions due to overlapping patches
• Solution: Apply convolution to entire image at once!
Convolutional Layer:

Apply convolutional kernel to each input map of each fragment, same number of fragments as previous level.

Max Pooling Layer:

Pixel at \((\bar{x}, \bar{y})\)in output map is max of all pixels from input map at \((x, y)\) such that

\[
\begin{align*}
  o_x + k\bar{x} &\leq x \leq o_x + k\bar{x} + k - 1 \\
o_y + k\bar{y} &\leq y \leq o_y + k\bar{y} + k - 1
\end{align*}
\]

Generates \(k^2\) new fragments at each max pooling layer.
Giusti et al.: Fast Image Scanning With Deep Convolutional Networks
Giusti et. al: Results

| Layer (l) | s   | $s_{l-1}$ | $|P_{l-1}|$ | $|P_l|$ | $w_l$ | $k_l$ | $F_l$ | FLOPS$_l^{\text{patch}}$ $\cdot 10^9$ | FLOPS$_l^{\text{image}}$ $\cdot 10^9$ | speedup |
|-----------|-----|-----------|---------|--------|------|------|-----|-------------------------------|-------------------------------|---------|
| 1         | 512 | 559       | 1       | 48     | 92   | 4    | 1   | 3408                         | 0.5                           | 7114.8  |
| 3         | 512 | 279       | 48      | 48     | 42   | 5    | 4   | 53271                        | 35.9                          | 1485.1  |
| 5         | 512 | 139       | 48      | 48     | 18   | 4    | 16  | 6262                         | 22.8                          | 274.7   |
| 7         | 512 | 69        | 48      | 48     | 6    | 4    | 64  | 695                          | 22.5                          | 30.9    |
| Total     |     |           |         |        |      |      |     | 63636                        | 81.6                          | 779.8   |

Provides massive improvements in speed for sliding window CNNs!
Algorithm Overview
Algorithm Overview: Training

Train Classifier → Train Localization Regressor
Algorithm Overview: Training

Train Classifier

Train Localization Regressor
Algorithm Overview: Training

Train Classifier → Train Localization Regressor
Algorithm Overview: Training

- Input: Images with classification and bounding box
- Training objective: Minimize l2 norm between generated bounding box and ground truth
- One regressor generated for each possible image class
- Output: (x,y) coordinates of top left, top right corner of bounding box
Algorithm Overview: Runtime

1. Perform classification at each location using trained CNN
2. Perform localization on all classified regions generated by classifier
Algorithm Overview: Runtime

3. Merge bounding boxes with sufficient overlap from localization and sufficient confidence of being same object from classifier
Breakdown By Task
Classification
OverFeat Feature Extraction

- First 5 layers of Deep Convolutional Neural Net: similar to Krizhevsky’s
- Images downsampled to 256x256
- No contrast normalization, non-overlapping pooling

<table>
<thead>
<tr>
<th>Layer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Output 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage</td>
<td>conv + max</td>
<td>conv + max</td>
<td>conv</td>
<td>conv</td>
<td>conv + max</td>
<td>full</td>
<td>full</td>
<td>full</td>
</tr>
<tr>
<td># channels</td>
<td>96</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
<td>3072</td>
<td>4096</td>
<td>1000</td>
</tr>
<tr>
<td>Filter size</td>
<td>11x11</td>
<td>5x5</td>
<td>3x3</td>
<td>3x3</td>
<td>3x3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Conv. stride</td>
<td>4x4</td>
<td>1x1</td>
<td>1x1</td>
<td>1x1</td>
<td>1x1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pooling size</td>
<td>2x2</td>
<td>2x2</td>
<td>-</td>
<td>-</td>
<td>2x2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pooling stride</td>
<td>2x2</td>
<td>2x2</td>
<td>-</td>
<td>-</td>
<td>2x2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Zero-Padding size</td>
<td>-</td>
<td>1x1x1x1</td>
<td>1x1x1x1</td>
<td>1x1x1x1</td>
<td>1x1x1x1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Spatial input size</td>
<td>231x231</td>
<td>24x24</td>
<td>12x12</td>
<td>12x12</td>
<td>12x12</td>
<td>6x6</td>
<td>1x1</td>
<td>1x1</td>
</tr>
</tbody>
</table>
Figure 3: 1D illustration (to scale) of output map computation for classification, using $y$-dimension from scale 2 as an example (see Table 5). (a): 20 pixel unpooled layer 5 feature map. (b): max pooling over non-overlapping 3 pixel groups, using offsets of $\Delta = \{0, 1, 2\}$ pixels (red, green, blue respectively). (c): The resulting 6 pixel pooled maps, for different $\Delta$. (d): 5 pixel classifier (layers 6,7) is applied in sliding window fashion to pooled maps, yielding 2 pixel by $C$ maps for each $\Delta$. (e): reshaped into 6 pixel by $C$ output maps.
Multi-Scale Classification

- Classification performed at 6 scales at test time, but only 1 scale at runtime
- Increases robustness of model

<table>
<thead>
<tr>
<th>Scale</th>
<th>Input size</th>
<th>Layer 5 pre-pool</th>
<th>Layer 5 post-pool</th>
<th>Classifier map (pre-reshape)</th>
<th>Classifier map size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>245x245</td>
<td>17x17</td>
<td>(5x5)x(3x3)</td>
<td>(1x1)x(3x3)x'C'</td>
<td>3x3x'C'</td>
</tr>
<tr>
<td>2</td>
<td>281x317</td>
<td>20x23</td>
<td>(6x7)x(3x3)</td>
<td>(2x3)x(3x3)x'C'</td>
<td>6x9x'C'</td>
</tr>
<tr>
<td>3</td>
<td>317x389</td>
<td>23x29</td>
<td>(7x9)x(3x3)</td>
<td>(3x5)x(3x3)x'C'</td>
<td>9x15x'C'</td>
</tr>
<tr>
<td>4</td>
<td>389x461</td>
<td>29x35</td>
<td>(9x11)x(3x3)</td>
<td>(5x7)x(3x3)x'C'</td>
<td>15x21x'C'</td>
</tr>
<tr>
<td>5</td>
<td>425x497</td>
<td>32x35</td>
<td>(10x11)x(3x3)</td>
<td>(6x7)x(3x3)x'C'</td>
<td>18x24x'C'</td>
</tr>
<tr>
<td>6</td>
<td>461x569</td>
<td>35x44</td>
<td>(11x14)x(3x3)</td>
<td>(7x10)x(3x3)x'C'</td>
<td>21x30x'C'</td>
</tr>
</tbody>
</table>

Table 5: Spatial dimensions of our multi-scale approach. 6 different sizes of input images are used, resulting in layer 5 unpoled feature maps of differing spatial resolution (although not indicated in the table, all have 256 feature channels). The (3x3) results from our dense pooling operation with \((\Delta_x, \Delta_y) = \{0, 1, 2\}\). See text and Fig. 3 for details for how these are converted into output maps.
Classification: CNNs and Sliding Windows

- Single output:
  - 1x1 output
  - no feature space
  - **blue**: feature maps
  - **green**: operation kernel
  - typical training setup

![Diagram of CNN architecture](image-url)

OverFeat • Pierre Sermanet • New York University
Classification: CNNs and Sliding Windows

- **Multiple outputs:**
  - 2x2 output
  - input stride 2x2
  - recompute only extra yellow areas

OverFeat • Pierre Sermanet • New York University
Classification: CNNs and Sliding Windows

- With feature space
  - 3 input channels
  - 4 feature maps
  - 2 feature maps
  - 4 feature maps
  - 2 outputs (e.g. 2-class classifier)
# Classification: Results

<table>
<thead>
<tr>
<th>Approach</th>
<th>Top-1 error %</th>
<th>Top-5 error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krizhevsky et al. [15]</td>
<td>40.7</td>
<td>18.2</td>
</tr>
<tr>
<td>OverFeat - 1 fast model, scale 1, coarse stride</td>
<td>39.28</td>
<td>17.12</td>
</tr>
<tr>
<td>OverFeat - 1 fast model, scale 1, fine stride</td>
<td>39.01</td>
<td>16.97</td>
</tr>
<tr>
<td>OverFeat - 1 fast model, 4 scales (1,2,4,6), fine stride</td>
<td>38.57</td>
<td>16.39</td>
</tr>
<tr>
<td>OverFeat - 1 fast model, 6 scales (1-6), fine stride</td>
<td>38.12</td>
<td>16.27</td>
</tr>
<tr>
<td>OverFeat - 1 accurate model, 4 corners + center + flip</td>
<td>35.60</td>
<td>14.71</td>
</tr>
<tr>
<td>OverFeat - 1 accurate model, 4 scales, fine stride</td>
<td>35.74</td>
<td>14.18</td>
</tr>
<tr>
<td>OverFeat - 7 fast models, 4 scales, fine stride</td>
<td>35.10</td>
<td>13.86</td>
</tr>
<tr>
<td>OverFeat - 7 accurate models, 4 scales, fine stride</td>
<td>33.96</td>
<td>13.24</td>
</tr>
</tbody>
</table>

![Error Rate Chart](chart.png)

- **Clarifai** ImageNet11 pre-training: 11.2% Top-5 error rate
- **NUS** validation fine-tuning: 13.0%
- **ZF**: 13.5%
- **Andrew Howard**: 13.6%
- **OverFeat** 7 big models: 14.2%
- **7 fast models**: 13.6%
- **UvA - Euvision**: 14.3%
- **Adobe**: 15.2%
- **VGG**: 15.2%
- **SuperVision 7 models + ImageNet11**: 15.3%
- **Cognitive Vision**: 16.1%
- **SuperVision 5 models**: 16.4%

Legend:
- **ILSVRC12**: Yellow
- **ILSVRC13**: Blue
- **Post competition**: Red
Localization
Training Localizer

- Use same first 5 layers as trained classifier
- Remove fully connected layers, replace with regressor
- Train again on labeled input with bounding boxes
Localization: Fully Connected Layers

Figure 8: Application of the regression network to layer 5 features, at scale 2, for example. (a) The input to the regressor at this scale are 6x7 pixels spatially by 256 channels for each of the (3x3) $\Delta_x, \Delta_y$ shifts. (b) Each unit in the 1st layer of the regression net is connected to a 5x5 spatial neighborhood in the layer 5 maps, as well as all 256 channels. Shifting the 5x5 neighborhood around results in a map of 2x3 spatial extent, for each of the 4096 channels in the layer, and for each of the (3x3) $\Delta_x, \Delta_y$ shifts. (c) The 2nd regression layer has 1024 units and is fully connected (i.e. the purple element only connects to the purple element in (b), across all 4096 channels). (d) The output of the regression network is a 4-vector (specifying the edges of the bounding box) for each location in the 2x3 map, and for each of the (3x3) $\Delta_x, \Delta_y$ shifts.
Localization: Bounding Boxes Produced By Regression
Localization: Combing Predictions

Algorithm:

(a) Assign to $C_s$ the set of classes in the top $k$ for each scale $s \in 1\ldots6$, found by taking the maximum detection class outputs across spatial locations for that scale.

(b) Assign to $B_s$ the set of bounding boxes predicted by the regressor network for each class in $C_s$, across all spatial locations at scale $s$.

(c) Assign $B \leftarrow \bigcup_s B_s$

(d) Repeat merging until done:

(e) $(b_1^*, b_2^*) = \arg\min_{b_1 \neq b_2 \in B} \text{match}\_\text{score}(b_1, b_2)$

(f) If $\text{match}\_\text{score}(b_1^*, b_2^*) > t$, stop.

(g) Otherwise, set $B \leftarrow B \setminus \{b_1^*, b_2^*\} \cup \text{box}\_\text{merge}(b_1^*, b_2^*)$
Localization: Results

- SCR, 4 scales: 30.0%
- SCR, 3 scales: 31.3%
- SCR, 2 scales: 31.5%
- SCR, 1 scale: 36.0%
- SCR, centered crop: 40.0%
- PCR, 3 scales: 44.1%

Top 5 error rate
Detection
Detection:
- 200 classes
- Smaller objects than classification/localization
- Any number of objects (including zero)
- Penalty for false positives
Differences Between Detection and Location

- Can now have many objects instead of just one
- Penalized for incorrect guesses
- Need to distinguish background from objects
Training Detector

• Almost identical to classification/localization training
• New class added – background
• Background class updated on the fly: extremely incorrect classifications are used to train background class
Detection Results

Figure 10: ILSVRC12 and ILSVRC13 competitions results (test set). Our entry is the winner of the ILSVRC13 localization competition with 29.9% error (top 5). Note that training and testing data is the same for both years. The OverFeat entry uses 4 scales and a single-class regression approach.
Conclusion

OverFeat provides a way to extract powerful CNN based features for image classification, localization and detection with high speed and precision.
Thanks!