The structure of typical convolutional networks

Receptive fields and image segmentation

Object detection

A CNN for image classification

Recall: the goal of this model is to map an input image to a class prediction.
The AlexNet model

A breakthrough in image understanding

Each large block represents a data tensor
Each smaller block represents a filter
The filter size and stride are shown
The number of filters can be deduced from the number of feature channels
There are two parallel streams in this network (for efficiency reasons)

How deep is deep enough?

AlexNet (2012)

5 convolutional layers

3 fully-connected layers

AlexNet (2012)

VGG-M (2013)

VGG-VD-16 (2014)

AlexNet (2012)

VGG-M (2013)

VGG-VD-16 (2014)

GoogLeNet (2014)
How deep is deep enough?


16 convolutional layers
50 convolutional layers
152 convolutional layers

Accuracy

3 \times more accurate in 3 years

Remark: 101 ResNet layers same size/speed as 16 VGG-VD layers

Reason: far fewer feature channels (quadratic speed/space gain)

Moral: optimize your architecture
Model size

Num. of parameters is about the same

Remark: 101 ResNet layers same size/speed as 16 VGG-VD layers

Reason: far fewer feature channels (quadratic speed/space gain)

Moral: optimize your architecture

The structure of typical convolutional networks

Receptive fields and image segmentation

Object detection

Semantic image segmentation

Label individual pixels

Receptive field

The part of the image looked at by a neuron

Receptive Field (RF) of a neuron
- The subset of the image affecting the value of a neuron

Small vs large RFs
- Small RF: spatially specific, but can only account for small visual structures
- Large RF: spatially a-specific, but can account for large visual structure

How to make the RF large
- Use large filters
- Chain several filters
- Interleave downsampling along the chain
  E.g. downsampling 2x increases the RF size 2x.
Comparing the receptive fields
Convolutional vs fully connected layers

**Convolutionsal layers**
- Neurons are spatially selective, can be used to localize things.

**Fully connected layers**
- Neurons are global, do not characterize well position.

Which one is more useful for pixel level labelling?

A fully connected layer is just a large filter
The filter support fills the entire input tensor

\[
F(k) = K \times w(k) \times 1 \times 1 \times K
\]

Fully-convolutional neural networks

- Class predictions

Dense evaluation
- Apply the whole network convolutional
- Computes a vector of class probabilities at each pixel

Downsampling
- For efficiency, the input data is substantially down sampled in the network
- The output is fairly low resolution (e.g. 1/32 of original)

The object detection problem

The goal of object detection is to simultaneously classify, enumerate, and localise known object types in an image.

A key challenge is that the number of object instances is not known a priori.

Detections with CNNs

Region-based Convolutional Neural Network (R-CNN)

CNNs compute a fixed number of image features. A new computational mechanism is needed in order to detect a variable number of objects.

Region-based CNN (R-CNN) use a region proposal algorithm to extract a large number of potential object regions, and then a CNN to assess each one of them.

Region proposal algorithm

Obtain a shortlist of regions that may contain objects

A region proposal algorithm produces a shortlist of regions that are likely to contain whole objects.

The Selective Search method by [van de Sande, Uijlings et al.]:

- Uses hierarchical segmentation based on colour uniformity and image edges.
- Produces about ~2000 regions/image with a >95% probability of hitting any relevant object in the image.
A region proposal is slightly dilated to capture some visual context and then cropped and resized in order to be passed to a CNN.

The cropped and resize region is passed through a CNN to extract a corresponding feature vector (or image representation).

The feature vector is then classified by means of a linear predictor (a Support Vector Machine, similar to a perceptron). There are C + 1 possible object types, including "no object" (background).

A second linear regression is used to refine the bounding box location. In the example, the person's legs were not included in the proposal, but regression can fix this mistake.
Positive and negative training regions
Based on overlap with ground truth bounding box

- **a positive training region**
  - overlap > 70%

- **a negative training region**
  - overlap < 30%

R-CNN results on PASCAL VOC
At the time of introduction (2013)

Despite its conceptual simplicity, at the time of introduction R-CNN was substantially better than all existing methods. This is due to the power of the CNN classifier. Importantly, the CNN is **pre-trained** on the ImageNet data (1M images) for classification (using only image-level labels), then **fine-tuned** on PASCAL VOC data (5K images) for object detection (using region-level labels).

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td>29.6%</td>
</tr>
<tr>
<td>UVA sel. search (Uijlings et al. 2013)</td>
<td>41.7%</td>
<td>40.4%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>54.2%</td>
<td>50.2%</td>
</tr>
<tr>
<td>SegDPM (Fidler et al. 2013)</td>
<td>58.5%</td>
<td>53.7%</td>
</tr>
<tr>
<td>R-CNN (TorontoNet)</td>
<td>62.1%</td>
<td>62.9%</td>
</tr>
<tr>
<td>R-CNN (TorontoNet) + bbox regression</td>
<td>66.0%</td>
<td>62.9%</td>
</tr>
</tbody>
</table>

R-CNN shortcomings
Not a fully-integrated CNN model

- Components such as the SVM are **not integrated** in the CNN framework.
- **Slow** due to the need of evaluating a CNN a thousand or more times per image.

Towards better R-CNNs
Integrate more of the blocks as CNN components

R-CNN can be improved substantially in three ways:
- By integrating all blocks in a end-to-end trainable CNN
- By accelerating region-specific computations
- By replacing region proposal generation with something better
Accelerating R-CNN

**Problem:** The fundamental bottleneck is evaluating the CNN from scratch for each image region.

**Solution:** compute all the convolutional features just once, and then crop directly the resulting feature map. Only the fully-connected layers are evaluated for each region.

**How:** spatial pooling layer.

---

The Spatial Pooling (SP) layer

As a building block

SP extracts a feature vector for each of the $R$ regions. The output are thus $R$ tensor of size $1 \times 1 \times C$.

Alternatively, this can be seen as a single $1 \times 1 \times C \times R$ tensor.

---

The Spatially Pyramid Pooling Layer

SP with multiple subdivisions

SPP is similar to SP, but pools features in the tiles of a grid-like subdivision of the region. The resulting feature vector captures the spatial layout of the original region.

---

**Summary**

Fast R-CNN

**Example detections**

R-CNN

Detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet.

**PASCAL VOC Leaderboards**

Detection challenge (comp4: train on own data)

http://tinyurl.com/h7uzkov

2014 4× improvement in accuracy 2016

**Fast and Faster R-CNN performance**

Both faster and better!

<table>
<thead>
<tr>
<th>Method</th>
<th>Time / image</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>~50s</td>
<td>66.0</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>~2s</td>
<td>66.9</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>198ms</td>
<td>69.9</td>
</tr>
</tbody>
</table>