AIMS Big Data
Lectures 4: Deep learning 2 of 2
Andrea Vedaldi
MT 2016
For slides and up-to-date information:
http://www.robots.ox.ac.uk/~vedaldi/teach.html

Typical CNN architectures and transfer learning

Visualizing representations

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

How deep is deep enough?

How deep is deep enough?
How deep is deep enough?


- AlexNet (2012)
- VGG-M (2013)
- VGG-VD-16 (2014)
- GoogLeNet (2014)

16 convolutional layers
50 convolutional layers
152 convolutional layers


Accuracy

3 × 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

Pre-training and transfer learning

[Evaluations in A. S. Razavian, 2014, Chatfield et al., 2014]

- Pertained layers
- Fine-tuned layers

CNN as universal representations
- First several layers in most CNNs are generic
- They can be reused when training data is comparatively scarce

Application
- Pre-train on ImageNet classification
  1M images
- Cut at some deep conv or FC layer to get features

Transfer learning

Deep representations are generic

Deep representations used as local features

A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

In R-CNN and similar models, the most important shared component are the convolutional features.
Demo

Typical CNN architectures and transfer learning

Visualizing representations

Segmentation

Object detection

Visualization: Pre-Image

The representation is not injective

The reconstruction ambiguity provides useful information about the representation

The representation is not injective

The reconstruction ambiguity provides useful information about the representation
Finding a Pre-Image

A simple yet general and effective method

\[ \min_x \| \Phi(x) - \Phi_0 \|_2^2 \]

Start from \textit{random noise}

Optimize using stochastic \textit{gradient descent}

Related Work

Analysis tools
Visualizing higher-layer features of a deep network
Ethan et al. 2009
[intermediate features]

Deep inside convolutional networks
Simonyan et al. 2014
[deepest features, aka “deep dreams”]

DeConvNets
Zeiler et al. In ECCV, 2014
[intermediate features]

Understanding neural networks through deep visualisation
Yosinski et al. 2015
[intermediate features]

Artistic tools
Google’s “inceptionism”
Mordvintsev et al. 2015

Style synthesis and transfer
Gatys et al. 2015

Inverting a Deep CNN

AlexNet
[Krizhevsky et al. 2012]
Inverting a Deep CNN

Original Image

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original Image

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original Image

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original Image
Inverting a Deep CNN

Original Image

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original Image

Inverting a Deep CNN

Original Image

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original Image
Inverting a Deep CNN

<table>
<thead>
<tr>
<th>conv 1</th>
<th>conv 2</th>
<th>conv 3</th>
<th>conv 4</th>
<th>conv 5</th>
<th>fc 6</th>
<th>fc 7</th>
<th>fc 8</th>
</tr>
</thead>
</table>

Original Image

Inverting a Deep CNN

<table>
<thead>
<tr>
<th>conv 1</th>
<th>conv 2</th>
<th>conv 3</th>
<th>conv 4</th>
<th>conv 5</th>
<th>fc 6</th>
<th>fc 7</th>
<th>fc 8</th>
</tr>
</thead>
</table>

Original Image

Inverting a Deep CNN

<table>
<thead>
<tr>
<th>conv 1</th>
<th>conv 2</th>
<th>conv 3</th>
<th>conv 4</th>
<th>conv 5</th>
<th>fc 6</th>
<th>fc 7</th>
<th>fc 8</th>
</tr>
</thead>
</table>

Original Image

Inverting a Deep CNN

<table>
<thead>
<tr>
<th>conv 1</th>
<th>conv 2</th>
<th>conv 3</th>
<th>conv 4</th>
<th>conv 5</th>
<th>fc 6</th>
<th>fc 7</th>
<th>fc 8</th>
</tr>
</thead>
</table>

Original Image
4.1 Data Augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the training set by a factor of 2048, though the resulting training examples are, of course, highly interdependent. Without this scheme, our network suffers from substantial overfitting, which would have turned out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random 256→224→224 patches (and their horizontal reflections) from the training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, we add multiples of the found principal components.

The second form of data augmentation consists of altering the intensities of the RGB channels in layer on the ten patches.

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the training set by a factor of 2048, though the resulting training examples are, of course, highly interdependent. Without this scheme, our network suffers from substantial overfitting, which would have...
ImageNet training set. To each training image, we add multiples of the found principal components, five times. This forced us to use much smaller networks. At test time, the network makes a prediction by extracting features from these patches (and their horizontal reflections) from the (normalized, pooled) output of the second convolutional layer. The fourth convolutional layer has 256 kernels of size 224 x 224 x 3 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 256 kernels of size 224 x 224 x 3. This increases the size of our network's remaining layers. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the GPUs are trained on these extracted patches (the four corner patches and the center patch) as well as their horizontal reflections (hence ten patches in all), and averaging the predictions made by the network's softmax layer on the ten patches.

4.1 Data Augmentation

This is the reason why the input images in Figure 2 are 224 x 224 x 3-dimensional.

The first form of data augmentation consists of generating image translations and horizontal reflections (hence ten patches in all), and averaging the predictions made by the network's softmax layer on the ten patches. Without this scheme, our network suffers from substantial overfitting, which would have made each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.
Network comparison
“conv5” features

<table>
<thead>
<tr>
<th>AlexNet</th>
<th>VGG-M</th>
<th>VGG-VD</th>
</tr>
</thead>
</table>

The starting point is not an image
Remember: the starting point is not an image

Caricaturization
[Google Inceptionism 2015, Mahendran et al. 2015]

Emphasise patterns that are detected by a certain representation

$$\min_x -\langle \Phi(x_0), \Phi(x) \rangle - R_{TV}(x) + R_u(x)$$

Key differences:
- the starting point is the image $x_0$
- particular configurations of features are emphasized, not individual features
Caricaturization (VGG-M)

Surprisingly, the filters learned by discriminative neural networks capture well the “style” of an image.

This can be used to transfer the style of an image (e.g. a painting) to any other.

Optimisation based


Feed-forward neural network equivalents


Interlude: neural art

Generation by moment matching

Moment matching

- Content statistics: same as inversion
- Style statistics: cross-channel correlations

$x^* = \arg \min_x E(x; x_{content}, x_{style})$
Typical CNN architectures and transfer learning

Visualizing representations

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

Convolutional 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

\[ F(k) \]

\[ K \times 1 \times 1 \]

\[ 1 \times 1 \times K \]
Fully-convolutional neural networks

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.
Next term!