Module 9A: Introduction to Convolutional Neural Networks (CNNs)

Convolution for feature extraction

- Linear filtering

\[ w(x,y) \ast f(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) f(x-i, y-j) \]
Convolution for feature extraction

* Linear filtering

\[ w(x, y) * f(x, y) = \sum_{x=a}^{b} \sum_{y=b}^{h} w(s, t) f(x-s, y-t) \]

Example

\[ w(x, y) * f(x, y) = \sum_{x=a}^{b} \sum_{y=b}^{h} w(s, t) f(x-s, y-t) \]

Flip mask w.r.t. signal

Also see Chapter 3.4 from Gonzalez & Woods

Example kernels (1)

\[ w = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \]

Low-pass filter

Example kernels (2)

\[ w = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

High-pass filter

Example kernels (3)

Example kernels (4)

Also see Chapter 3.4 from Gonzalez & Woods
Convolution for feature extraction
Example kernels (5)

Convolution for feature extraction
Example kernels (6)
- Pass-band filters
  - Scale
  - Orientation

Convolution for feature extraction
Example kernels (7)

Convolution for feature extraction
Color kernels (1)
$m \times m$ kernel

Convolution for feature extraction
Color kernels (2)

Convolution for feature extraction
Color kernels (3)
$m \times m \times 3$ kernel
Convolution for feature extraction

Color kernels (4)

- Single filters to find specific color changes

Convolution for feature extraction

Color kernels (5)

- Single filters to find specific color changes

Convolution for feature extraction

Color kernels (6)

- Quadrature filter pairs to find local frequency content

Convolution for feature extraction

Color kernels (7)

- Quadrature filter pairs to find local frequency content

Convolution for feature extraction

- A convolution operation can be employed to extract information from an image
  - Edges
  - Specific frequencies (low-pass, high-pass, band-pass)
  - Orientation and scale
  - Color variations

Module 9A: Introduction to Convolutional Neural Networks (CNNs)

Neural networks
Neural networks

- Biologically inspired method
  - Built from “perceptrons” (simplified model of a neuron)
  - Use of hidden layers performing non-linear operations

- Important parameters
  - Number of hidden layers / layer sizes
  - Activation function
  - Learning rate

Hidden layers of perceptrons (neurons)
- Layering makes computation more efficient
- Network capacity increases with the number of perceptrons

Perceptron
- Linear combination of inputs followed by non-linearity

Activation functions
- Sigmoid: $\sigma(x) = \frac{1}{1 + e^{-x}}$
- Tanh: $\tanh(x)$

Rectified Linear Unit (ReLU): $\max(0,x)$

Backpropagation
- Efficient way to find the gradient of the loss function
  - Use a forward pass to find the loss
  - Use a backward pass to find the gradient
  - Update the weights using gradient descent

Module 9A: Introduction to Convolutional Neural Networks (CNNs)

Convolutional neural networks

This part is largely based on a series of lectures from Stanford. Please see [http://cs231n.github.io](http://cs231n.github.io) for the original material and find the lectures on YouTube (search for cs231).
ConvNets
Introduction

- Convolutional Neural Network (CNN) or just Convolutional Network (ConvNet)
- Idea pretty old!

Fukushima, 1980, Neurocognition

The rise of deep learning!

The rise of deep learning!

The rise of deep learning!

Nature cover, January 2016
Science cover, July 2017

Massive labeled datasets: ImageNet
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- 1.2M images, 1000 categories
- Increased hardware capabilities
- Very powerful GPU's

Some test images for “Hammer” class

First breakthrough: 2012 ILSVRC won by a ConvNet

ImageNet error over time

ILSVRC top-5 error vs ImageNet

Top5 error: 16.422%
Top5 error: >26.172%
ConvNets
Introduction

* 2012 ILSVRC winner: AlexNet

AlexNet architecture

ConvNets
Introduction

* Traditional approach vs. ConvNets

Traditional approach
ConvNets

“End-to-end Learning”

ConvNets
Introduction

* Hierarchical representation

Courtesy Zieler and Fergus, ECCV, 2013

ConvNets
Architecture

* Layer types

Convolutional layer: Perform basic convolution(s) of the previous layer with a set of convolution kernels (filters)
Normalization layer: Enforces similar statistics between the layers
Activation layer: Non-linear activation function, “fires” based on weighted input
Pooling layer: Reducing the filter result at a region to a single number (subsampling)
Fully connected layer: Basically a regular (feed-forward) neural network

ConvNets
Architecture

* Convolution layer

MxNx3 RGB image

NxMx1 Convolution result

Convolution

PxQx3 filter

MxNx3 RGB image
ConvNets Architecture

- Convolution layer
  - Set of $D$ filters
  - $M \times N \times 3$ RGB image
  - $M \times N \times D$ Convolution result

- Activation layer
  - $x \rightarrow \max(D, x)$

- Pooling layer
  - Form of sub-sampling

- Fully connected layer
  - All the "pixels" after the last pooling (or ReLU) layer are connected to inputs of the fully connected layer

- Have a look at AlexNet again
  - Total number of parameters in the first conv layer?
  - How many filters are used in the first layer?
  - Total number of parameters in the first pool layer?
ConvNets

- ConvNet layers are used as lego blocks

Different architectures have been proposed

- GoogleNet
  - Based on so-called inception module
- VGGNet
- ResNet

Module 9A: Introduction to Convolutional Neural Networks (CNNs)

Transfer learning

- We want to use ConvNets, but...
  - They require a lot of labeled examples
  - Millions of parameters to estimate!
  - It takes weeks to train them
- Solution: Transfer learning!
  - Do not start from scratch
  - Use networks trained on large data sets and retrain/modify them for your purpose

Transfer learning

- ConvNet as feature extractor
  - Get the activations from one of the fully connected layers and use them as features
  - Train a classifier using these CNN codes
- Fine-tuning the ConvNet
  - Retrain only the last layer(s) of the network
  - Use back-propagation but fix the weights from all other layers.
Transfer learning

Oquab et al., CVPR 2014

When and how to fine-tune?

- **CNN codes, later layers**
  - Fine-tune, later layers
- **CNN codes, earlier layers**
  - Fine-tune, up to earlier layers

New vs old data similarity
- **small**
  - CNN codes, later layers
  - Fine-tune, later layers
- **large**
  - CNN codes, earlier layers
  - Fine-tune, up to earlier layers

Size of new data set
- **small**
- **large**

Transfer learning

When and how to fine-tune?

- **CNN codes, later layers**
  - Fine-tune, later layers
- **CNN codes, earlier layers**
  - Fine-tune, up to earlier layers

New vs old data similarity
- **small**
  - CNN codes, later layers
  - Fine-tune, later layers
- **large**
  - CNN codes, earlier layers
  - Fine-tune, up to earlier layers

Size of new data set
- **small**
- **large**

Classification methods

ConvNet Transfer Learning

- **MATLAB**
  - There’s a very nice tutorial in the Matlab documentation:

Make sure you have at least Matlab 2016b installed and a CUDA®-enabled NVIDIA® GPU with compute capability 3.0 or higher.

Toolboxes required: Parallel Computing Toolbox, Neural Network Toolbox

Summary

- **Convolution**
  - A convolution operation can be employed to extract information from an image (e.g. edges, colors, texture)

- **Neural networks**
  - Biologically inspired method for machine learning
  - Built from “perceptrons” (simplified model of a neuron)
  - Use of hidden layers performing non-linear operations

Summary

- **Convolutional Neural Networks (CNN/ConvNets)**
  - Revitalized due tremendous increase in computational capabilities in combination with the availability of large, labeled data sets.
  - Very good generalization capabilities!
    - Surpassing human performance in many tasks already
    - A lot of data and a lot of time (or $$...) is required to train a network from scratch

- **Transfer learning**
  - Don’t fully retrain a ConvNet, either
    - use CNN codes from the fully-connected layers (FC6, FC7,...);
    - retrain only a part of the network with new data.
  - Our best guess for medical image analysis!