Traditional Image Categorization:
Training phase

Training Images

Training

Image Features

Classifier Training

Trained Classifier

Training Labels

Slide credit: Jia-Bin Huang
Traditional Image Categorization: Testing phase

Training Images

Training

Training Labels

Image Features

Classifier Training

Trained Classifier

Testing

Image Features

Trained Classifier

Prediction

Outdoor

Test Image

Slide credit: Jia-Bin Huang
Features have been key

SIFT [Lowe IJCV 04]

HOG [Dalal and Triggs CVPR 05]

SPM [Lazebnik et al. CVPR 06]

Textons

and many others:

SURF, MSER, LBP, GLOH, .....
Learning a Hierarchy of Feature Extractors

- Hierarchical and expressive feature representations
- Trained end-to-end, rather than hand-crafted for each task
- Remarkable in transferring knowledge across tasks
Neural Networks
Biological neuron and Perceptrons

A biological neuron

An artificial neuron (Perceptron)

Input: $x_1, x_2, x_3, \ldots, x_D$

Weights: $w_1, w_2, w_3, \ldots, w_D$

Output: $\text{sgn}(w \cdot x + b)$

Slide credit: Jia-Bin Huang
Simple, Complex and Hypercomplex cells

David H. Hubel and Torsten Wiesel

Suggested a **hierarchy of feature detectors** in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.
Hubel-Wiesel Architecture and Multi-layer Neural Network

Hubel and Weisel’s architecture

Multi-layer Neural Network

Slide credit: Jia-Bin Huang
Neuron: Linear Perceptron

- Inputs are feature values
- Each feature has a weight
- Sum is the activation

\[
\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)
\]

- If the activation is:
  - Positive, output +1
  - Negative, output -1
Two-layer perceptron network
Two-layer perceptron network
Two-layer perceptron network

Slide credit: Pieter Abeel and Dan Klein
Two-layer perceptron network

$h_w(f(x))$

Slide credit: Pieter Abeel and Dan Klein
Neural networks

Linear score function:

2-layer Neural Network

\[ f = Wx \]
\[ f = W_2 \max(0, W_1 x) \]

Non-linearity
Activation functions

Sigmoid
\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

\[ \text{tanh} \quad \text{tanh}(x) \]

ReLU \quad \text{max}(0, x)

Leaky ReLU
\[ \text{max}(0.1x, x) \]
Neural networks

Linear score function:

\[ f = Wx \]

\[ f = W_2 \max(0, W_1 x) \]

2-layer Neural Network

3-layer Neural Network

\[ f = W_3 \max(0, W_2 \max(0, W_1 x)) \]
Multi-layer neural network
Learning w

- Training examples
  \[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\]

- Objective: a misclassification loss
  \[\min_w \sum_{i=1}^{m} \left( y^{(i)} - h_w(f(x^{(i)})) \right)^2\]

- Procedure:
  - Gradient descent or hill climbing

Slide credit: Pieter Abeel and Dan Klein
Neural network properties

- Theorem (Universal function approximators): A two-layer network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

- Practical considerations:
  - Can be seen as learning the features
  - Large number of neurons
    - Danger for overfitting
  - Hill-climbing can get stuck in local optima
Convolutional Neural Networks
Images as input to neural networks
Images as input to neural networks

Example: 200x200 image
40K hidden units
~2B parameters!!!
Images as input to neural networks

Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
Motivation

• Sparse interactions – *receptive fields*
  – Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  – Composition of layers will expand local -> global.
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good
when input image is registered (e.g.,
face recognition).
STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).
Motivation

• Sparse interactions – *receptive fields*
  – Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  – Composition of layers will expand local -> global.

• Parameter sharing
  – ‘Tied weights’ – use same weights for more than one perceptron in the neural network.
  – Leads to *equivariant representation*
    • If input changes (e.g., translates), then output changes similarly
Share the same parameters across different locations (assuming input is stationary):
Filtering reminder:
Correlation (rotated convolution)

$$I[.,.]$$

$$h[.,.]$$

$$h[m, n] = \sum_{k, l} f[k, l] I[m + k, n + l]$$

Credit: S. Seitz
Convolutional Layer

Perceptron: \[ \text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \]

\[ w \cdot x = \sum_j w_j x_j \]

This is convolution!

Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

Shared weights
Convolutional Layer

Learn multiple filters.
Filter = ‘local’ perceptron. Also called kernel.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters
Interpretation

- prediction of class
- distributed representations
- feature sharing
- compositionality

Input image

Lee et al. “Convolutional DBN's ...” ICML 2009
Stride = 1
Stride = 1
Stride = 3
Stride = 3
Stride = 3
Stride = 3
Pooling Layer

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
By *pooling* responses at different locations, we gain robustness to the exact spatial location of image features.
Pooling is similar to downsampling...

...except sometimes we don’t want to blur, as other functions might be better for classification.
Pooling Layer: Receptive Field Size

$h^{n-1}$ → Conv. layer → $h^n$ → Pool. layer → $h^{n+1}$
Pooling Layer: Examples

Max-pooling:

$$h^n_j(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y})$$

Average-pooling:

$$h^n_j(x, y) = \frac{1}{K} \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y})$$
Max pooling

Single depth slice

X

Y

1 0 2 3
4 6 6 8
3 1 1 0
1 2 2 4

→

6 8
3 4
Yann LeCun’s MNIST CNN architecture
Convolutions: More detail

32x32x3 image

- Height: 32
- Width: 32
- Depth: 3
Convolutions: More detail

32x32x3 image

5x5x3 filter
Convolutions: More detail

Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolutions: More detail

For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolutions: More detail

- Conv, ReLU
  - e.g. 6 5x5x3 filters
- Conv, ReLU
  - e.g. 10 5x5x6 filters
- Conv, ReLU

Andrej Karpathy
Convolutions: More detail

Output size: \((N - F) / \text{stride} + 1\)
- Layers
- Kernel sizes
- Strides
- # channels
- # kernels
- Max pooling
AlexNet diagram (simplified)

Input size
227 x 227 x 3

Conv 1
11 x 11 x 3
Stride 4
96 filters

Conv 2
5 x 5 x 48
Stride 1
256 filters

Conv 3
3 x 3 x 256
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
256 filters

[Krizhevsky et al. 2012]