CNN
!!! Warning !!!

Learning jargon is always painful...
...even if the concepts behind the jargon are not hard.

So, let’s get used to it.

“In mathematics you don't understand things. You just get used to them.”

von Neumann (a joke)
Gartner Hype Cycle
The limits of learning?
So far...

• PASCAL VOC = ~75%
• ImageNet = ~75%; human performance = ~95%

Smart human brains used intuition and understanding of how we think vision works, and it’s pretty good.
Image formation (+database+labels) → Filtering (gradients/transforms) → Feature points (saliency+description) → Dictionary building (compression) → Classifier (decision making)

Recognition: Classification → Object Detection → Segmentation

Captured+manual.
Hand designed.
Hand designed.
Learned.
Well, what do we have?

Best performing visions systems have commonality:

- Hand designed features
  - Gradients + non-linear operations (exponentiation, clamping, binning)
  - Features in combination (parts-based models)
  - Multi-scale representations

- Machine learning from databases

- Some classifiers (e.g., SVM)
But it’s still not that good...

• PASCAL VOC = ~75%
• ImageNet = ~75%; human performance = ~95%

Problems:
- Lossy features
- Lossy quantization
- Imperfect classifier
But it’s still not that good…

• PASCAL VOC = ~75%
• ImageNet = ~75%; human performance = ~95%

How to solve?
• Features: More principled modeling?
• Quantization: More data and more compute?
  It’s just an interpolation problem; let’s represent the space with less approximation.
• Classifier: ...
Wouldn’t it be great if we could...

- Image formation (+database+labels)
- Filtering (gradients/transforms)
- Feature points (saliency+description)
- Dictionary building (compression)
- Classifier (decision making)

Recognition: Classification Object Detection Segmentation

Captured+manual.
Learned. (space specified a bit)
Learned.
Learned.
End to end learning!
Neural Networks
Neural Networks

• Basic building block for composition is a perceptron (Rosenblatt c.1960)

• Linear classifier – vector of weights w and a ‘bias’ b

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

\[w \cdot x \equiv \sum_j w_j x_j\]
Binary classifying an image

• Each pixel of the image would be an input.
• So, for a 28 x 28 image, we vectorize.
• \( x = 1 \times 784 \)

• \( w \) is a vector of weights for each pixel, 784 x 1
• \( b \) is a scalar bias per perceptron

• result = \( xw + b \) \rightarrow (1x784) \times (784x1) + b = (1x1)+b
Neural Networks - multiclass

• Add more perceptrons

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]

Binary output
Binary output
Binary output
Multi-class classifying an image

• Each pixel of the image would be an input.
• So, for a 28 x 28 image, we vectorize.
• \( x = 1 \times 784 \)

• \( W \) is a matrix of weights for each pixel/each perceptron
  • \( W = 10 \times 784 \) (10-class classification)
• \( b \) is a bias per perceptron (vector of biases); \( (1 \times 10) \)

• result = \( xW + b \)  -> \( (1\times784) \times (784 \times 10) + b \)
  -> \( (1 \times 10) + (1 \times 10) = \text{output vector} \)
Bias convenience

• To turn this classification operation into a multiplication only:
  • Create a ‘fake’ feature with value 1 to represent the bias
  • Add an extra weight that can vary

\[
\begin{align*}
\text{output} &= \begin{cases} 
0 & \text{if } w \cdot x \leq 0 \\
1 & \text{if } w \cdot x > 0 
\end{cases} \\
w \cdot x & \equiv \sum_j w_j x_j
\end{align*}
\]
Attempt to represent complex functions as compositions of smaller functions.

Outputs from one perception are fed into inputs of another perceptron.
Sets of layers and the connections (weights) between them define the network architecture.
Composition

Layers that are in between the input and the output are called hidden layers, because we are going to learn their weights via an optimization process.
Composition

It’s all just matrix multiplication!
GPUs -> special hardware for fast/large matrix multiplication.

Nielsen
Problem 1 with all linear functions

• We have formed chains of linear functions.
• We know that linear functions can be reduced
  • \( g = f(h(x)) \)

Our composition of functions is really just a single function : ( 
Problem 2 with all linear functions

- Linear classifiers: small change in input can cause large change in binary output
  = problem for composition of functions

Activation function
Problem 2 with all linear functions

• Linear classifiers: small change in input can cause large change in binary output.

• We want:

\[ w + \Delta w \]

causes a small change in the output
Let’s introduce non-linearities

• We’re going to introduce non-linear functions to transform the features.

\[ \sigma(w \cdot x + b) \]

\[ \sigma(z) \equiv \frac{1}{1 + e^{-z}}. \]
Multi-layer perceptron (MLP)

• ...is a ‘fully connected’ neural network with non-linear activation functions.

• ‘Feed-forward’ neural network
MLP

• Use is grounded in theory
  • Universal approximation theorem (Goodfellow 6.4.1)

• Can represent a NAND circuit, from which any binary function can be built by compositions of NANDs

• With enough parameters, it can approximate any function (next lecture).
Supervised Learning

\[
\{(x^i, y^i), i = 1 \ldots P\} \quad \text{training dataset}
\]

\(x^i\) i-th input training example

\(y^i\) i-th target label

\(P\) number of training examples

Goal: predict the target label of unseen inputs.
Supervised Deep Learning

Classification

Denoising

OCR

“dog”

“2 3 4 5”
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: \( \text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \)

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

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

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

Shared weights
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

high-level parts

mid-level parts

low level parts

Input image

Lee et al. “Convolutional DBN's ...” ICML 2009
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?
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

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

6  8
3  4
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:

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

L2-pooling:

\[ h^n_j(x, y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y})^2} \]

L2-pooling over features:

\[ h^n_j(x, y) = \sqrt{\sum_{k \in N(j)} h^{n-1}_k(x, y)^2} \]
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.
Yann LeCun’s MNIST CNN architecture
Convolutions: More detail

32x32x3 image

32  height
32  width
3    depth
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]
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

- Layer 1: 3x3 input, 32 output channels, 32x32 Conv-ReLU layer with 6 5x5x3 filters.
- Layer 2: 32x32 input, 28 output channels, 28x28 Conv-ReLU layer with 10 5x5x6 filters.
- Layer 3: 28x28 input, 24 output channels, 24x24 Conv-ReLU layer.
- ....
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
CONV NETS: EXAMPLES

- Scene Parsing

Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013
Pinheiro et al. “Recurrent CNN for scene parsing” arxiv 2013
CONV NETS: EXAMPLES

- Segmentation 3D volumetric images

Ciresan et al. “DNN segment neuronal membranes...” NIPS 2012
Turaga et al. “Maximin learning of image segmentation” NIPS 2009
CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

Ciresan et al. “MCDNN for image classification” CVPR 2012
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014
Dataset: ImageNet 2012

- **mammal** → **placental** → **carnivore** → **canine** → **dog** → **working dog** → **husky**

- **Eskimo dog**, **husky** (breed of heavy-coated Arctic sled dog)
- **working dog** (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
- **dog**, **domestic dog**, *Canis familiaris* (a member of the genus *Canis* (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) "the dog barked all night")
- **canine**, **canid** (any of various fissiped mammals with nonretractile claws and typically long muzzles)
- **carnivore** (a terrestrial or aquatic flesh-eating mammal) "terrestrial carnivores have four or five clawed digits on each limb"
- **placental**, **placental mammal**, **therian, theria** (mammals having a placenta, all mammals except monotremes and marsupials)
- **mammal**, **mammalian** (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
- **vertebrate, cranate** (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
- **chordate** (any animal of the phylum Chordata having a notochord or spinal column)
- **animal, animate being, beast, brute, creature, fauna** (a living organism characterized by voluntary movement)
- **organism, living thing, animate thing** (a living (or once living) entity)
- **whole, unit** (an assemblage of parts that is regarded as a single entity) "how big is that part compared to the whole?"; "the team is a unit"
- **object, physical object** (a tangible and visible entity, an entity that can cast a shadow) "it was full of rackets, balls and other objects"
- **physical entity** (an entity that has physical existence)
- **entity** (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Deng et al. “Imagenet: a large scale hierarchical image database” CVPR 2009
Architecture for Classification

Total nr. params: 60M
4M  LINEAR  4M
16M FULLY CONNECTED  16M
37M FULLY CONNECTED  37M

Total nr. flops: 832M
74M MAX POOLING
74M
442K CONV  224M
1.3M CONV  149M
884K CONV

105M MAX POOLING
307K LOCAL CONTRAST NORM  223M
307K CONV

884K LOCAL CONTRAST NORM
35K CONV  105M
input

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Results: ILSVRC 2012

TASK 1 - CLASSIFICATION

Error %

CNN  SIFT+FV  SVM1  SVM2  NCM

35  30  25  30  35

TASK 2 - DETECTION

Error %

CNN  DPM-SVM1  DPM-SVM2

20  30  50

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
More ConvNet explanations

• https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/