1950
Future Vision

2017 MWF 1PM 368
Computer Vision

EYE ROBOT
CSCI 1430

ROBOT
ISAAC ASIMOV
Neural Networks: example

Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).
Forward Propagation

\[ \mathbf{x} \rightarrow \max(0, \mathbf{W}^1 \mathbf{x}) \rightarrow \max(0, \mathbf{W}^2 h^1) \rightarrow \mathbf{W}^3 h^2 \rightarrow o \]

\[ \mathbf{x} \in \mathbb{R}^D \quad \mathbf{W}^1 \in \mathbb{R}^{N_1 \times D} \quad \mathbf{b}^1 \in \mathbb{R}^{N_1} \quad h^1 \in \mathbb{R}^{N_1} \]

\[ h^1 = \max(0, \mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) \]

\( \mathbf{W}^1 \) 1-st layer weight matrix or weights
\( \mathbf{b}^1 \) 1-st layer biases

The non-linearity \( u = \max(0, v) \) is called **ReLU** in the DL literature. Each output hidden unit takes as input all the units at the previous layer: each such layer is called “fully connected”.
Forward Propagation

\[ h^1 \in \mathbb{R}^{N_1} \quad W^2 \in \mathbb{R}^{N_2 \times N_1} \quad b^2 \in \mathbb{R}^{N_2} \quad h^2 \in \mathbb{R}^{N_2} \]

\[ h^2 = \max(0, W^2 h^1 + b^2) \]

\( W^2 \) 2-nd layer weight matrix or weights

\( b^2 \) 2-nd layer biases
Forward Propagation

\[ h^2 \in \mathbb{R}^{N_2}, \quad W^3 \in \mathbb{R}^{N_3 \times N_2}, \quad b^3 \in \mathbb{R}^{N_3}, \quad o \in \mathbb{R}^{N_3} \]

\[ o = \max \left( 0, W^3 h^2 + b^3 \right) \]

\( W^3 \)  3-rd layer weight matrix or weights

\( b^3 \)  3-rd layer biases
Alternative Graphical Representation

$h^k \xrightarrow{\text{max} \left(0, W^{k+1} h^k \right)} h^{k+1}$

$h^k \xrightarrow{W^{k+1}} h^{k+1}$
How Good is a Network?

$$y = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & k & 0 & \ldots & 0 \end{bmatrix}$$

Probability of class k given input (softmax):

$$p(c_k = 1 | x) = \frac{e^{o_k}}{\sum_{j=1}^{C} e^{o_j}}$$

(Per-sample) Loss; e.g., negative log-likelihood (good for classification of small number of classes):

$$L(x, y ; \theta) = - \sum_j y_j \log p(c_j | x)$$
**Training**

**Learning** consists of minimizing the loss (plus some regularization term) w.r.t. parameters over the whole training set.

\[ \theta^* = \arg \min_{\theta} \sum_{n=1}^{P} L(x^n, y^n; \theta) \]

**Question:** How to minimize a complicated function of the parameters?

**Answer:** Chain rule, a.k.a. **Backpropagation**! That is the procedure to compute gradients of the loss w.r.t. parameters in a multi-layer neural network.

Universality

• A single-layer network can learn any function:
  • So long as it is differentiable
  • To some approximation;
    More perceptrons = a better approximation

• Visual proof (Michael Nielson):
If a single-layer network can learn any function...

- ...given enough parameters...
- ...then why do we go deeper?

Intuitively, composition is efficient because it allows reuse.

Empirically, deep networks do a better job than shallow networks at learning such hierarchies of knowledge.
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
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)

\[ I[., .] \quad h[., .] \]

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

Credit: S. Seitz
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

\[ \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \times \text{Shared weights} = \text{output} \]
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
Convolutional Layer

\[ h^n_j = \max \left( 0, \sum_{k=1}^{K} h^{n-1}_k \ast w^n_{kj} \right) \]

- \( n \) = layer number
- \( K \) = kernel size
- \( j \) = # channels (input) or # filters (depth)
\[ h_j^n = \max(0, \sum_{k=1}^{K} h_{k}^{n-1} * w_{kj}^{n}) \]

output
feature map

input feature map

kernel
\[ h_j^n = \max \left( 0, \sum_{k=1}^{K} h_k^{n-1} \ast w_{kj}^n \right) \]

**output feature map**

**input feature map**

**kernel**
Stride = 1
Stride = 1
Stride = 3
Stride = 3
Stride = 3
Stride = 3
Key Ideas

A standard neural net applied to images:
- scales quadratically with the size of the input
- does not leverage stationarity
Key Ideas

A standard neural net applied to images:
- scales quadratically with the size of the input
- does not leverage stationarity

Solution:
- connect each hidden unit to a small patch of the input
- share the weight across space

This is called: convolutional layer.
A network with convolutional layers is called convolutional network.
- add multiple filters per layer

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998
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: 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:

\[ h_j^n(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y}) \]

Average-pooling:

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

L2-pooling:

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

L2-pooling over features:

\[ h_j^n(x, y) = \sqrt{\sum_{k \in N(j)} h_k^{n-1}(x, y)^2} \]
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: \((P+K-1) \times (P+K-1)\)
If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$
Local Contrast Normalization
Local Contrast Normalization

We want the same response.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

\( N(x, y) = \) model pixel values in window as a normal distribution

\( m = \) mean
\( \sigma = \) variance

Note: computational cost is negligible w.r.t. conv. layer.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i \left( N(x, y) \right)}{\sigma^i \left( N(x, y) \right)} \]

Performed also across features and in the higher layers.

Effects:
– improves invariance
– improves optimization
– increases sparsity

**Note:** computational cost is negligible w.r.t. conv. layer.
ConvNets: Typical Stage

One stage (zoom)

Filter Bank

Rectification + Contrast Normalization

Pooling

courtesy of K. Kavukcuoglu
ConvNets: Typical Stage

One stage (zoom)

Conceptually similar to: SIFT, HoG, etc.
Note: after one stage the number of feature maps is usually increased (conv. layer) and the spatial resolution is usually decreased (stride in conv. and pooling layers). Receptive field gets bigger.

Reasons:
- gain invariance to spatial translation (pooling layer)
- increase specificity of features (approaching object specific units)
ConvNets: Typical Architecture

One stage (zoom)

Whole system

Input Image → 1\textsuperscript{st} stage → 2\textsuperscript{nd} stage → 3\textsuperscript{rd} stage → Fully Conn. Layers → Class Labels
ConvNets: Typical Architecture

Whole system

Input Image → 1st stage → 2nd stage → 3rd stage → Fully Conn. Layers → Class Labels

Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM
Lazebnik et al. “...Spatial Pyramid Matching...” CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM
ConvNets: Training

All layers are differentiable (a.e.).
We can use standard back-propagation.

Algorithm:
  Given a small mini-batch
  - F-PROP
  - B-PROP
  - PARAMETER UPDATE
Outline

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

- OCR / House number & Traffic sign classification

Ciresan et al. "MCDNN for image classification" CVPR 2012
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

- Object detection

Szegedy et al. “DNN for object detection” NIPS 2013
CONV NETS: EXAMPLES

- Face Verification & Identification

Dataset: ImageNet 2012

- mammal ➔ placental ➔ carnivore ➔ canine ➔ dog ➔ working dog ➔ husky

- *S: (a) Eskimo dog, husky (breed of heavy-coated Arctic sled dog)
  - direct hypernym / inherited hypernym / sister term
  - *S: (a) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
  - *S: (a) 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"
  - *S: (a) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
  - *S: (a) carnivore (a terrestrial or aquatic flesh-eating mammal) "terrestrial carnivores have four or five clawed digits on each limb"
  - *S: (a) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
  - *S: (a) mammal, mammals (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)
  - *S: (a) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
  - *S: (a) chordate (any animal of the phylum Chordata having a notochord or spinal column)
  - *S: (a) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
  - *S: (a) organism, being (a living thing that has (or can develop) the ability to act or function independently)
  - *S: (a) living thing, animate thing (a living (or once living) entity)
  - *S: (a) 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"
  - *S: (a) object, physical object (a tangible and visible entity; an entity that can cast a shadow) "it was full of rackets, balls and other objects"
  - *S: (a) physical entity (an entity that has physical existence)
  - *S: (a) 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
<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>mite</td>
<td>container ship</td>
<td>motor scooter</td>
<td>leopard</td>
</tr>
<tr>
<td>black widow</td>
<td>lifeboat</td>
<td>go-kart</td>
<td>jaguar</td>
</tr>
<tr>
<td>cockroach</td>
<td>amphibian</td>
<td>moped</td>
<td>cheetah</td>
</tr>
<tr>
<td>tick</td>
<td>fireboat</td>
<td>bumper car</td>
<td>snow leopard</td>
</tr>
<tr>
<td>starfish</td>
<td>drilling platform</td>
<td>golfcart</td>
<td>Egyptian cat</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>cherry</td>
<td>Madagascar cat</td>
</tr>
<tr>
<td>convertible</td>
<td>agaric</td>
<td>dalmatian</td>
<td>squirrel monkey</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>grape</td>
<td>spider monkey</td>
</tr>
<tr>
<td>pickup</td>
<td>jelly fungus</td>
<td>elderberry</td>
<td>titi</td>
</tr>
<tr>
<td>beach wagon</td>
<td>gill fungus</td>
<td>currant</td>
<td>indri</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man's-fingers</td>
<td>currant</td>
<td>howler monkey</td>
</tr>
</tbody>
</table>
Architecture for Classification

category prediction

LINEAR

FULLY CONNECTED

FULLY CONNECTED

MAX POOLING

CONV

CONV

CONV

MAX POOLING

LOCAL CONTRAST NORM

CONV

MAX POOLING

LOCAL CONTRAST NORM

CONV

input
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
Architecture for Classification

Total nr. params: 60M
4M LINEAR
16M FULLY CONNECTED
37M FULLY CONNECTED
MAX POOLING
442K CONV
1.3M CONV
884K CONV
MAX POOLING
307K LOCAL CONTRAST NORM
35K CONV

Total nr. flops: 832M
4M
16M
37M
74M
224M
149M
223M
105M

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

Task 1 - Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>15</td>
</tr>
<tr>
<td>SIFT+FV</td>
<td>25</td>
</tr>
<tr>
<td>SVM1</td>
<td>27</td>
</tr>
<tr>
<td>SVM2</td>
<td>30</td>
</tr>
<tr>
<td>NCM</td>
<td>33</td>
</tr>
</tbody>
</table>

Task 2 - Detection

<table>
<thead>
<tr>
<th>Method</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>30</td>
</tr>
<tr>
<td>DPM-SVM1</td>
<td>48</td>
</tr>
<tr>
<td>DPM-SVM2</td>
<td>50</td>
</tr>
</tbody>
</table>

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Phew!

• Friday:

• Regularization