Convolutional Neural Networks

Machine Perception

Otmar Hilliges

30 August 2018

[Some slides adopted from Karpathy, De Freitas, Serre]

Advanced Interactive Technologies

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

This morning

Introduction to Deep Learning

MLP

Universal function approximator theorem

Backprop Algorithm

① Backpropagation – single unit

$$* \; f^{(3)} = \|z^{(2)} - y\|^2 = z^{(2)T} z^{(2)} - 2 z^{(2)} y + y^T y$$

$$\frac{\partial z^{(3)}}{\partial z^{(2)}}^{*} = 2 z^{(2)} - 2y = 2(z^{(2)} - y)$$

$$\frac{\partial z^{(2)}}{\partial z^{(1)}} = \sigma(z^{(1)})(1 - \sigma(z^{(1)}))$$

$$\frac{\partial z^{(1)}}{\partial \theta^{(1)}} = z^{(0)} = \bar{x}$$

$$z^{(3)} = C(\theta)$$

$$\uparrow$$

$$\boxed{f^{(3)} = \|z^{(2)} - y\|^2}$$

$$z^{(2)} \uparrow$$

$$\boxed{f^{(2)} = \sigma(z^{(1)})}$$

$$z^{(1)} \uparrow$$

$$\boxed{f^{(1)} = \sum \theta^{(1)} z^{(0)}}$$

$$\uparrow$$

$$z^{(0)} = \bar{x}$$

$$\frac{\partial C}{\partial C} = \frac{\partial z^{(3)}}{\partial z^{(3)}} = 1$$

$$\frac{\partial z^{(3)}}{\partial z^{(2)}} \cdot 1$$

$$\frac{\partial z^{(2)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(3)}}{\partial z^{(2)}} \cdot 1$$

$$\frac{\partial z^{(1)}}{\partial \theta^{(1)}} \cdot \frac{\partial z^{(2)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(3)}}{\partial z^{(2)}} \cdot 1 = \frac{\partial C}{\partial \theta} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

$$= 2(z^{(2)} - y)\sigma(z^{(1)})(1 - \sigma(z^{(1)}))\bar{x}$$

(11) BP layer wise

$\delta^{(\ell-1)}$: Gradient that flows from layer (ℓ) to $(\ell-1)$

$\delta_i^{(\ell-1)}$: Gradient wrt to the ith unit in the $(\ell-1)^{th}$-layer

Example: "Layer 2 receives inputs $z^{(1)}$ and passes down gradients $\delta^{(1)}$".

(11a)

$$\delta_i^{(\ell-1)} = \frac{\partial C}{\partial z_i^{(\ell-1)}} = \sum_{j=1}^{N} \underbrace{\frac{\partial C}{\partial z_j^{(\ell)}}}_{\delta_j^{(\ell)}} \frac{\partial z_j^{(\ell)}}{\partial z_i^{(\ell-1)}}$$

(11b)

$$\delta^{(\ell-1)} = \sum_{j=1}^{N} \delta_j^{(\ell)} \frac{\partial z_j^{(\ell)}}{\partial z^{(\ell-1)}}$$

$$= \underbrace{\frac{\partial C}{\partial z^{(\ell)}}}_{\in \mathbb{R}^{1\times n}} \underbrace{\frac{\partial z^{(\ell)}}{\partial z^{(\ell-1)}}}_{\in \mathbb{R}^{n\times m}}$$

$$\overset{\langle 1 \rangle}{z_j^{(\ell)}} = \sigma(\overset{\langle 1,M\rangle}{\theta_j^{(\ell)}} \overset{\langle M,1\rangle}{z^{(\ell-1)}})$$

$$\overset{\langle N,1\rangle}{z^{(\ell)}} = \sigma(\overset{\langle N,M\rangle}{\theta^{(\ell)}} \overset{\langle M,1\rangle}{z^{(\ell-1)}})$$

$$\sigma\left((\theta_1, \theta_2, \dots) \begin{pmatrix} z_1 \\ z_2 \\ \vdots \end{pmatrix} \right)$$

(III) BP - parameter update

$$\frac{\partial C}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial \theta_j^{(l)}} \Big\} \text{ weight update for } j^{th} \text{ unit}$$

$$\frac{\partial C}{\partial \theta^{(l)}} = \sum_{j=1}^{N} \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial \theta^{(l)}}$$

$j=1 \rightarrow$

$j=2 \rightarrow$

$\vdots$

$$\left.\begin{array}{c} \begin{array}{cc} & M \end{array} \\ N\begin{bmatrix} \alpha_1^1 & \cdots & \alpha^1 \\ 0 & \cdots & 0 \\ & \cdots & 0 \end{bmatrix} \\ \begin{array}{cc} & M \end{array} \\ N\begin{bmatrix} 0 & \cdots & 0 \\ \alpha_1^2 & \cdots & \alpha^2 \\ 0 & \cdots & 0 \end{bmatrix} \end{array}\right\} \quad \begin{array}{cc} & M \\ N & \begin{bmatrix} \alpha_{11}^1 & \cdots & \alpha_m^1 \\ \alpha_{11}^2 & \cdots & \alpha_m^2 \\ \vdots & \ddots & \end{bmatrix} \end{array}$$

$$= \sum_{j=1}^{N} \delta_j^{(l)} \frac{\partial z_j^{(l)}}{\partial \theta^{(l)}}$$

$$= \frac{\partial C}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial \theta^{(l)}}$$

Now

Neuroscience Motivation for CNNs

Convolutional Neural Networks for Perception

Motivation – How Do I See?

Figure 8. Visual input to the brain goes from eye to LGN and then to primary visual cortex, or area V1, which is located in the posterior of the occipital lobe.
Adapted from Polyak (1957).

[Kruger, Maersk, Jannssen et al. TPAMI 2012]

Feedforward object recognition (RSVP)

Specificity versus Invariance

Goal: Classification performance

Trade-off between:

Specificity

Invariance

- Affine transforms
- Lighting

Impacts:

Generalization ability

[Riesenhuber, Poggio. 2000. Models of Object Recognition. Nature]

Hubel & Wiesel – Visual Cortex & IT

1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...

Huber & Wiesel – Mapping Simple and Complex cells

[Hubel & Wiesel. 1962.
Receptive fields of single neurones in the cat's striate cortex.
J Physiol.1959;148:574–591.]

https://www.youtube.com/watch?v=8VdFf3egwfg

Invariance to affine transforms

Receptive fields

Particular part of the body surface in which a stimulus will trigger firing in sensory neurons

- Part of a hair in cochlea, piece of retina, tongue

Receptive fields exist in auditory system, the somatosensory system, and the visual system

Concept can be extended up to the neural system

- Many sensory receptors all synapsing with a single cell collectively form the receptive field of that cell
- E.g., a group of ganglion cells in retina form the receptive field for a cell in the brain

[http://en.wikipedia.org/wiki/Receptive_field]

Receptive Fields

[Hubel & Wiesel]

Neurocognitron

[Fukushima 1980]

HMAX Model

S (simple) cells are tuned to specific stimuli and have typically small receptive fields

C (complex) cells combine output from various S units to increase invariance and receptive field

Many iterations of these operations allow for the construction of complex objects from low-level features

[Adopted from: Thomas Serre et al. PNAS 2007]

HMAX Model

S-cell response:

$$y = exp - \frac{1}{2\sigma^2} \sum_{j=1}^{n_{s_k}} \left(w_j - x_j\right)^2$$

C-cell response:

$$y = \max_{j=1 \ldots n_{C_k}} x_j$$

[Adopted from: Thomas Serre et al. PNAS 2007]

[Adopted from: Thomas Serre et al. PNAS 2007]

Results

‘Solves’ the categorization problem - if mask forces feedforward processing

[Thomas Serre et al. PNAS 2007]

LeNet-5

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]

ImageNet - 2012

ImageNet Classification with

Deep Convolutional Neural Networks

IMAGENET

[Krizhevsky, Sutskever, Hinton, 2012]

CNNs @ Large Scale

Classification

Retrieval

[Krizhevsky 2012]

Fine-Grained Prediction

Detection

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation

[Farabet et al., 2012]

Advanced Topics

[Vinyals et al., 2015]

Convolutional Neural Networks

What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Linear Transforms

Given a transform T, a function f, defining a translation or shift operation and input vectors $\boldsymbol{u}, \boldsymbol{v}$ scalars α, β

<u>Definition:</u> Transform T is *linear* if,

$$T(\alpha \boldsymbol{u} + \beta \boldsymbol{v}) = \alpha T(\boldsymbol{u}) + \beta T(\boldsymbol{v})$$

<u>Definition:</u> A transform T is *invariant* to f if,

$$T\big(f(\boldsymbol{u})\big) = T(\boldsymbol{u})$$

<u>Definition:</u> A transform T is *equivariant* to f if,

$$T\big(f(\boldsymbol{u})\big) = f\big(T(\boldsymbol{u})\big)$$

Any linear, shift-equivariant transform T can be written as a convolution

Linear Filtering

Operations are "shift-invariant" if *K* does NOT depend on (*i*,*j*): using same weights <u>everywhere</u>!

Linear operations can be written:

$$I'(i,j) = \sum_{(m,n)\in N(i,j)} K(\cancel{i,j}, m, n) I(i+m, j+n)$$

I is the input image; *I'* is the output of the operation.

k is the *kernel* of the operation. *N*(*m*,*n*) is a neighbourhood of (*m*,*n*).

Correlation
(e.g. template matching)

$$o(i,j) = c_{11}\, I(i-1,j-1) + c_{12}\, I(i-1,j) + c_{13}\, I(i-1,j+1) + c_{21}\, I(i,j-1) + c_{22}\, I(i,j) + c_{23}\, I(i,j+1) + c_{31}\, I(i+1,j-1) + c_{32}\, I(i+1,j) + c_{33}\, I(i+1,j+1)$$

Convolution

(e.g. point spread function)

Kernel

K(-1,-1)	K(0,-1)	K(1,-1)
K(-1,0)	K(0,0)	K(1,0)
K(-1,1)	K(0,1)	K(1,1)

$$
\begin{aligned}
I'(x,y) = {} & K(1,1)I(x-1,y-1) + K(0,1)I(x,y-1) + K(-1,1)I(x+1,y-1) \\
& + K(1,0)I(x-1,y) + K(0,0)I(x,y) + K(-1,0)I(x+1,y) \\
& + K(1,-1)I(x-1,y+1) + K(0,-1)I(x-1,y) + K(-1,-1)I(x+1,y+1)
\end{aligned}
$$

Correlation

$$I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) I(i+m, j+n)$$

Convolution

$$I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) I(i-m, j-n)$$

$$= \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(-m,-n) I(i+m, j+n)$$

So if *K(i,j) = K(-i, -j)*, then Correlation == Convolution

Convolution as Matrix multiplication

Convolution operation on finite signals and kernels (i.e., images) can be implemented via matrix-vector multiplication:

$$I(u) = 0 \text{ for } u \notin [1{:}\,n], \qquad K(u) = 0 \text{ for } u \notin [1{:}\,m], m \leq n$$

$$(I * K) = \begin{pmatrix} k_1 & 0 & \cdots & 0 \\ k_2 & k_1 & \vdots & \vdots \\ k_3 & k_2 & k_1 & \\ \vdots & k_3 & \vdots & \\ 0 & \vdots & 0 & k_m \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ \vdots \\ I_n \end{pmatrix}$$

Differentiation and convolution

Recall, for 2D function, f(x,y):

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon} \right)$$

This is linear and shift invariant, so must be the result of a convolution.

We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

(which is obviously a convolution)

-1	1

[Slide: D.A. Forsyth]

Vertical gradients from finite differences

[Slide: D.A. Forsyth]

Convolution Layer

32x32x3 image

[Slide adopted from: Li, Karpathy]

Convolution Layer

32x32x3 image

5x5x3 filter

Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

[Slides adopted from: Li, Karpathy]

Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

[Slides adopted from: Li, Karpathy]

Convolution Layer

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

[Slides adopted from: Li, Karpathy]

Convolution Layer

[Slides adopted from: Li, Karpathy]

Convolution Layer

consider a second, green filter

[Slides adopted from: Li, Karpathy]

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!

[Slides adopted from: Li, Karpathy]

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

[Slides adopted from: Li, Karpathy]

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

[Slides adopted from: Li, Karpathy]

Layer 1

[Zeiler & Fergus]

Layer 2

[Zeiler & Fergus]

The brain/neuron view of CONV Layer

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

[Slides adopted from: Li, Karpathy]

The brain/neuron view of CONV Layer

It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

[Slides adopted from: Li, Karpathy]

The brain/neuron view of CONV Layer

An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

[Slides adopted from: Li, Karpathy]

The brain/neuron view of CONV Layer

E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

[Slides adopted from: Li, Karpathy]

Weight sharing

[image source: http://www.jefkine.com/]

[Slides adopted from: Li, Karpathy]

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

[Slides adopted from: Li, Karpathy]

MAX POOLING

Single depth slice

x

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

max pool with 2x2 filters
and stride 2

6	8
3	4

[Slides adopted from: Li, Karpathy]

[deeplearningbook.org]