CS 559 Deep Learning

Convolutional Neural Networks - Basics

Gokberk Cinbis
Mini-batch SGD

Loop:
1. **Sample** a batch of data
2. **Forward** prop it through the graph, get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient
Parameter updates

We covered:
sgd,
momentum,
nag,
adagrad,
rmsprop,
adadelta (not in this vis),
we did not cover adadelta

Image credits: Alec Radford
Dropout

Forces the network to have a redundant representation.

- has an ear
- has a tail
- is furry
- has claws
- mischievous look
- cat score

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Introduction to ConvNets

[LeNet-5, LeCun 1980]
A bit of history:

Hubel & Wiesel, 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...

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Video time

https://youtu.be/8VdFf3egwfg?t=1m10s
A bit of history

Topographical mapping in the cortex: nearby cells in cortex represented nearby regions in the visual field
Hierarchical organization

Hubel & Weisel
- topographical mapping

Featural hierarchy
- hyper-complex cells
- complex cells
- simple cells

- high level
- mid level
- low level
A bit of history:

Neurocognitron

[Fukushima 1980]

“sandwich” architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling
A bit of history:
Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]

LeNet-5
A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]
Fast-forward to today: ConvNets are everywhere

Classification

Retrieval

[Krizhevsky 2012]
Fast-forward to today: ConvNets are everywhere

Detection

Segmentation

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

[Farabet et al., 2012]
Fast-forward to today: ConvNets are everywhere

self-driving cars

NVIDIA Tegra X1
Fast-forward to today: ConvNets are everywhere

[Simonyan et al. 2014]

[Goodfellow 2014]
Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]

[Mnih 2013]
Fast-forward to today: ConvNets are everywhere

[Ciresan et al. 2013]

[Sermanet et al. 2011]
[Ciresan et al.]
Fast-forward to today: ConvNets are everywhere

[Denil et al. 2014]

[Cs 559 Deep Learning]
Whale recognition, Kaggle Challenge

Mnih and Hinton, 2010
Image Captioning

[Vinyals et al., 2015]
reddit.com/r/deepdream

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition [Cadieu et al., 2014]
Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition

[Cadieu et al., 2014]
Details of Convolutional Neural Networks
Convolution Layer

32x32x3 image

width

height

depth
Convolution Layer

32x32x3 image

5x5x3 filter

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

Filters always extend the full depth of the input volume

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

32x32x3 image
5x5x3 filter $w$

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. $5\times5\times3 = 75$-dimensional dot product + bias)

$$w^T x + b$$
Convolution Layer

- 32x32x3 image
- 5x5x3 filter
- Convolve (slide) over all spatial locations

activation map
Convolution Layer

consider a second, green filter

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps
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!
**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions.

![Diagram of ConvNet](image)

- **CONV**
- **ReLU**
- e.g. 6 5x5x3 filters
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions.

- **CONV, ReLU** e.g. 6 5x5x3 filters
- **CONV, ReLU** e.g. 10 5x5x6 filters
- **CONV, ReLU**
[From recent Yann LeCun slides]

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Preview

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Hubel & Weisel

Featural hierarchy

[From recent Yann LeCun slides]
We call the layer convolutional because it is related to convolution of two signals:

\[ f[x,y] \ast g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2] \]

Elementwise multiplication and sum of a filter and the signal (image)
Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
A closer look at spatial dimensions:

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter

=> 5x5 output
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 2
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 2
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 2 => 3x3 output!
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 3?
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 3?

doesn’t fit!
cannot apply 3x3 filter on 7x7 input with stride 3.
Output size:
\((N - F) / \text{stride} + 1\)

e.g. \(N = 7, F = 3:\)
- \(\text{stride 1} \Rightarrow (7 - 3)/1 + 1 = 5\)
- \(\text{stride 2} \Rightarrow (7 - 3)/2 + 1 = 3\)
- \(\text{stride 3} \Rightarrow (7 - 3)/3 + 1 = 2.33\)
In practice: Common to zero pad the border

\[ \frac{(N - F)}{\text{stride}} + 1 \]

(e.g. input 7x7
3x3 filter, applied with \textbf{stride 1} pad \textbf{with 1 pixel} border => what is the output?)

(recall:)
\[ \frac{(N - F)}{\text{stride}} + 1 \]
In practice: Common to zero pad the border

Input 7x7, 3x3 filter, applied with stride 1, pad with 1 pixel border => what is the output?

7x7 output!
In practice: Common to zero pad the border

E.g. input 7x7

3x3 filter, applied with stride 1

Pad with 1 pixel border => what is the output?

7x7 output!

In general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)

E.g. F = 3 => zero pad with 1

F = 5 => zero pad with 2

F = 7 => zero pad with 3
Remember back to…
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn’t work well.
Examples time:

Input volume: \textbf{32x32x3}
10 5x5 filters with stride 1, pad 2

Output volume size: ?
Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

\[
\frac{(32+2*2-5)}{1}+1 = 32 \text{ spatially, so } 32x32x10
\]
Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
Examples time:

Input volume: \(32 \times 32 \times 3\)
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has \(5 \times 5 \times 3 + 1 = 76\) params (+1 for bias)
=> \(76 \times 10 = 760\)
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- \( F = 3, S = 1, P = 1 \)
- \( F = 5, S = 1, P = 2 \)
- \( F = 5, S = 2, P = ? \) (whatever fits)
- \( F = 1, S = 1, P = 0 \)
(btw, 1x1 convolution layers make perfect sense)

1x1 CONV with 32 filters

(each filter has size 1x1x64, and performs a 64-dimensional dot product)
Example: CONV layer in Torch

**SpatialConvolution**

```python
code
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW, dH, [padW, padH]])
```

Applies a 2D convolution over an input image composed of several input planes. The input tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane`: The number of expected input planes in the image given into `forward()`.
- `nOutputPlane`: The number of output planes the convolution layer will produce.
- `kW`: The kernel width of the convolution
- `kH`: The kernel height of the convolution
- `dW`: The step of the convolution in the width dimension. Default is 1.
- `dH`: The step of the convolution in the height dimension. Default is 1.
- `padW`: The additional zeros added per width to the input planes. Default is 0, a good number is `(kW-1)/2`.
- `padH`: The additional zeros added per height to the input planes. Default is `padW`, a good number is `(kH-1)/2`.

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

If the input image is a 3D tensor `nInputPlane x height x width`, the output image size will be `nOutputPlane x height x width` where

```latex
\begin{align*}
\text{oWidth} &= \text{floor}((\text{width} + 2\times \text{padW} - \text{kW}) / \text{dW} + 1) \\
\text{oHeight} &= \text{floor}((\text{height} + 2\times \text{padH} - \text{kH}) / \text{dH} + 1)
\end{align*}
```

**Summary**: To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Example: CONV layer in Caffe

```cpp
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  # learning rate and decay multipliers for the filters
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param { lr_mult: 2 decay_mult: 0 }
  convolution_param {
    num_output: 96 # learn 96 filters
    kernel_size: 11 # each filter is 11x11
    stride: 4 # step 4 pixels between each filter application
    weight_filler {
      type: "gaussian" # initialize the filters from a Gaussian
      std: 0.01 # distribution with stdev 0.01 (default mean: 0)
    } }" bias_filler {
      type: "constant" # initialize the biases to zero (0)
      value: 0
    } }
```

**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$
  - Their spatial extent $F$
  - The stride $S$
  - The amount of zero padding $P$

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
The brain/neuron view of CONV Layer

32x32x3 image
5x5x3 filter

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)
The brain/neuron view of CONV Layer

32x32x3 image
5x5x3 filter

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)
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”
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
The brain/neuron view of CONV Layer

Q: Why share parameters spatially and use local connectivity?
two more layers to go: POOL/FC
Pooling layer
- makes the representations smaller and more manageable
- operates over each activation map independently:
MAX POOLING

Single depth slice

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

max pool with 2x2 filters and stride 2

```
6 8
3 4
```
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent $F$,
  - the stride $S$,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers
Common settings:

- \( F = 2, \ S = 2 \)
- \( F = 3, \ S = 2 \)

- Accepts a volume of size \( W_1 \times H_1 \times D_1 \)
- Requires three hyperparameters:
  - their spatial extent \( F \),
  - the stride \( S \),
- Produces a volume of size \( W_2 \times H_2 \times D_2 \) where:
  - \( W_2 = (W_1 - F) / S + 1 \)
  - \( H_2 = (H_1 - F) / S + 1 \)
  - \( D_2 = D_1 \)
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers
Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks
[ConvNetJS demo: training on CIFAR-10]

http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html
Next: Case studies