CS5670: Computer Vision

Convolutional neural networks

Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Slides from Fei-Fei Li, Justin Johnson, Serena Yeung
http://vision.stanford.edu/teaching/cs231n/
Announcements

• Project 5 (NeRF): To be assigned this Thursday, April 20
  – Due Wednesday, May 3 at 8pm
• In-class final exam planned for the last day of class: Tuesday, May 9
• Sample final exam to be released soon
Readings

• Neural networks

• Convolutional neural networks
Recap: Image Classification – a core task in computer vision

- Assume given set of discrete labels, e.g. 
  \{\text{cat, dog, cow, apple, tomato, truck, ...}\}

\[
f(\text{apple}) = \text{“apple”} \\
f(\text{tomato}) = \text{“tomato”} \\
f(\text{cow}) = \text{“cow”}
\]
Recap: linear classification

- What we have: a score function and loss function
  - Score function maps an input data instance (e.g., an image) to a vector of scores, one for each category
  - Last time, our score function is based on linear classifier

\[ f(x, W) = Wx + b \]

- Find **W** and **b** that minimize a loss over labeled training data, e.g. cross-entropy loss

\[ L = \frac{1}{N} \sum_i - \log \left( \frac{e^{f_{yi}}}{\sum_j e^{f_j}} \right) \]
Linear classifiers separate features space into half-spaces

\[ f(x_i, W) = W x_i + b \]

Figure credit: Fei-Fei Li and Andrej Karpathy
Neural networks

(Before) Linear score function:  \( f = Wx \)

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Neural networks

(Before) Linear score function: \( f = Wx \)

(Now) 2-layer Neural Network

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

(Before) Linear score function:
\[ f = Wx \]

(Now) 2-layer Neural Network
\[ f = W_2 \max(0, W_1 x) \]

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Neural networks

(Before) Linear score function:
\[ f = Wx \]

(Now) 2-layer Neural Network
\[ f = W_2 \max(0, W_1 x) \]

• Total number of weights to learn:
\[ 3,072 \times 100 + 100 \times 10 = 308,200 \]

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Neural networks

(Before) Linear score function:
\[ f = Wx \]

(Now) 2-layer Neural Network or 3-layer Neural Network
\[ f = W_2 \max(0, W_1 x) \]
\[ f = W_3 \max(0, W_2 \max(0, W_1 x)) \]

also called “Multilayer Perceptrons” (MLPs)

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Neural networks

• Very coarse generalization of neural networks:
  – Linear functions chained together and separated by non-linearities (activation functions), e.g. “max”

\[ f = W_3 \max(0, W_2 \max(0, W_1 x)) \]

– Why separate linear functions with non-linear functions?
– Very roughly inspired by real neurons
Activation functions

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

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

ReLU
\[ \max(0, x) \]

Leaky ReLU
\[ \max(0.1x, x) \]

Maxout
\[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

ELU
\[ \begin{cases} 
  x & x \geq 0 \\
  \alpha(e^x - 1) & x < 0 
\end{cases} \]
Neural network architecture

• Computation graph for a 2-layer neural network

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
• **Deep** networks typically have many layers and potentially millions of parameters

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Deep neural network

- *Inception* network (Szegedy et al, 2015)
- 22 layers

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

- Just like a linear classifier – but in this case, just one layer of a larger *network*

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Summary so far

• A classic neural network arranges neurons into fully-connected layers
• The **layer** abstraction enables efficient implementations of neural networks using vectorized operations like matrix multiplication

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Leung
Optimizing parameters with gradient descent

• How do we find the best $W$ and $b$ parameters?

• In general: gradient descent
  1. Start with a guess of a good $W$ and $b$ (or randomly initialize them)
  2. Compute the loss function for this initial guess and the gradient of the loss function
  3. Step some distance in the negative gradient direction (direction of steepest descent)
  4. Repeat steps 2 & 3

• Note: efficiently performing step 2 for deep networks is called backpropagation
Gradient descent: walk in the direction opposite gradient

- **Q**: How far?
- **A**: Step size: *learning rate*
  - Too big: will miss the minimum
  - Too small: slow convergence
2D example of gradient descent

• In reality, in deep learning we are optimizing a highly complex loss function with millions of variables (or more)

• More on this later...

2D example: TensorFlow Playground

Tinker With a Neural Network Right Here in Your Browser. Don’t Worry, You Can’t Break It. We Promise.

https://playground.tensorflow.org
Questions?
Convolutional neural networks (or CNNs, or ConvNets)
A bit of history...

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

\[
f(x) = \begin{cases} 
1 & \text{if } w \cdot x + b > 0 \\
0 & \text{otherwise} 
\end{cases}
\]

recognized
letters of the alphabet

update rule:

\[w_i(t + 1) = w_i(t) + \alpha (d_j - y_j(t))x_{j,i}\]

Frank Rosenblatt, ~1957: Perceptron
A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning

A bit of history:
Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]
First strong results

**Acoustic Modeling using Deep Belief Networks**
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010

**Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition**
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

**Imagenet classification with deep convolutional neural networks**

A bit of history:
ImageNet Classification with Deep Convolutional Neural Networks
[Krizhevsky, Sutskever, Hinton, 2012]

"AlexNet"

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Fast-forward to today: *ConvNets* are everywhere

* and other recent architectures, like *Transformers*


Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Fast-forward to today: ConvNets are everywhere

Detection

Segmentation

Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

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


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

Self-driving cars (video courtesy Tesla)
https://www.tesla.com/AI

NVIDIA Tesla A6000 Ada GPU

Cloud TPU v4 Pods
https://cloud.google.com/tpu/
Fast-forward to today: ConvNets are everywhere

[Levy et al. 2016]

[Dieleman et al. 2014]

[Seremanet et al. 2011]
[Ciresan et al.]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
No errors

- A white teddy bear sitting in the grass

Minor errors

- A man in a baseball uniform throwing a ball
- A cat sitting on a suitcase on the floor

Somewhat related

- A woman is holding a cat in her hand
- A woman standing on a beach holding a surfboard

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]
Text-to-image

An astronaut  Teddy bears  A bowl of soup

riding a horse  lounging in a tropical resort in space  playing basketball with cats in space

in a photorealistic style  in the style of Andy Warhol  as a pencil drawing

https://openai.com/dall-e-2/
“A computer vision class watching a cool lecture, crayon drawing”
Stable Diffusion XL

Create and inspire using the worlds fastest growing open source AI platform.

With Stable Diffusion XL, you can create descriptive images with shorter prompts and generate words within images. The model is a significant advancement in image generation capabilities, offering enhanced image composition and face generation that results in stunning visuals and realistic aesthetics.

SDXL is currently in beta on DreamStudio and other leading imaging applications. Like all of Stability AI’s foundation models, SDXL will be released as open source for optimal accessibility in the near future.

DreamStudio
What is a ConvNet?

• Version of deep neural networks designed for signals
  – 1D signals (e.g., speech waveforms)
  – 2D signals (e.g., images)
Motivation – Feature Learning
Life Before Deep Learning

Input Pixels → Extract Hand-Crafted Features → Concatenate into a vector $\mathbf{x}$ → Linear Classifier → SVM → Ans

Figure: Karpathy 2016
Why use features? Why not pixels?

Q: What would be a very hard set of classes for a linear classifier to distinguish?

(assuming $x = \text{pixels}$)

$$f(x_i, W) = W x_i + b$$
Goal: linearly separable classes

\[ f(x_i, W) = W x_i + b \]
Aside: Image Features

\[ f(x) = Wx \]

Feature Representation

Class scores

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Image Features: Motivation

Cannot separate red and blue points with linear classifier
Image Features: Motivation

\[ f(x, y) = (r(x, y), \theta(x, y)) \]

Cannot separate red and blue points with linear classifier

After applying feature transform, points can be separated by linear classifier

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Example: Color Histogram
Example: Histogram of Oriented Gradients (HoG)

Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins

Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, "Object recognition from local scale-invariant features," ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005
Image features vs ConvNets

Feature Extraction

10 numbers giving scores for classes

training

10 numbers giving scores for classes

training

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Last layer of many CNNs is a linear classifier

This piece is just a linear classifier

Input Pixels

Perform everything with a big neural network, trained end-to-end

Key: perform enough processing so that by the time you get to the end of the network, the classes are linearly separable
Visualizing AlexNet in 2D with t-SNE

(2D visualization using t-SNE)

(c) DeCAF\(_1\)

(d) DeCAF\(_6\)

Convolutional neural networks

Layer types:
- *Convolutional layer*
- Pooling layer
- Fully-connected layer
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

input

1

3072

$Wx$

10 x 3072 weights

activation

1

10

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

input

1

3072

\[ Wx \]

10 x 3072 weights

activation

1

10

1 number:
the result of taking a dot product between a row of W and the input
(a 3072-dimensional dot product)

Same as a linear classifier!

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Convolution Layer

32x32x3 image -> preserve spatial structure
Convolution Layer

A 32x32x3 image is convolved with a 5x5x3 filter. The process involves sliding the filter over the image spatially, computing dot products at each location.
Convolution Layer

32x32x3 image

5x5x3 filter

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

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

Number of weights: $5 \times 5 \times 3 + 1 = 76$
(vs. 3072 for a fully-connected layer)
(+1 for bias term)

Adapted from Fei-Fei Li & Andrej Karpathy & Serena Lee
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

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Convolution Layer

Consider a second, green filter

32x32x3 image
5x5x3 filter

Convolve (slide) over all spatial locations

Activation maps

32 32
3 32

28 28
1 28

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
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!

(total number of parameters to learn: $6 \times (75 + 1) = 456$)
How many parameters are in a convolution layer consisting of 3 3x3 filters (each with bias term)?
**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions.

- CONV, ReLU
  - e.g. 6
  - 5x5x3 filters

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions.
Preview

[Zeiler and Fergus 2013]

Visualisation of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].
A closer look at spatial dimensions:

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations
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:
\[
\frac{(N - F)}{\text{stride}} + 1
\]

E.g. \(N = 7, F = 3\):

- Stride 1 \(\Rightarrow \frac{(7 - 3)}{1} + 1 = 5\)
- Stride 2 \(\Rightarrow \frac{(7 - 3)}{2} + 1 = 3\)
- Stride 3 \(\Rightarrow \frac{(7 - 3)}{3} + 1 = 2.33\)
In practice: Common to zero pad the border

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E.g. input 7x7
3x3 filter, applied with **stride 1**
**pad with 1 pixel** border => what is the output?

(recall:)

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

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
Examples time:

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

Output volume size: ?
Examples time:

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

Output volume size:
$(32+2\times2-5)/1+1 = 32$ spatially, so $32 \times 32 \times 10$
Examples time:

Input volume: $32 \times 32 \times 3$
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)
=> $76 \times 10 = 760$
“1x1 convolutions”

1x1 CONV with 32 filters

(each filter has size 1x1x64, and performs a 64-dimensional dot product)
Convolutional layer—properties

- Small number of parameters to learn compared to a fully connected layer
- Preserves spatial structure—output of a convolutional layer is shaped like an image
- **Translation equivariant**: passing a translated image through a convolutional layer is (almost) equivalent to translating the convolution output (but be careful of image boundaries)
Pooling layer
- makes the representations smaller and more manageable
- operates over each activation map independently:
MAX POOLING

Single depth slice

max pool with 2x2 filters and stride 2

Slide credit: Fei-Fei Li & Andrej Karpathy & Serena Leung
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]

ConvNetJS CIFAR-10 demo

Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and vertically.

By default, in this demo we’re using Adadelta which is one of per-parameter adaptive step size methods, so we don’t have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you’d like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html
AlexNet (2012)

6M parameters in total

Input image (RGB)

Max pooling

Stride of 4

224

11

11

55

55

5

96

224

3

CONV1

CONV2

CONV3

CONV4

CONV5

Max pooling

Max pooling

5 Convolution layers

3 Fully-connected layers

Output: 1,000-D vector (probabilities over 1,000 ImageNet categories)

“AlexNet” [Krizhevsky et al. NIPS 2012]  
“GoogLeNet” [Szegedy et al. CVPR 2015]  
“VGG Net” [Simonyan & Zisserman, ICLR 2015]  
“ResNet” [He et al. CVPR 2016]
Big picture

• A convolutional neural network can be thought of as a function from images to class scores
  – With millions of adjustable weights...
  – ... leading to a very non-linear mapping from images to features / class scores.
  – We will set these weights based on classification accuracy on training data...
  – ... and hopefully our network will generalize to new images at test time
Data is key—enter ImageNet

• ImageNet (and the ImageNet Large-Scale Visual Recognition Challenge, aka **ILSVRC**) has been key to training deep learning methods

• **ILSVRC**: 1,000 object categories, each with ~700-1300 training images. Test set has 100 images per categories (100,000 total).

• Standard ILSVRC error metric: top-5 error
  – if the correct answer for a given test image is in the top 5 categories, your answer is judged to be correct
Performance improvements on ILSVRC

• ImageNet Large-Scale Visual Recognition Challenge
• Held from 2011-2017
• 1000 categories, 1000 training images per category
• Test performance on held-out test set of images
Questions?