Deep Learning
Deep learning with convolutional neural networks in computer vision
Classical Computer Vision Pipeline.

CV experts
1. Select / develop features: SURF, HoG, SIFT, RIFT, …
2. Add on top of this Machine Learning for multi-class recognition and train classifier

Classical CV feature definition is domain-specific and time-consuming

http://courses.cs.tau.ac.il/Caffe_workshop/Bootcamp/Lecture 1 CNN introduction.pptx
Deep Learning – based Vision Pipeline.

Deep Learning:
• Build features automatically based on training data
• Combine feature extraction and classification

DL experts: define NN topology and train NN

Deep Learning promise:
train good feature automatically,
same method for different domain

http://courses.cs.tau.ac.il/Caffe_workshop/Bootcamp/Lecture 1 CNN introduction.pptx
Goal

• Learn to create good, hierarchical features for visual input!

• Such features improve classification

• Convolutional networks learn to generate good features
Why use hierarchical multi-layered models?

Argument 1: visual scenes are hierarchically organised

- **Object**
  - Object parts
  - Primitive features
  - Input image

- **Trees**
  - Bark, leaves, etc.
  - Oriented edges
  - Forest image

From [http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/ml-lecture-3-slides.pdf](http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/ml-lecture-3-slides.pdf)
Why use hierarchical multi-layered models?

Argument 2: biological vision is hierarchically organised

- Object
  - Object parts
    - Primitive features
      - Input image
  - Trees
    - Bark, leaves, etc.
      - Oriented edges
        - Forest image
          - Photo-receptors retina
  - Inferotemporal cortex
    - V4: different textures
      - V1: simple and complex cells

From http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/ml-lecture-3-slides.pdf
Why use hierarchical multi-layered models?

Argument 3: shallow architectures are inefficient at representing deep functions

single layer neural network implements: \( x = f_\theta(z) \)

networks we met last lecture with large enough single hidden layer can implement any function 'universal approximator'

shallow networks can be computationally inefficient

however, if the function is 'deep' a very large hidden layer may be required
Summary

- higher level layers encode more **abstract features**
- higher level layers show more **invariance to instantiation parameters**
  - translation
  - rotation
  - lighting changes
- a method for **learning feature detectors**
  - first layer learns edge detectors
  - subsequent layers more complex
  - integrates training of the classifier with training of the featural representation

From [http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/ml-lecture-3-slides.pdf](http://learning.eng.cam.ac.uk/pub/Public/Turner/Teaching/ml-lecture-3-slides.pdf)
Large Scale Visual Recognition Challenge (ILSVRC) 2010-2012

1000 object classes
1,431,167 images


http://ai.stanford.edu/~olga/slides/ImageNetAnalysis_bavm_10_5_13.pptx
Variety of object classes in ILSVRC

**PASCAL**

- birds: bird
- bottles: bottle
- cars: car

**ILSVRC**

- flamingo
- cock
- ruffed grouse
- quail
- partridge
- pill bottle
- beer bottle
- wine bottle
- water bottle
- pop bottle
- race car
- wagon
- minivan
- jeep
- cab
English foxhound

An English breed slightly larger than the American foxhounds originally used to hunt in packs.

Survive (0)
- range animal (0)
- creepy-crawly (0)
- domestic animal, domesticated animal (213)
  - domestic cat, house cat, Felis domesticus, Felis catus (18)
  - dog, domestic dog, Canis familiaris (189)
    - pooch, doggie, doggy, barker, bow-wow (0)
      - hunting dog (101)
        - sporting dog, gun dog (28)
        - dachshund, dachsie, badger dog (1)
        - terrier (37)
        - courser (0)
      - hound, hound dog (29)
        - Plott hound (0)
        - wolfhound (2)
        - Scottish deerhound, deerhound (0)
        - coonhound (2)
        - foxhound (3)
          - Walker hound, Walker foxhound (0)
          - American foxhound (0)
          - English foxhound (0)
          - Weimaraner (0)
          - otterhound, otter hound (0)
          - bloodhound, sleuthhound (0)
          - Norwegian elkhound, elkhound (0)
          - Saluki, gazelle hound (0)
          - Afghan hound, Afghan (0)
          - staghound (0)
          - greyhound (2)
          - beagle (0)
          - harrier (0)
          - basset, basset hound (0)
          - bluetick (0)
          - redbone (0)
ILSVRC top-5 error on ImageNet

- Blue: Traditional CV
- Purple: Deep Learning
- Red: Human

http://image.slidesharecdn.com/foundersdevelopers-dl-160403174257/95/deep-learning-cases-text-and-image-processing-6-638.jpg?cb=1459705535
Building Blocks of Convolutional Networks (ConvNets)

Input

Convolution Layer

Nonlinearity

Pooling Layer

Fully Connected Layers

Softmax Layer

Probability Distribution over Classes

Repeat
Building Blocks of Convolutional Networks (ConvNets)

- Input
- Convolution Layer
- Nonlinearity
- Pooling Layer
- Repeat...
- Fully Connected Layers
- Softmax Layer
- Probability Distribution over Classes

Feature Extraction

Classification
Convolution Layer

32x32x3 image

32 height
32 width
3 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

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map

Fei-Fei Li & Andrej Karpathy & Justin Johnson  Lecture 7 - 14  27 Jan 2016
Consider a second, green filter.

Convolve (slide) over all spatial locations.

32x32x3 image
5x5x3 filter

Activation maps

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 7 - 15
27 Jan 2016
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!
Nonlinear activation functions:

- **Sigmoid**
- **Rectified Linear Unit (ReLU)**

Benefits of ReLU vs. Sigmoid: (1) ReLU provides *sparse* activations; (2) ReLU avoids *vanishing gradients*; (3) Very amenable to GPUs
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

Fei-Fei Li & Andrej Karpathy & Justin Johnson  Lecture 7 - 55  27 Jan 2016
Fully Connected Layer (FC layer)
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks
Case Study: ResNet [He et al., 2015]
ILSVRC 2015 winner (3.6% top 5 error)

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)  VGG, 19 layers (ILSVRC 2014)  ResNet, 152 layers (ILSVRC 2015)

2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

Fei-Fei Li & Andrej Karpathy & Justin Johnson  Lecture 7 - 80  27 Jan 2016
Revolution of Depth

ImageNet Classification top-5 error (%)


(slide from Kaiming He’s recent presentation)
How to Understand ConvNets?
Visualize the filters/kernels (raw weights)

only interpretable on the first layer :(

Fei-Fei Li & Andrej Karpathy & Justin Johnson  Lecture 9   8   3 Feb 2016
Visualizing the representation

4096-dimensional “code” for an image (layer immediately before the classifier)

can collect the code for many images
Visualizing the representation

t-SNE visualization
[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places, dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D
t-SNE visualization:

two images are placed nearby if their CNN codes are close. See more:

http://cs.stanford.edu/people/karpathy/cnnembed/
Visualizing Activations

http://yosinski.com/deepvis

YouTube video
https://www.youtube.com/watch?v=AgkflQ41GaM (4min)
Q: can we find an image that maximizes some class score?
Optimization to Image

1. feed in zeros.

2. set the gradient of the scores vector to be \([0, 0, \ldots, 1, \ldots, 0]\), then backprop to image

3. do a small “image update”

4. forward the image through the network.

5. go back to 2.

\[
\arg \max_I \left[ S_c(I) - \lambda \| I \|_2^2 \right]
\]

score for class c (before Softmax)

From http://cs231n.stanford.edu/
1. Find images that maximize some class score:
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

1. Find images that maximize some class score:

- Washing machine
- Computer keyboard
- Kit fox
- Goose
- Ostrich
- Limousine
Proposed a different form of regularizing the image

\[
\arg\max_I S_c(I) - \lambda \|I\|_2^2
\]

More explicit scheme:

Repeat:
- Update the image \( \mathbf{x} \) with gradient from some unit of interest
- Blur \( \mathbf{x} \) a bit
- Take any pixel with small norm to zero (to encourage sparsity)
[Understanding Neural Networks Through Deep Visualization, Yosinski et al., 2015]

http://yosinski.com/deepvis

From http://cs231n.stanford.edu/

Fei-Fei Li & Andrej Karpathy & Justin Johnson  Lecture 9 - 40  3 Feb 2016
From http://cs231n.stanford.edu/
Question: Given a CNN code, is it possible to reconstruct the original image?
Find an image such that:
- Its code is similar to a given code
- It “looks natural” (image prior regularization)

\[
x^* = \underset{x \in \mathbb{R}^{H \times W \times C}}{\text{argmin}} \, \ell(\Phi(x), \Phi_0) + \lambda \mathcal{R}(x)
\]

\[
\ell(\Phi(x), \Phi_0) = \|\Phi(x) - \Phi_0\|^2
\]
Understanding Deep Image Representations by Inverting Them
[Mahendran and Vedaldi, 2014]

original image

reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes
Reconstructions from the representation after the last pooling layer (immediately before the first Fully Connected layer)
Multiple reconstructions. Images in quadrants all “look” the same to the CNN (same code)
Google “Deep Dream”

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
NeuralStyle

[A Neural Algorithm of Artistic Style by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, 2015]
good implementation by Justin in Torch:
https://github.com/jcjohnson/neural-style
make your own easily on deepart.io
Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)

e.g. at CONV5_1 layer we would have a [14x14x512] array of target activations
Step 2: Extract style targets (Gram matrices of ConvNet activations of all layers for the given style image)

style gram matrices

\[ G = V^T V \]

e.g., at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)
Step 3: Optimize over image to have:
- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

\[
L_{total}(\tilde{p}, \tilde{a}, \tilde{x}) = \alpha L_{content}(\tilde{p}, \tilde{x}) + \beta L_{style}(\tilde{a}, \tilde{x})
\]

(+Total Variation regularization (maybe))
“To generate the images that mix the content of a photograph with the style of a painting, we jointly minimize the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN.“
“To generate the images that mix the content of a photograph with the style of a painting, we jointly minimize the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN."

\[ L_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2. \]

Activation of \( i \)th filter at position \( j \) in layer \( l \) for generated image

Same, but for original content image
“To generate the images that mix the content of a photograph with the style of a painting, we jointly minimize the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN.

\[
\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 .
\]

Activation of \( i \)th filter at position \( j \) in layer \( l \) for generated image

\[
E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2
\]

Inner product between feature map \( i \) and \( j \) in layer \( l \) for generated image

\[
\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l
\]

Same, but for original style image
“To generate the images that mix the content of a photograph with the style of a painting, we jointly minimize the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN.“

\[
\mathcal{L}_{\text{content}}(\tilde{p}, \tilde{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 .
\]

Activation of \(i\)th filter at position \(j\) in layer \(l\) for generated image

\[
E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2
\]

Inner product between feature map \(i\) and \(j\) in layer \(l\) for generated image

\[
\mathcal{L}_{\text{style}}(\tilde{a}, \tilde{x}) = \sum_{l=0}^{L} w_l E_l
\]

Same, but for original content image

\[
\mathcal{L}_{\text{total}}(\tilde{p}, \tilde{a}, \tilde{x}) = \alpha \mathcal{L}_{\text{content}}(\tilde{p}, \tilde{x}) + \beta \mathcal{L}_{\text{style}}(\tilde{a}, \tilde{x})
\]
We can pose an optimization over the input image to maximize any class score. That seems useful.

Question: Can we use this to “fool” ConvNets?

spoiler alert: yeah
[Intriguing properties of neural networks, Szegedy et al., 2013]
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images]

Nguyen, Yosinski, Clune, 2014

>99.6% confidences

- robin
- cheetah
- armadillo
- lesser panda
- centipede
- peacock
- jackfruit
- bubble
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
Nguyen, Yosinski, Clune, 2014]

>99.6% confidences

king penguin  starfish  baseball  electric guitar
freight car  remote control  peacock  African grey
“Deep Neural Networks are Easily Fooled”

Nguyen et al., 2015

https://www.youtube.com/watch?v=M2IebCN9Ht4
Adversarial Examples
Figure 5: Adversarial examples generated for AlexNet [9]. (Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an “ostrich, Struthio camelus”. Average distortion based on 64 examples is 0.006508. Please refer to http://goo.gl/huaGPb for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.
How to find adversarial perturbations

Image: \( x \in \mathbb{R}^m \)

Label: \( l \in \{1 \ldots k\} \)

Minimize \( \| r \|_2 \) subject to \( f(x + r) = l \)

\( x + r \) is the closest image to \( x \) classified as \( l \) by \( f \).
“Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition”

Figure 5: The eyeglass frames used by $S_C$ for dodging recognition against $DNN_B$. 
Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows $S_A$ (top) and $S_B$ (bottom) dodging against $DNN_B$. Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows $S_A$ impersonating Milla Jovovich (by Georges Biard; source: https://goo.gl/GlsW1C); (c) $S_B$ impersonating $S_C$; and (d) $S_C$ impersonating Carson Daly (by Anthony Quintano; source: https://goo.gl/VfnDct).