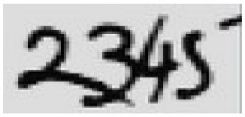
Elements in Convolutional Neural Networks

1

CONV NETS: EXAMPLES

- **OCR / House number & Traffic sign classification**



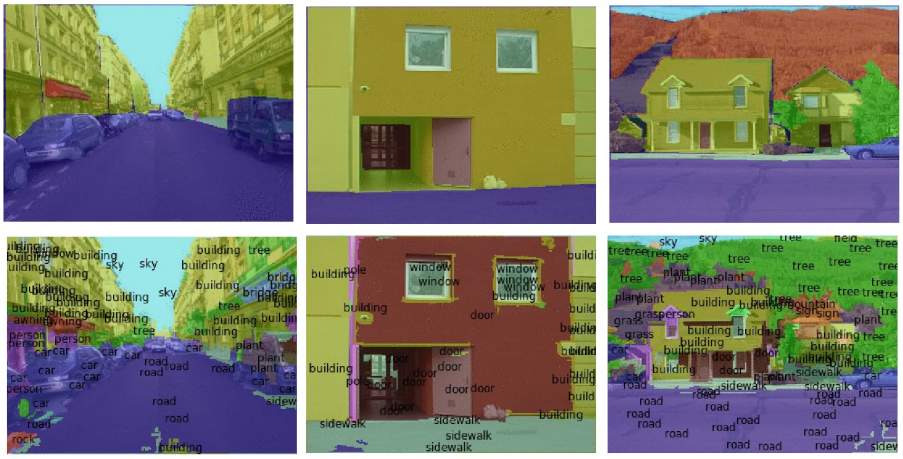
Ciresan et al. "MCDNN for image classification" CVPR 2012
Wan et al. "Regularization of neural networks using dropconnect" ICML 2013
Jaderberg et al. "Synthetic data and ANN for natural scene text recognition" arXiv 2014

82

2

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

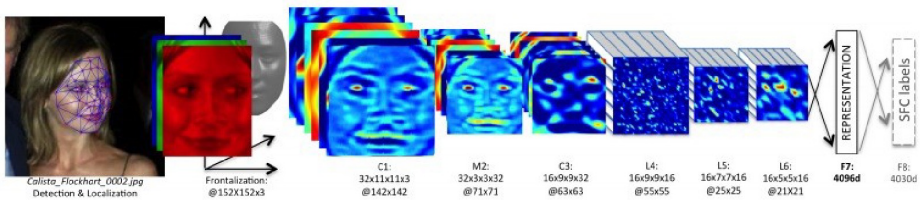
85

Ranzato

3

CONV NETS: EXAMPLES

- **Face Verification & Identification**



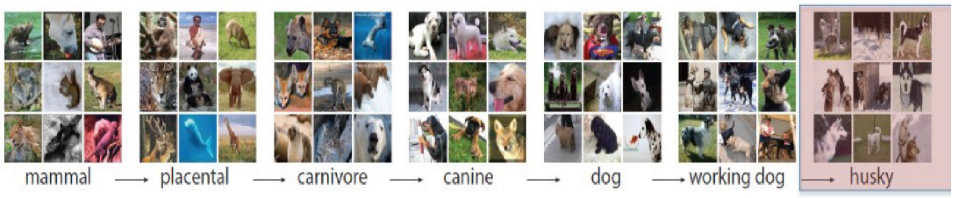
Taigman et al. "DeepFace..." CVPR 2014

92

Ranzato

4

Dataset: ImageNet 2012



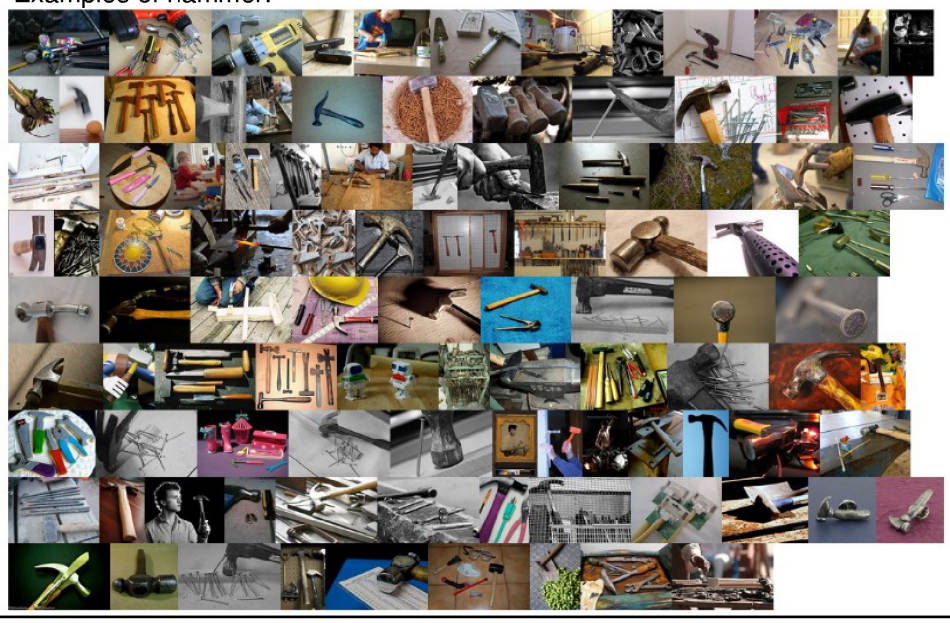
- S: (n) Eskimo dog, husky (breed of heavy-coated Arctic sled dog)
  - *direct hypernym / inherited hypernym / sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, Canis familiaris (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) *"this dog barked all night"*
        - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
          - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
            - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
              - S: (n) mammal, mammalian (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: (n) 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: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                    - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
                      - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                        - S: (n) living thing, animate thing (a living (or once living) entity)
                          - S: (n) 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: (n) 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: (n) physical entity (an entity that has physical existence)
                                - S: (n) 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

5

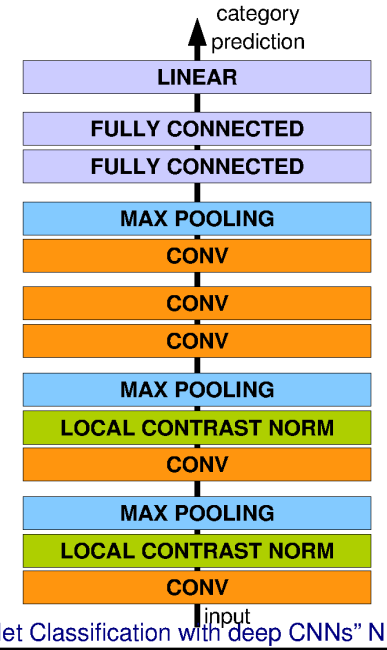
ImageNet

Examples of hammer:



6

Architecture for Classification



Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

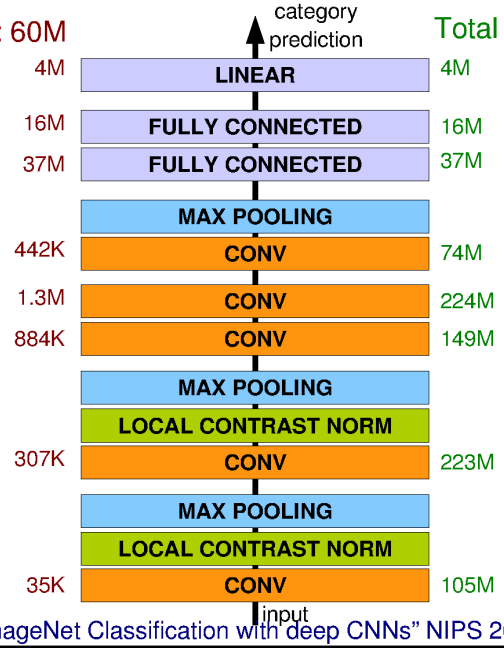
95 Ranzato

7

Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

96 Ranzato

8

Optimization

SGD with momentum:

- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

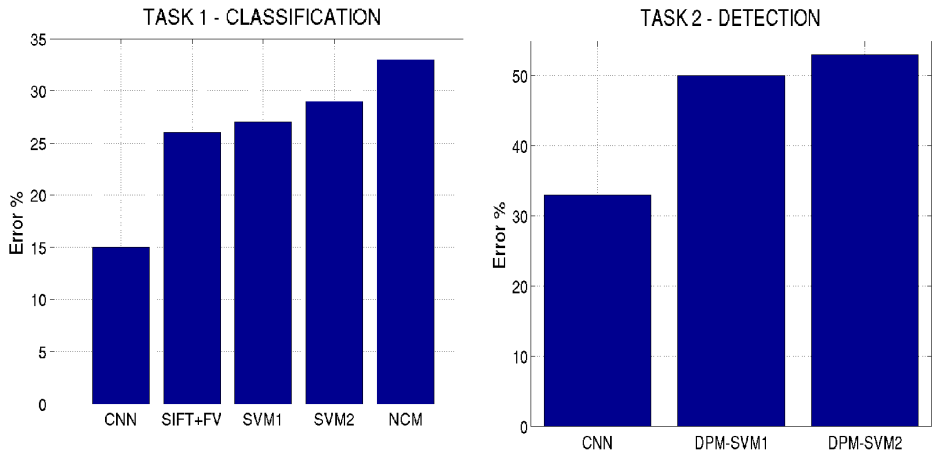
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

97

Ranzato

9

Results: ILSVRC 2012

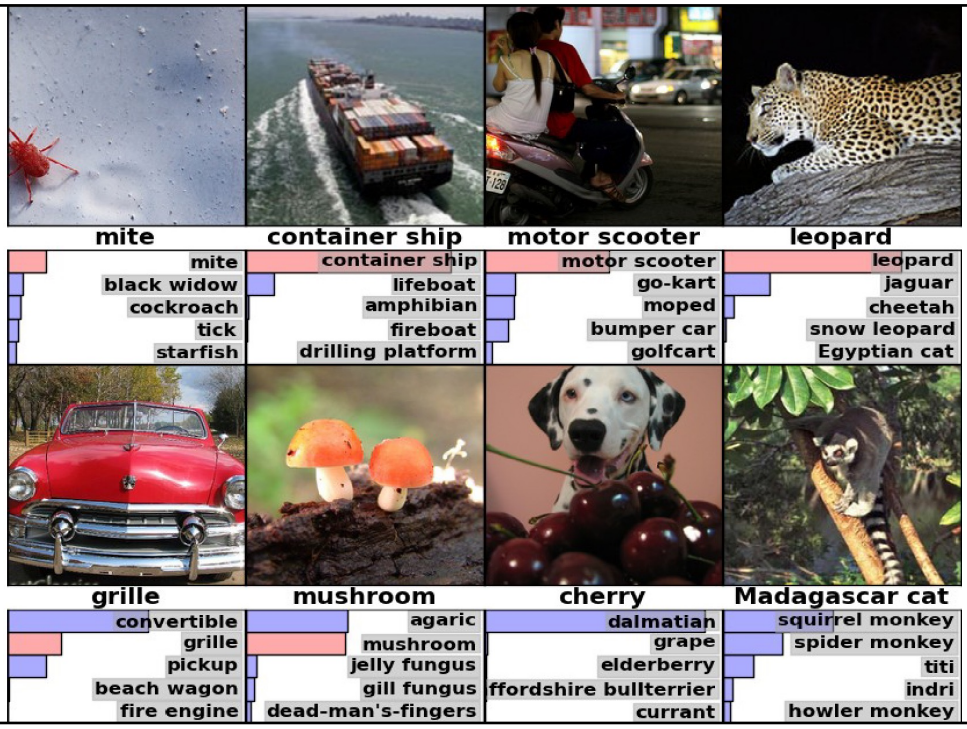


98

Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

Ranzato

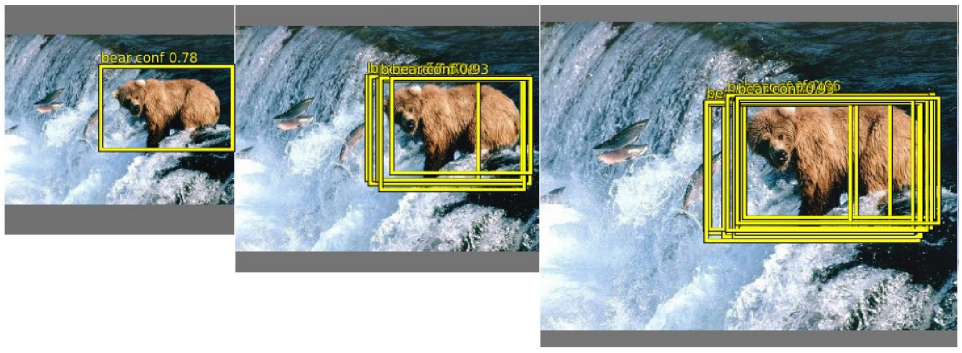
10



11

CONV NETS: EXAMPLES

- **Object detection**



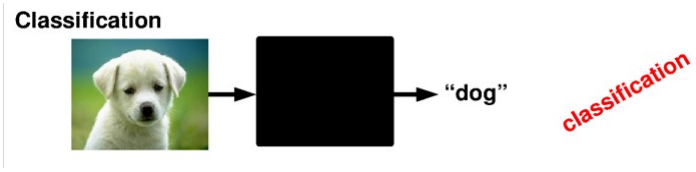
Sermanet et al. "OverFeat: Integrated recognition, localization, ..." arxiv 2013
Girshick et al. "Rich feature hierarchies for accurate object detection..." arxiv 2013
Szegedy et al. "DNN for object detection" NIPS 2013

91

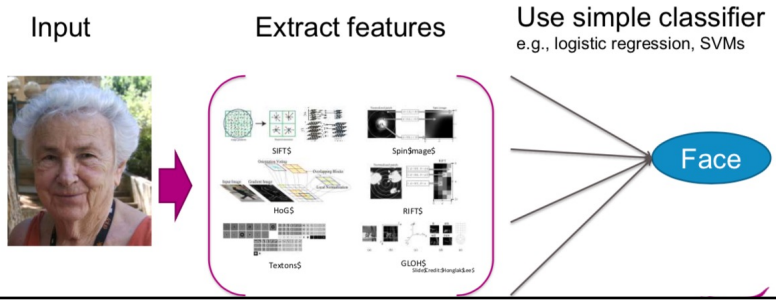
Ranzato

12

The problem: image classification

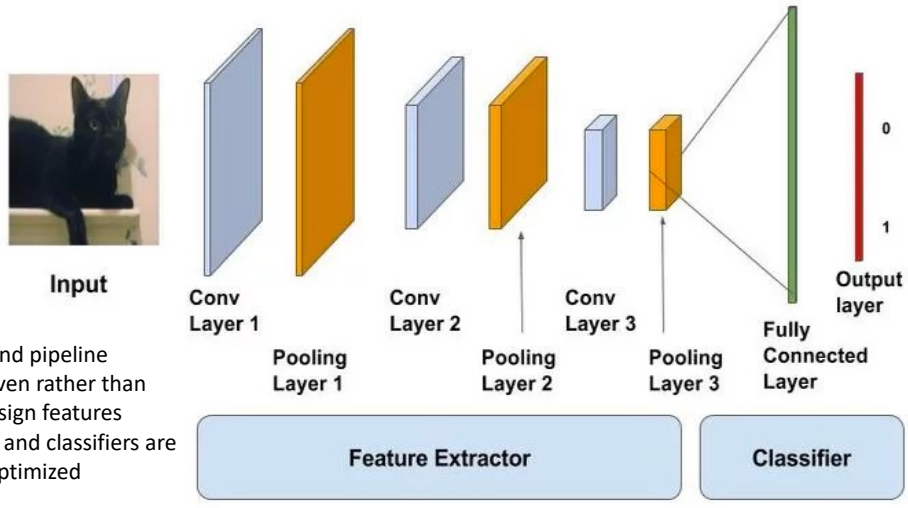


before learning:



13

Convolutional Neural Networks for image classification



- End to end pipeline
- data-driven rather than hand design features
- features and classifiers are jointly optimized

14

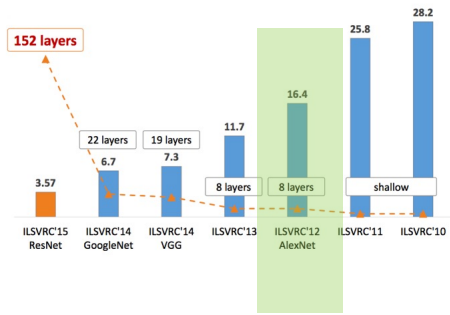
today's outline

- Why does AlexNet work (as opposed to previous work [LeCun 98])?
- Training your AlexNet
- Revolutions of AlexNet
  - VGG, ResNet (ResNexT), GoogLeNet
- Modifying CNNs for object detection*
  - R-CNN, Faster-RCNN(Mask-RCNN), SSD(Yolo)

15

AlexNet

- First CNN that works on ImageNet
  - Reduce errors from 25.8 to 16.4
  - Not much tricks used



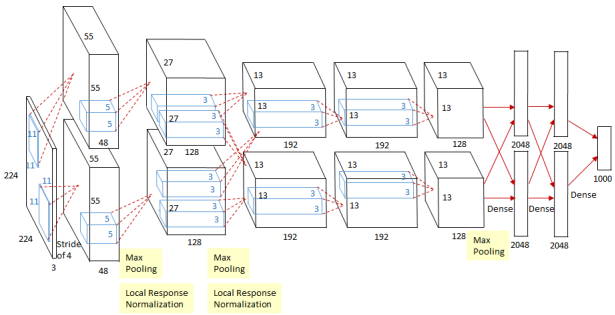
[PDF] ImageNet Classification with Deep Convolutional Neural ...
https://papers.nips.cc › paper › 4824-imagenet-classification-with-deep-co...
by A Krizhevsky - 2012 - Cited by 48095 - Related articles
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 dif-.
You've visited this page many times. Last visit: 10/19/19

People also search for
alexnet matlab zfnet
alexnet keras googlenet paper
alexnet github vggnet

16

Why is AlexNet working?

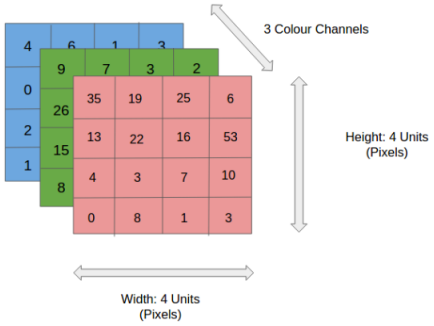
- Convolutional Layers (LeCun, 1998)
- Rectified Linear Units (Activation Function)
- Local Response Normalization
- Pooling
- Data Augmentation
- Dropout (overfitting prevention)
- GPU Training



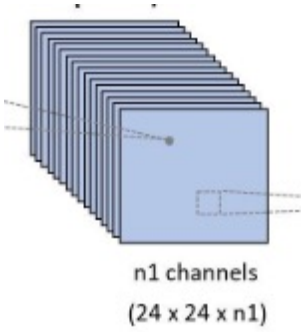
17

Convolutional Layers

Input: [(N,) H, W, C]. if input is an image C=3 (R, G, B)



Image



Intermediate layers. (C = n1)
each channel represents response to a specific convolutional kernel*

18

Convolutional Layers

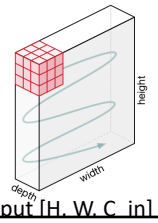
parameters: [k_w, k_h, C_in, C_out]

input [H, W, C_in]. (Here we assume a batch size of 1 for simplicity, otherwise[N, H, W, C_in])

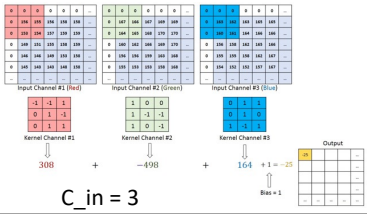
output [(N,) H', W', C_out]

how it works:

- sliding [H, W, C_in] into small regions [k_w, k_h, C_in], there will be total [H', W'] of these regions
- each [k_w, k_h, C_in] dot product with a convolutional kernel [k_w, k_h, C_in, 1] → [1, 1], the response map will be [H', W', 1]
- there is a total of C_out convolution kernel, stack together to [H', W', C_out]



Input [H, W, C_in]



C_in = 3

19

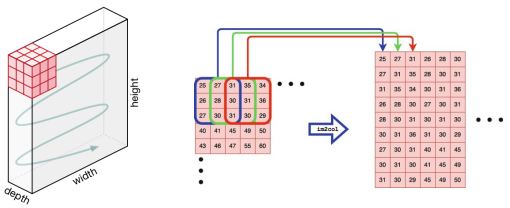
Implementing convolutional layers

parameters: [k_w, k_h, C_in, C_out]
input: [H, W, C_in]
output: [H', W', C_out]

implementation:

- sliding [H, W, C_in] into small region [k_w, k_h, C_in], there will be total [H', W'] of these regions. implementation
  - im2col
  - [H, W, C_in]. → [H' x W', k_w x k_h x C_in]. (2D matrix)
- each [k_w, k_h, C_in] dot product with a convolutional kernel [k_w, k_h, C_in, 1] → [1, 1], the response map will be [H',W', 1]
  - conv
  - matrix multiplication [H' x W', k_w x k_h x C_in] x [k_w x k_h x C_in, C_out] → [H' x W', C_out] (2D)
- there is a total of C_out convolution kernel, stack together to [H', W', C_out]
  - reshape
  - [H' x W', C_out] → [H', W', C_out]

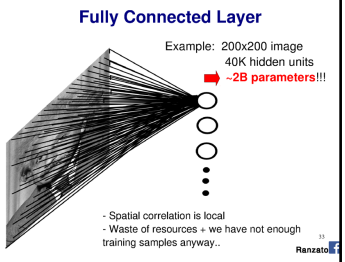
convolution is essentially a matrix multiplication over multiple regions

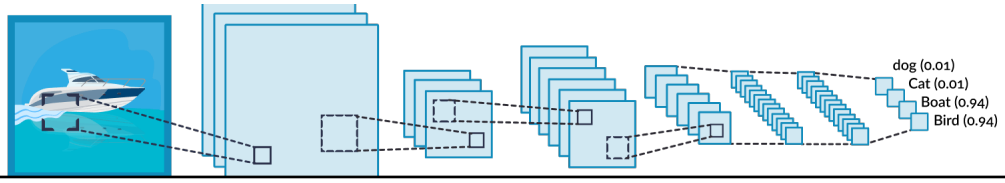


20

Convolutional Layers

- Why convolutional layers work?
  - reduces the number of parameters
  - exploit feature locality.
  - maintain spatial structure
  - A stack of CNNs do it at different granularities (receptive field), therefore being able to model *hierarchically* higher level features.





21

Convolutional Layers: what we didn't cover

- how we get [H', W'] from [H, W]
  - Padding
  - convolutional size [k_w, k_h]
  - striding
- group convolution
- transposed convolution (deconvolution)

for more details, go

A guide to convolution arithmetic for deep learning

Vincent Dumoulin[1]* and Francesco Visin[2]*[†]

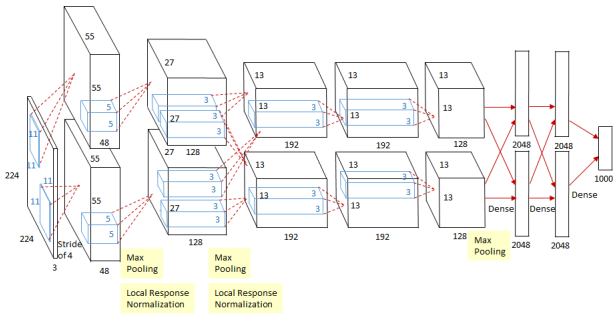
*MILA, Université de Montréal
[†]AIRLab, Politecnico di Milano

January 12, 2018

22

Why is AlexNet working?

- ~~Convolutional Layers (LeCun, 1998)~~
- Rectified Linear Units (Activation Function)
- Local Response Normalization
- Pooling
- Data Augmentation
- Dropout (overfitting prevention)
- GPU Training



23

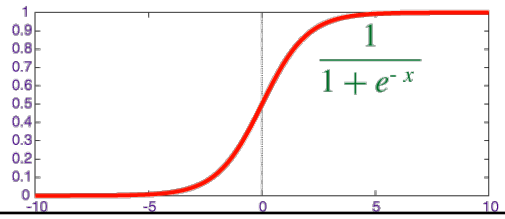
Activation Function

- Why do we need activation function (to add non-linearity)
  - if not using activation function, all transformations will be linear
  - example:
    - have two layers y = W_1 * W_2* x
    - if not using activation function y = W' * x where W' = W_1 * W_2
    - there is no need to use multiple layer
    - representative power is low
  - activation functions add non-linearly → improves representation power

24

Activation Function

- Per-element
  - x_out = f (x_input)
  - doesn't change input size
- Sigmoid:
  - very widely used in early work
  - biologically inspired
  - one of the major reasons making training deep neural networks difficult

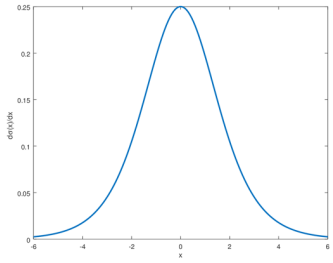


25

Why sigmoid not working with deep networks

derivatives

$$\begin{aligned}\frac{d\sigma(x)}{dx} &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right)\\ &= \frac{d}{dx}(1+e^{-x})^{-1}\\ &= -(1+e^{-x})^{-2}\cdot(-e^{-x})\\ &= \frac{e^{-x}}{(1+e^{-x})^2}\\ &= \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}}\\ &= \sigma(x)\cdot\sigma(-x)\end{aligned}$$



$$\frac{d\sigma(x)}{dx} = \sigma(x)\cdot(1-\sigma(x))$$

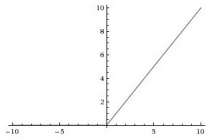
gradient always smaller than 1 → vanishing gradient

26

Rectified Linear Units (ReLUs)

$$A(x) = \max(0,x)$$

- gradient is 1 of >0, otherwise 0
- sparse activations



27

Activation functions: what we didn't cover

Leaky-relu and variations

Comparison of non-linear activation functions for deep neural networks on MNIST classification task

Dabal Pedamonti
Department of Computer Science
University of Edinburgh

Empirical Evaluation of Rectified Activations in Convolution Network

Bing Xu ANTINUCLEON@GMAIL.COM
University of Alberta
Naiyan Wang WINSTY@GMAIL.COM
Hong Kong University of Science and Technology
Tianqi Chen TQCHEN@CS.WASHINGTON.EDU
University of Washington
Mu Li MULI@CS.CMU.EDU
Carnegie Mellon University

28

Normalization in learning

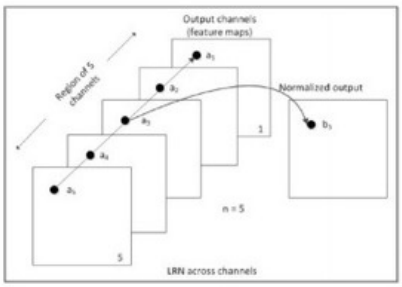
- "whitening". → 0 mean, 1 std
  - (i) the features are less correlated with each other, and
  - (ii) the features all have the same variance.
- It is well established that networks converge faster if the inputs have been whitened (ie zero mean, unit variances) and are uncorrelated
- Avoid numerical errors

29

Response Normalization

$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2)^\beta$$

where
$b^i_{x,y}$ – regularized output for kernel i at position x, y
$a^i_{x,y}$ – source ouput of kernel i applied at position x, y
N – total number of kernels
n – size of the normalization neigbourhood
$\alpha, \beta, k, (n)$ – hyperparameters



the output of ReLu is unbounded, sometimes hurts learning

"This sort of response normalization implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs computed using different kernels"

30

Batch Normalization

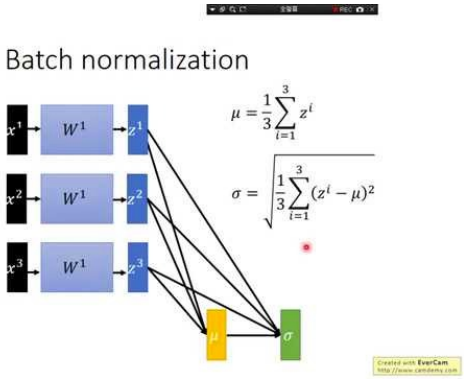
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

- Always normalize(whitening) the inputs into a good range

31



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

alpha, beta are one for each feature map (C_in total)
how many mean and variances? [H, W, C_in]
how about testing?

32

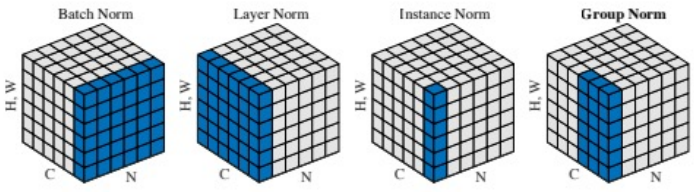
Benefits

- **Networks train faster** — Each training *iteration* will actually be slower because of the extra calculations during the forward pass and the additional hyperparameters to train during back propagation. However, it should converge much more quickly, so training should be faster overall.
- **Allows higher learning rates** — Gradient descent usually requires small learning rates for the network to converge. And as networks get deeper, their gradients get smaller during back propagation so they require even more iterations. Using batch normalization allows us to use much higher learning rates, which further increases the speed at which networks train.
- **Makes weights easier to initialize** — Weight initialization can be difficult, and it's even more difficult when creating deeper networks. Batch normalization seems to allow us to be much less careful about choosing our initial starting weights.
- **Makes more activation functions viable** — Some activation functions do not work well in some situations. Sigmoids lose their gradient pretty quickly, which means they can't be used in deep networks. And ReLUs often die out during training, where they stop learning completely, so we need to be careful about the range of values fed into them. Because batch normalization regulates the values going into each activation function, non-linearlities that don't seem to work well in deep networks actually become viable again.
- **Simplifies the creation of deeper networks** — Because of the first 4 items listed above, it is easier to build and faster to train deeper neural networks when using batch normalization. And it's been shown that deeper networks generally produce better results, so that's great.
- **Provides a bit of regularization** — Batch normalization adds a little noise to your network. In some cases, such as in Inception modules, batch normalization has been shown to work as well as dropout. But in general, consider batch normalization as a bit of extra regularization, possibly allowing you to reduce some of the dropout you might add to a network.
- **May give better results overall** — Some tests seem to show batch normalization actually improves the training results. However, it's really an optimization to help train faster, so you shouldn't think of it as a way to make your network better. But since it lets you train networks faster, that means you can iterate over more designs more quickly. It also lets you build deeper networks, which are usually better. So when you factor in everything, you're probably going to end up with better results if you build your networks with batch normalization.

33

Normalization: what we didn't cover

- layer normalization, instance normalization, group normalization
- alpha, beta stays the same
- mean, variance computed differently



Group Normalization

Yuxin Wu Kaiming He
Facebook AI Research (FAIR)
{yuxinwu,kaiminghe}@fb.com

34

Conv + ReLu + BN or Conv + BN + ReLu?

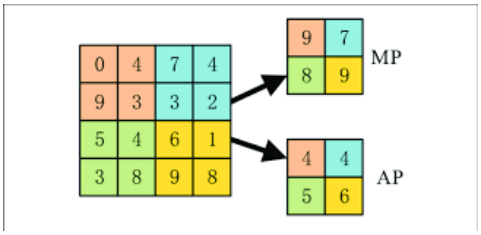
- The original BatchNorm paper prescribes using BN before ReLU
- recent code written by Christian [Szegedy, from the BN paper] applies relu before BN. The weight matrix W then looks at mean-centered data

BN before ReLu

Name	Accuracy	LogLoss	Comments
Before	0.474	2.35	As in paper
Before + scale&bias layer	0.478	2.33	As in paper
After	**0.499**	**2.21**	
After + scale&bias layer	0.493	2.24	

35

Pooling



Max pooling

Avg pooling

Parameters you need to be careful:
1. stride
2. kernel size

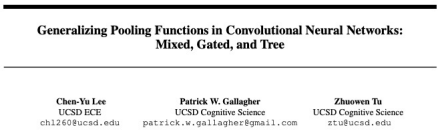
max pooling is more frequently used in early layers for
1. sparsity
2. "Or" operation, better for signal selection and higher signal to noise ratio

36

Pooling: what we didn't cover

Max-pooling and average pooling has their own drawbacks

how about learning more intelligent pooling?



Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree

Chen-Yu Lee, UCSD ECE, ch1260@ucsd.edu
Patrick W. Gallagher, UCSD Cognitive Science, patrick.w.gallagher@gmail.com
Zhuowen Tu, UCSD Cognitive Science, ztu@ucsd.edu

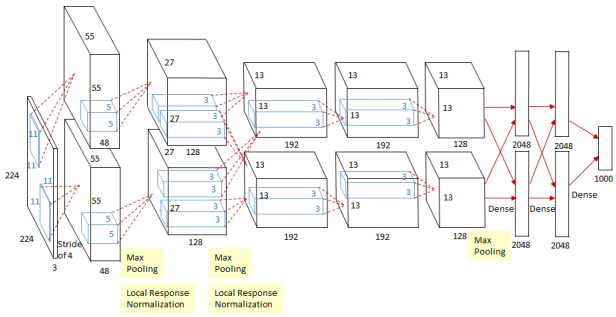
Region Ranking SVM for Image Classification

Zijun Wei Minh Hoai
Stony Brook University, Stony Brook, NY 11794, USA
{zijwei,minhhoai}@cs.stonybrook.edu

37

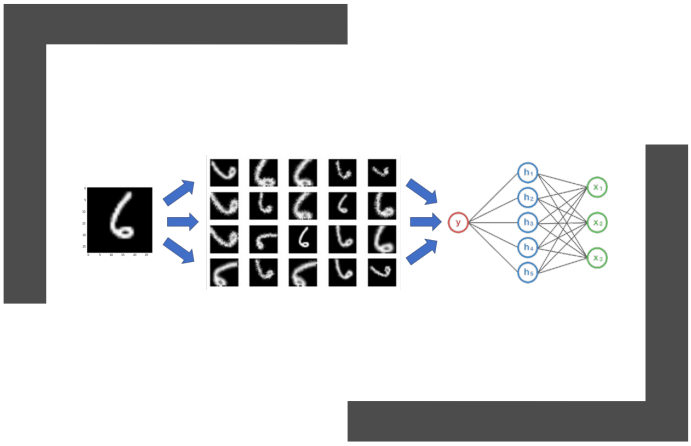
Why is AlexNet working?

- ~~Convolutional Layers (LeCun, 1998)~~
- ~~Rectified Linear Units (Activation Function)~~
- ~~Local Response Normalization~~
- ~~Pooling~~
- Data Augmentation
- Dropout (overfitting prevention)
- GPU Training (* most important)



38

Data Augmentation



1. Flip horizontally
2. Random crops/scales
3. Random mix/combinations of : -
translation
- rotation
- stretching
  - shearing,
  - lens distortions, … (go crazy)
4. Color jittering
(maybe even contrast jittering, etc.)
- Simple: Change contrast small amounts, jitter the color distributions, etc.

39

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Flip

label: cat or not cat?

Original image

Horizontal flip and vertical flip

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

40

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Crop

label: cat or not cat?

Original image

RoI crop

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

41

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Rotation

label: cat or not cat?

Original image

Rotation with different angles

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

42

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Adding noise

label: cat or not cat?

Original image

New images with different Gaussian noises

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

43

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Grayscale

label: cat or not cat?

Left: RGB image. Right: grayscale image

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

44

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Color Jittering

label: cat or not cat?

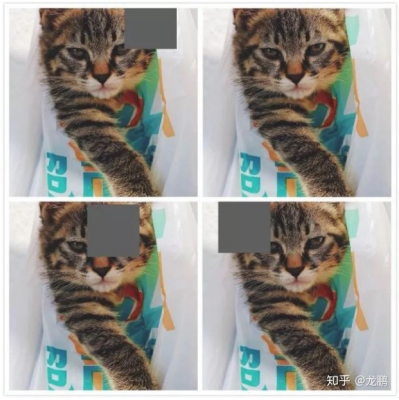
Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

45

Data augmentation

- Enlarge your dataset by synthesizing more data (label remains the same)
  - Random erasing (Zhun Zhong et al, AAAI'2020)



label: cat or not cat?

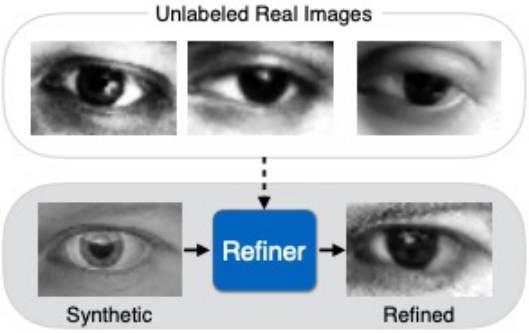
Random erasing

Slides by Heng Fan

Images from: https://zhuanlan.zhihu.com/p/38345420

46

More about data augmentation



The Effectiveness of Data Augmentation in Image Classification using Deep Learning

Jason Wang
Stanford University
450 Serra Mall
zwang01@stanford.edu

Luis Perez
Google
1600 Amphitheatre Parkway
nautilik@google.com

Learning from Simulated and Unsupervised Images through Adversarial Training

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, Russ Webb
Apple Inc
{a_shrivastava,tpf,otuzel,jsusskind,wenda_wang,rwebb}@apple.com

47

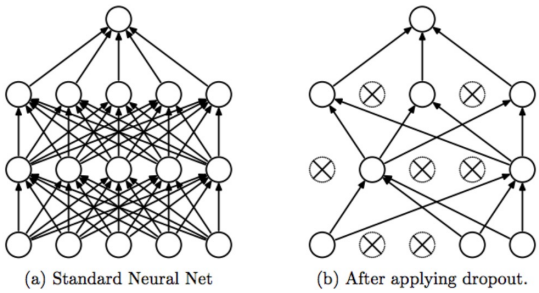
Dropout

- Dropout doesn't change input size, just randomly change some values to 0
- Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

benefits:

1. preventing complex co-adaptations on training data (co-adaptation means that some neurons are highly dependent on others)
2. creates ensembles

- Where is drop-out usually used in a network?
  fully-connected layers
- Other methods to prevent over-fitting?
  BN



(a) Standard Neural Net (b) After applying dropout.

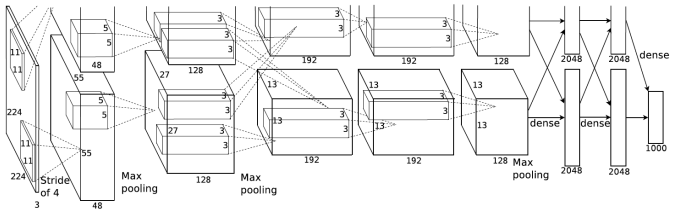
48

GPU training

Alex Krizhevsky is a hard-core engineer

with only 3G GPU memory, he had to distribute into two halves

- The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU
- The kernels of the third convolutional layer are connected to all kernel maps in the second layer
- The neurons in the fully connected layers are connected to all neurons in the previous layer



Parameters:
1st Conv: 11 * 11 * 3 * 96. (why 3?)
2nd Conv: 5 * 5* 48 (should be 96 if 1 GPU) * 256
3rd Conv: 3 * 3 * 256 (all connected to 2nd)* 384
4th Conv: 3 * 3 * 192 (should be 384 if 1 GPU) * 384
5th Conv: 3 * 3 * 192 (should be 384 if 1 GPU) * 256

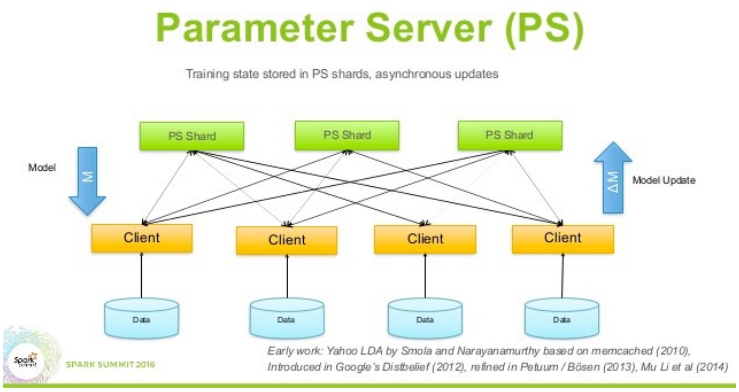
49

Now we have larger GPUs

- we don't have to be so careful about putting 1 network into multiple GPUs
- But we are facing the problem of training billions of images
- How to do distributed training?

50

Parameter server



- Asynchronous updates
  - don't need to wait for sync
  - usually work for fewer number of servers
  - converge slower
- Synchronous updates
  - need to wait for the slowest updates
  - scales comparably better

How to reduce the overhead of synchronous updates?

- using more machines. e.g.: deploy 15 machines, wait for the fastest 10

51

More about training on multiple GPUS

- communication
- hyper-parameter finding?
- Batch-normalization?

Scaling Distributed Machine Learning with the Parameter Server

Mu Li*†, David G. Andersen*, Jun Woo Park*, Alexander J. Smola*†, Amr Ahmed‡, Vanja Josifovski†, James Long†, Eugene J. Shekita†, Bor-Yiing Su†

*Carnegie Mellon University ‡Baidu †Google

{muli, dga, junwoop}@cs.cmu.edu, alex@smola.org, {amra, vanjaj, jamlong, shekita, boryiingsu}@google.com

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

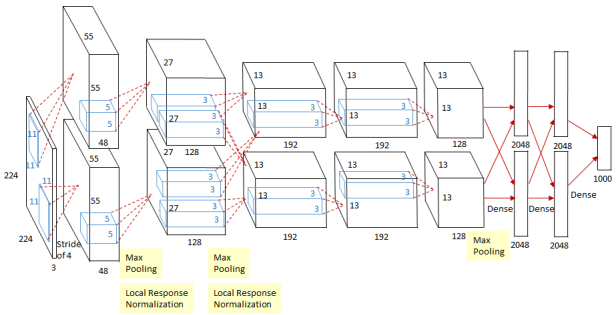
Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

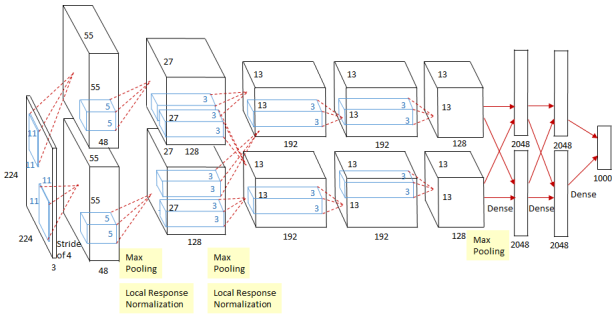
52

Why is AlexNet working?

- ~~Convolutional Layers (LeCun, 1998)~~
- ~~Rectified Linear Units (Activation Function)~~
- ~~Local Response Normalization~~
- ~~Pooling~~
- ~~Data Augmentation~~
- ~~Dropout (overfitting prevention)~~
- ~~GPU Training (* most important)~~



53



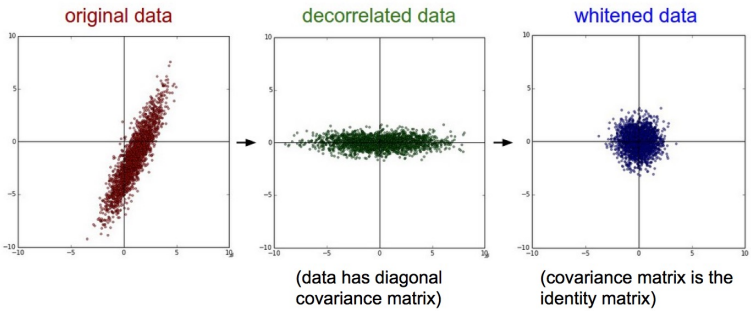
How to train a network, really?

54

How to train Neural Networks

Step 1: Preprocess the data

In practice, you may also see **PCA** and **Whitening** of the data



Slide credit:Fei-Fei Li

55

How to train Neural Networks

- **Step 2: Choose the architecture:**
  - How many Layers? How many nodes?
  - Conv or fully connected
  - loss function. why cross-entropy is better than L2 for classification?
- **Step 3: Initialize well**
  - set weights to small random numbers [Why? Why not just initialize to all zeros?]. (some initialization techniques Kaiming, Xaiver…)
  - set biases to zero
  - Double check that the loss is reasonable (loss upper bound, e.g, estimate the cross-entropy loss over N random predictions + some reg.)

Slide credit:Fei-Fei Li

56

How to train Neural Networks

- **Step 4: Let's try to train:**
  - start with small regularization and find learning rate that makes the loss go down.
  - **loss not going down:**
  - learning rate too low
  - **loss exploding:**
  - learning rate too high
- **Step 5: Cross-validation strategy:**
  - **coarse -> fine** cross-validation in stages
  - **First stage**: only a few epochs to get rough idea of what params work **Second stage**: longer running time, finer search
    ... (repeat as necessary)
  - Tip for detecting explosions in the solver:
    If the cost is ever > 3 * original cost, break out early

Slide credit:Fei-Fei Li

57

How to train Neural Networks

Normally you can't afford a huge computational budget for expensive cross-validations.
Need to rely more on intuitions and visualizations...

Visualizations to play with:

- **loss** function
- validation and training **accuracy**
- min,max,std for **values and updates**, (and monitor their ratio)
- **first-layer visualization** of weights (if working with images)

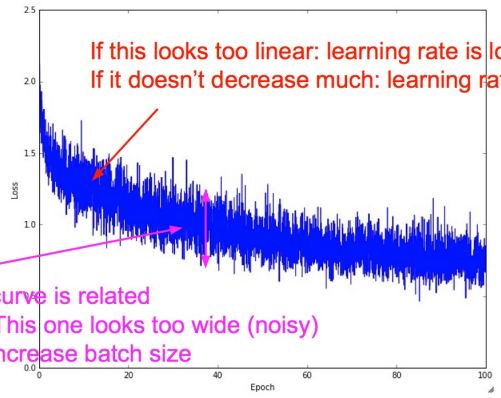
Seemingly unrelated: **Model Ensembles**
- One way to *always* improve final accuracy:

take several trained models and average their predictions

Slide credit:Fei-Fei Li

58

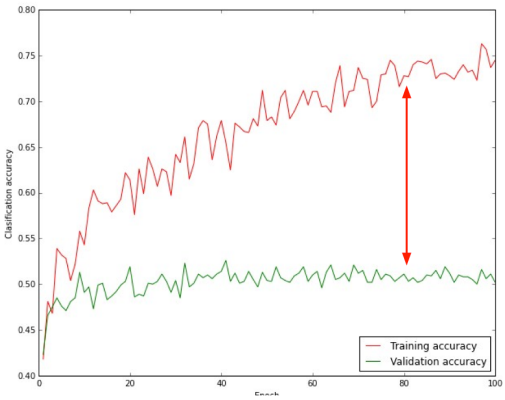
Monitor and visualize the loss curve



Slide credit:Fei-Fei Li

59

Monitor and visualize the accuracy:



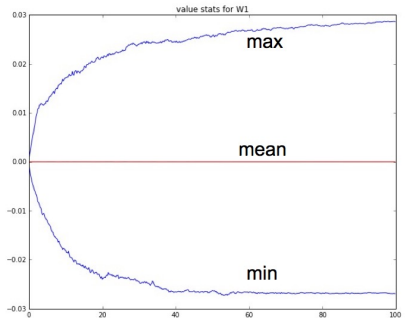
big gap = overfitting
=> increase regularization strength

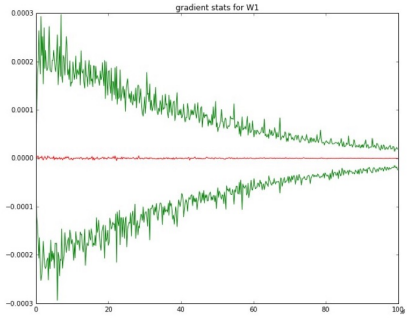
no gap
=> increase model capacity

Slide credit:Fei-Fei Li

60

Track the ratio of weight updates / weight magnitudes:





ratio between the values and updates: ~ 0.0002 / 0.02 = 0.01 (about okay)
want this to be somewhere around 0.01 - 0.001 or so

Slide credit:Fei-Fei Li

61

Dealing with data imbalance

- Data imbalance widely exists in real world.

In training data, samples of A are much more than samples of B

  - Some common solutions
    - Use weight in loss function, i.e., increase the weight for rare class
    - Under-sampling: Keep all samples in rare class, and randomly select a set of samples from abundant class
    - Over-sampling: Keep all samples in abundant class, and generate new samples in rare class (e.g., data augmentation)

62

62

Optimizer selection

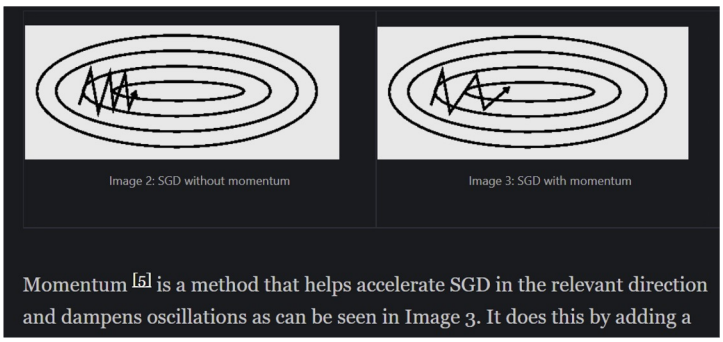
- Common optimizers
  - SGD: Stochastic gradient descent
  - AdaGrad: Adaptive gradient algorithm
  - RMSprop: Improved version of AdaGrad
  - ADAM: Adaptive moment estimation

My PhD student experience is to use ADAM for quick experiments, as it is less sensitive to learning rate.

A more detailed overview of gradient descent optimization algorithms is here: https://ruder.io/optimizing-gradient-descent/

63

63



64

Multi-scale straegy

- Let CNN learn features in different resolution

Image with scale=1

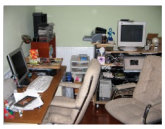
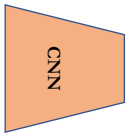
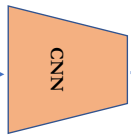
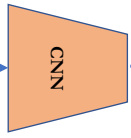


Image with scale=2/3

Image with scale=1/2







prediction

prediction

prediction

combination

1-2% performance gain

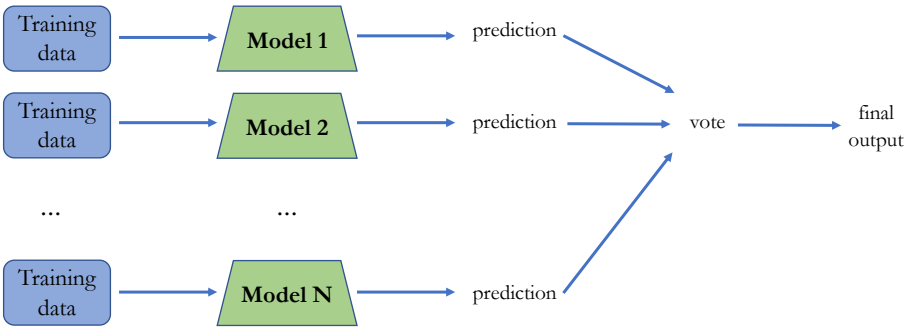
65

65

Ensemble model

- Combine different models for robustness

Widely adopted in competition



66

66

Evolution of AlexNet

- VGG
- GoogLeNet
- ResNet

67

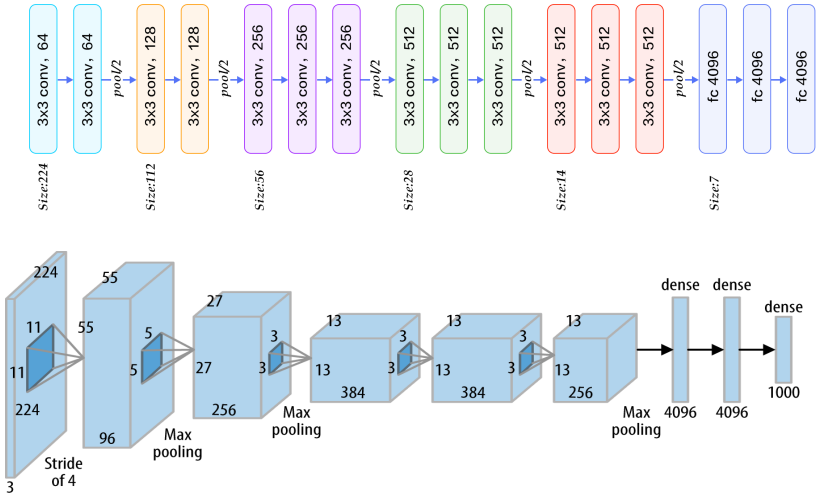
VGG

VERY DEEP CONVOLUTIONAL NETWORKS
FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman***
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

68

VGG

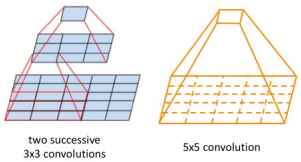


69

Design principle

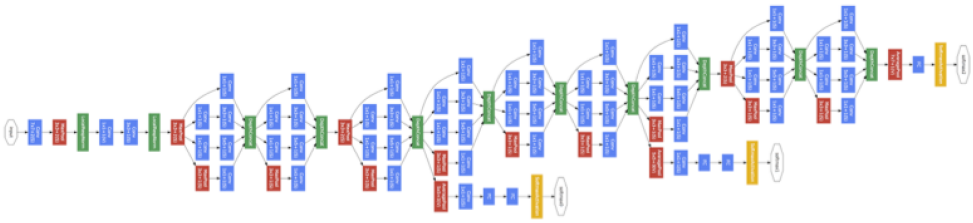
- Reduce the size of convolution
- 5 x 5 convolution == 2 stacks of 3 by 3 convolution
  - but saves a lot of parameters (5 * 5 = 25 vs. 2*3*3 = 18)
  - less parameters mean more constraints, easier to learn

Pros:
VGG is elegantly designed with valid reasons
VGG is modelized since 3 by 3 convolution is the only CNN module
Cons:
VGG is still one of the most redundant networks with heavy computation (many research on compression shows that VGG can be pruned with 90% of parameters without infecting too much performance)



70

GoogLeNet



Going Deeper with Convolutions

Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3], Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.
[1]{szegedy, jiayq, sermanet, dragomir, dumitru, vanhoucke}@google.com
[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

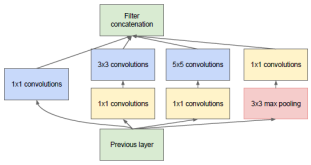
Convolution
Pooling
Softmax
Other

71

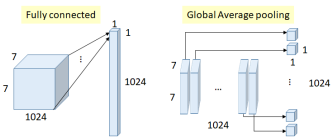
GoogLeNet

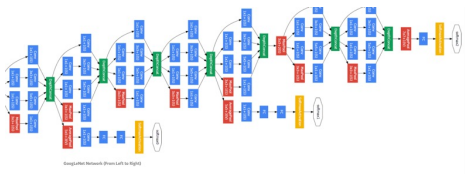
Key innovation

1. light-weight (using 1 x 1 convolutions to reduce channels)
2. global average pooling
3. intermediate layer training



(b) Inception module with dimensionality reduction





Why intermediate layer training?

72

GoogLeNet

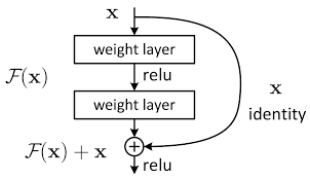
- Better performance
- Light weight

- The design intuition is not clear
- some papers reported less general features

73

ResNet

- BN makes deeper models not a problem
- optimization is still a problem because it's hard to approximate identity
  - y = f(x). vs y = h(x) + x. f(x) and h(x) were initialized with small random numbers, hard to learn identity mapping, but easier to learn residual
  - also information by-pass

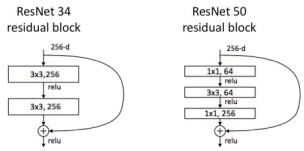


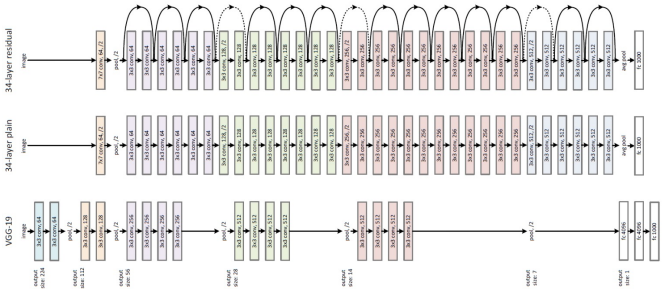
74

ResNet

- Based on identical blocks
- Can be well distributed

However
sometimes redundant





75

What we will not cover

- ResNeXt
- DenseNet

76

More models

- https://github.com/Cadene/pretrained-models.pytorch
- https://github.com/tensorflow/models/tree/master/research/slim

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50*	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101*	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152*	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8
MobileNet_v1_1.0_224	Code	mobilenet_v1_1.0_224.tgz	70.9	89.9
MobileNet_v1_0.50_160	Code	mobilenet_v1_0.50_160.tgz	59.1	81.9
MobileNet_v1_0.25_128	Code	mobilenet_v1_0.25_128.tgz	41.5	66.3
MobileNet_v2_1.4_224**	Code	mobilenet_v2_1.4_224.tgz	74.9	92.5
MobileNet_v2_1.0_224**	Code	mobilenet_v2_1.0_224.tgz	71.9	91.0
NASNet-A_Mobile_224#	Code	nasnet-a_mobile_04_10_2017.tar.gz	74.0	91.6
NASNet-A_Large_331#	Code	nasnet-a_large_04_10_2017.tar.gz	82.7	96.2
PNASNet-5_Large_331	Code	pnasnet-5_large_2017_12_13.tar.gz	82.9	96.2
PNASNet-5_Mobile_224	Code	pnasnet-5_mobile_2017_12_13.tar.gz	74.2	91.9

Model	Version	Acc@1	Acc@5
PNASNet-5-Large	Tensorflow	82.858	96.182
PNASNet-5-Large	Our porting	82.736	95.992
NASNet-A-Large	Tensorflow	82.693	96.163
NASNet-A-Large	Our porting	82.566	96.086
SENet154	Caffe	81.32	95.53
SENet154	Our porting	81.304	95.498
PolyNet	Caffe	81.29	95.75
PolyNet	Our porting	81.002	95.624
InceptionResNetV2	Tensorflow	80.4	95.3
InceptionV4	Tensorflow	80.2	95.3
SE-ResNeXt101_32x4d	Our porting	80.236	95.028
SE-ResNeXt101_32x4d	Caffe	80.19	95.04
InceptionResNetV2	Our porting	80.170	95.234
InceptionV4	Our porting	80.062	94.926
DualPathNet107_5k	Our porting	79.746	94.684
ResNeXt101_64x4d	Torch7	79.6	94.7
DualPathNet131	Our porting	79.432	94.574
DualPathNet92_5k	Our porting	79.400	94.620
DualPathNet98	Our porting	79.224	94.488
SE-ResNeXt50_32x4d	Our porting	79.076	94.434
SE-ResNeXt50_32x4d	Caffe	79.03	94.46
Xception	Keras	79.000	94.500
ResNeXt101_64x4d	Our porting	78.956	94.252
Xception	Our porting	78.888	94.292

77

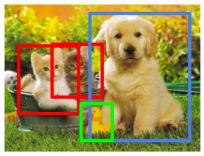
Detection

- What's the different between detection and classification?

Classification

CAT

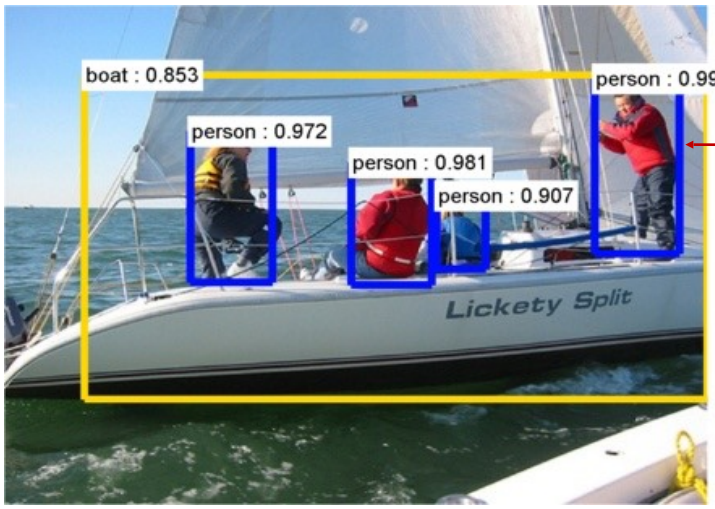
Object Detection



CAT, DOG, DUCK

78

Object detection with bounding boxes



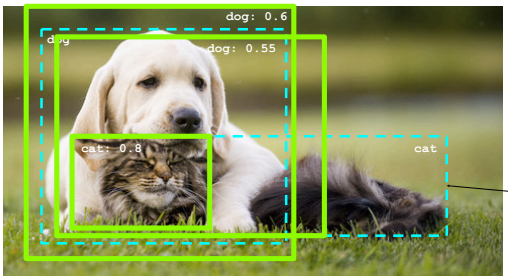
"Object detection"

Source: R. Girshic

79

Evaluating an object detector

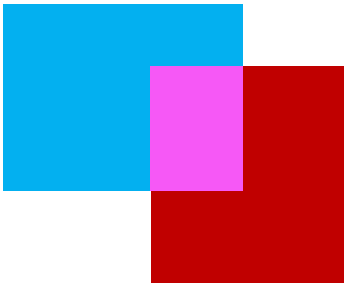
- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
  - **Intersection over union** (IoU): Area(GT ∩ Det) / Area(GT ∪ Det) > 0.5



Source: S. Lazebnik

80

Evaluating an object detector



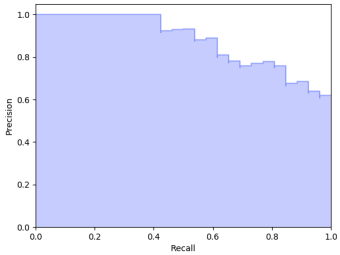
$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Intersection over union (also known as Jaccard similarity)

Source: B. Hariharan

Evaluating an object detector

- For each class, plot Recall-Precision curve and compute Average Precision (area under the curve)
- Take mean of AP over classes to get mAP

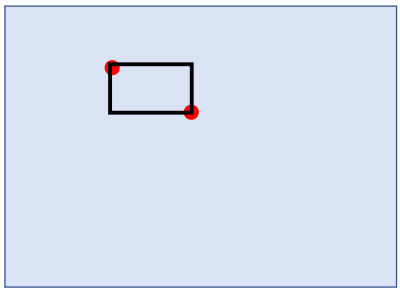


Precision:
true positive detections / total detections
Recall:
true positive detections / total positive test instances

Source: S. Lazebnik

Detection as classification

- Run through every possible box and classify
  - Well-localized object of class k or not?
- How many boxes?
  - Every pair of pixels = 1 box
  - $\binom{N}{2}$ = O(N²)
  - For 300 x 500 image, N = 150K
  - 2.25 x 10^{10} boxes!
- Related challenge: almost all boxes are negative!



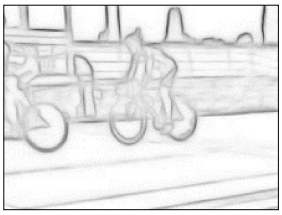
8

Source: B. Hariharan

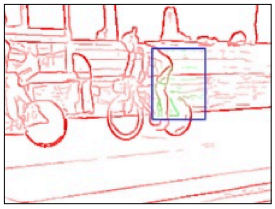
Selective search

Stage 1: generate candidate bounding boxes

Input image



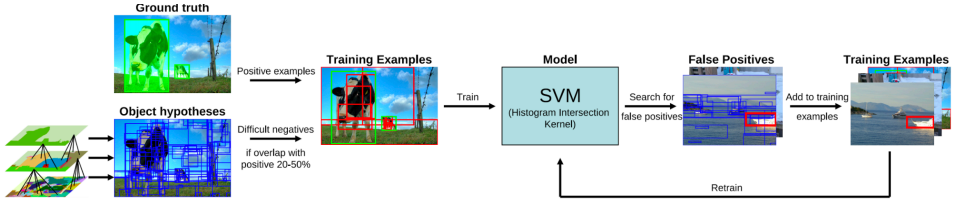
Edge detection



Bounding box proposal

[Zitnick and Dollar, "Edge Boxes…", 2014]

Stage 2: apply classifier only to each candidate bounding box



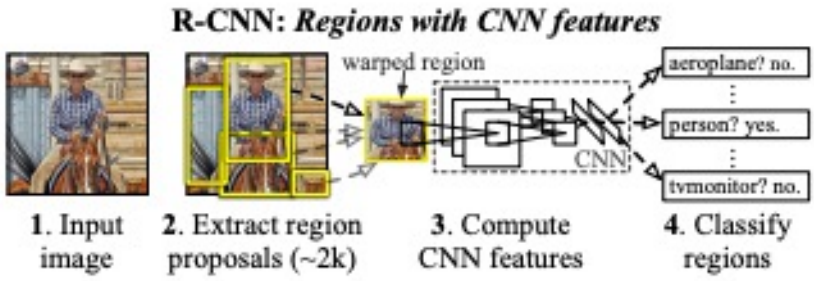
Source: Torralba, Freeman, Isola

9

[Uijlings et al., "Selective Search for Object Recognition", 2013]

Idea 1: Candidate location + classifier evaluation

- R-CNN



85

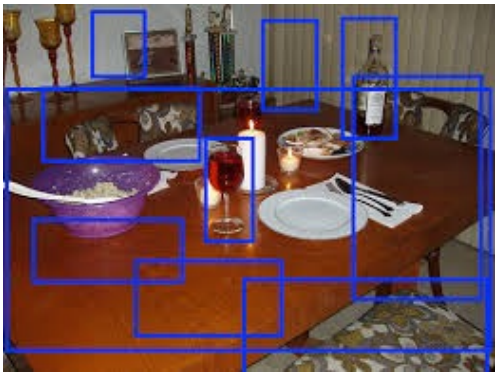
candidate location generation

- Method 1: Sliding window

86

candidate location generation

- Method 2: Objectness…



- ”Coarse to fine”
- What metrics to use to evaluate an object proposal algorithm

87

Some more computer vision design principles

- Coarse to fine
- Multi-scale
- “Invariance” (rotation invariant, translation invariant) to be robust and save parameters
- Locality

88

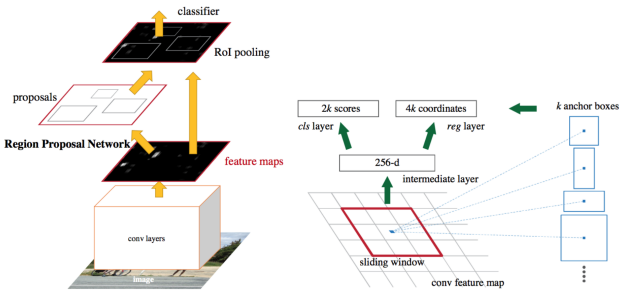
R-CNN is extremely slow

Because it need to evaluate each proposal separately

89

Faster RCNN

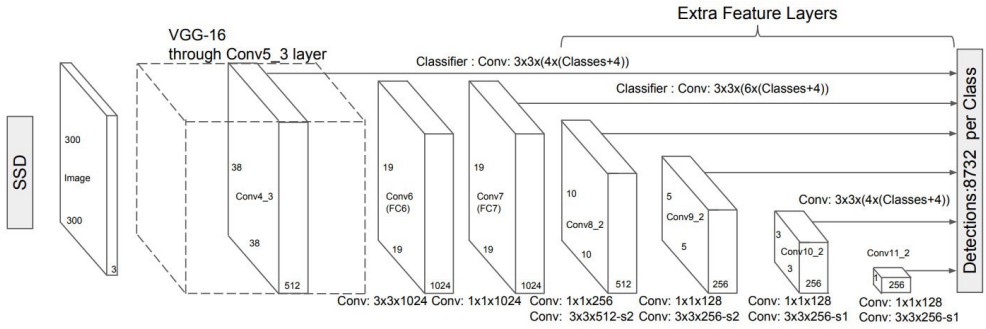
- One network for two tasks (proposal + detection)
- We will talk about details if we have time.



90

SSD

- Why bother with proposals if we have enough bounding boxes already



91

detection model-zoos

- https://github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md

RetinaNet Baselines

backbone	type	lr schd	im/ gpu	train mem (GB)	train time (s/iter)	train time total (hr)	inference time (s/im)	box AP	mask AP	kp AP	prop. AR	model id	download links
R-50-FPN	RetinaNet	1x	2	6.8	0.483	12.1	0.125	35.7	-	-	-	36768636	model | boxes
R-50-FPN	RetinaNet	2x	2	6.8	0.482	24.1	0.127	35.7	-	-	-	36768677	model | boxes
R-101-FPN	RetinaNet	1x	2	8.7	0.666	16.7	0.156	37.7	-	-	-	36768744	model | boxes
R-101-FPN	RetinaNet	2x	2	8.7	0.666	33.3	0.154	37.8	-	-	-	36768840	model | boxes
X-101-64x4d-FPN	RetinaNet	1x	2	12.6	1.613	40.3	0.341	39.8	-	-	-	36768875	model | boxes
X-101-64x4d-FPN	RetinaNet	2x	2	12.6	1.625	81.3	0.339	39.2	-	-	-	36768907	model | boxes
X-101-32x8d-FPN	RetinaNet	1x	2	12.7	1.343	33.6	0.277	39.5	-	-	-	36769563	model | boxes
X-101-32x8d-FPN	RetinaNet	2x	2	12.7	1.340	67.0	0.276	38.6	-	-	-	36769641	model | boxes

92