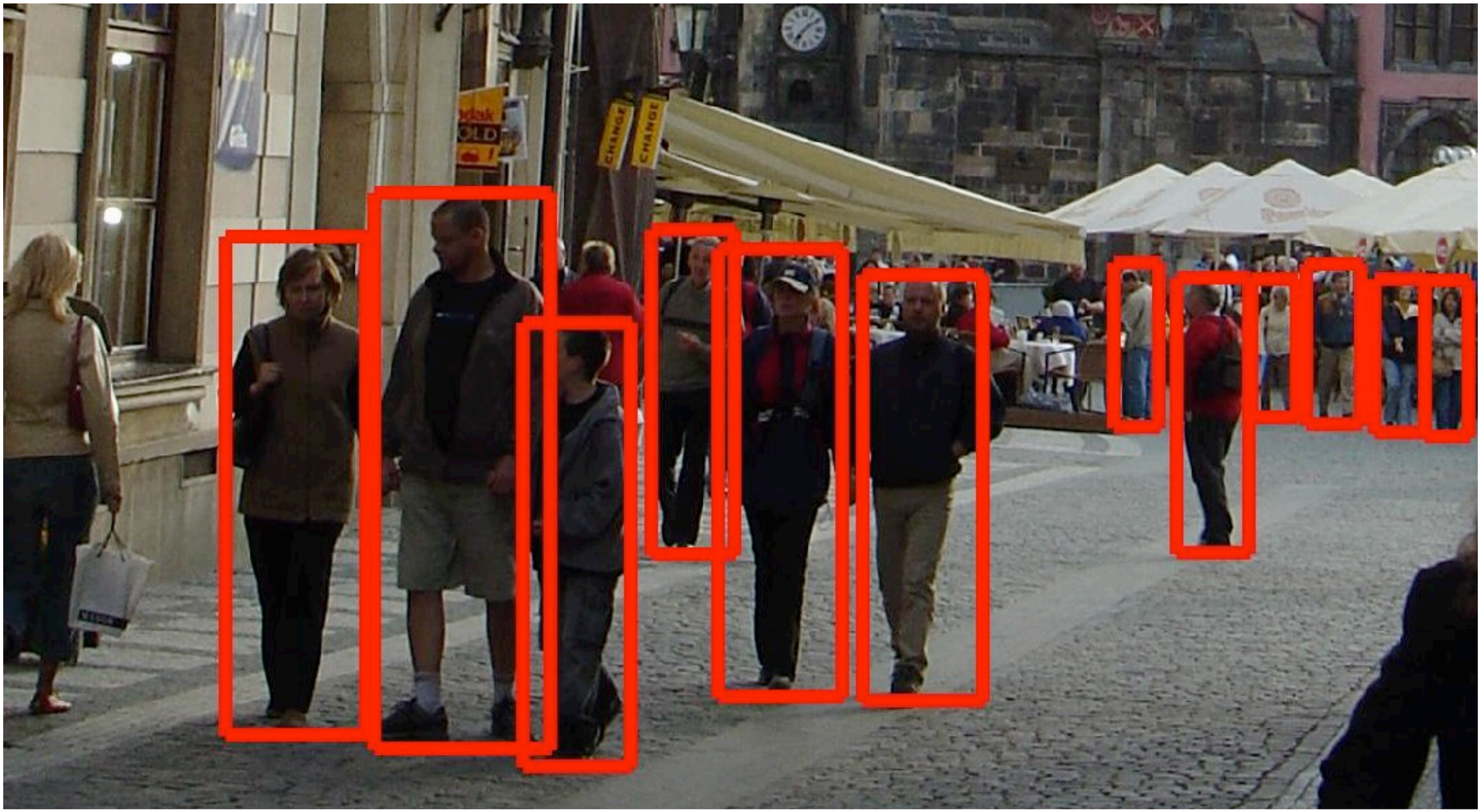
Deep learning

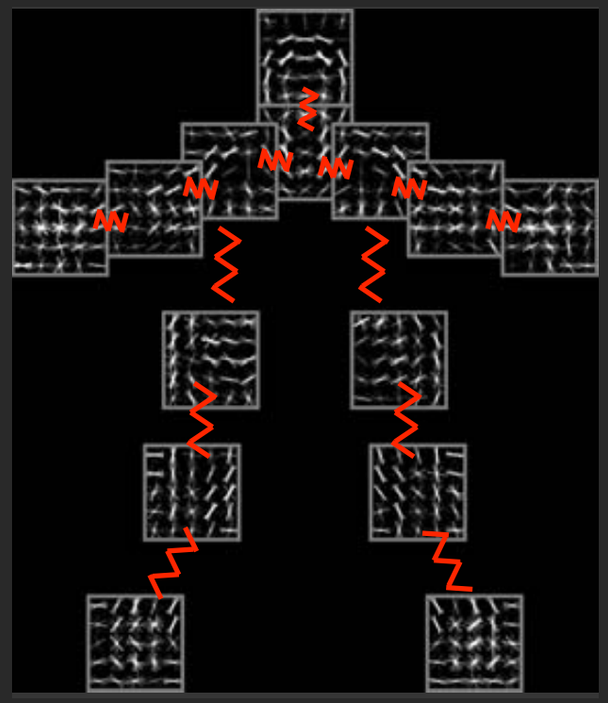
Outline

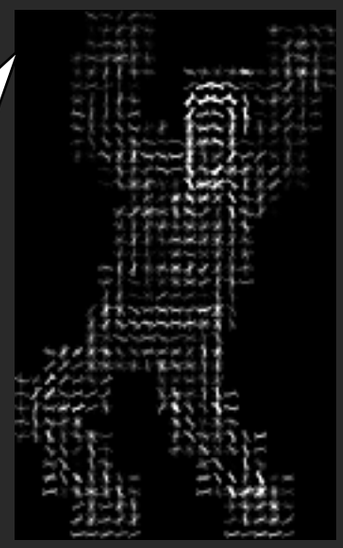
- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale DAGs, recurrence, LSTMs
- Why does it work so well?

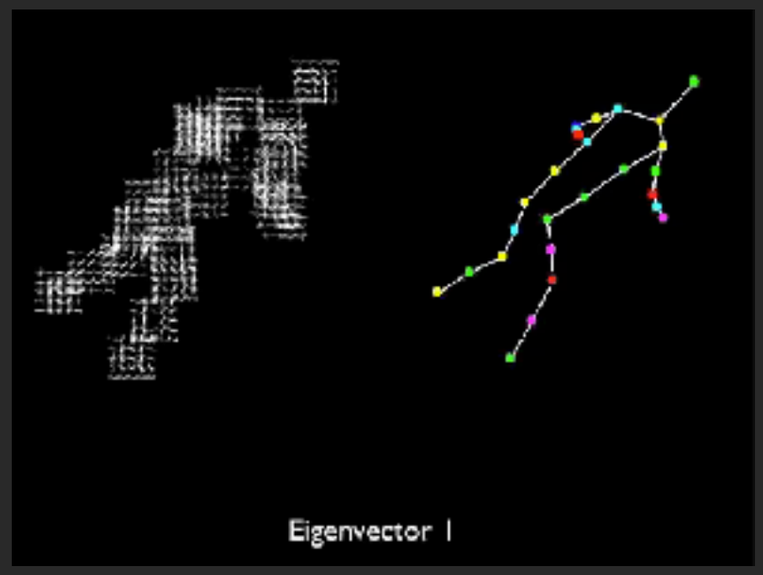
(we'll probably get through 1/2 of this today…)


CHANGE
CHANGE

Parts to the rescue!

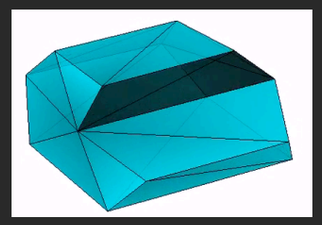




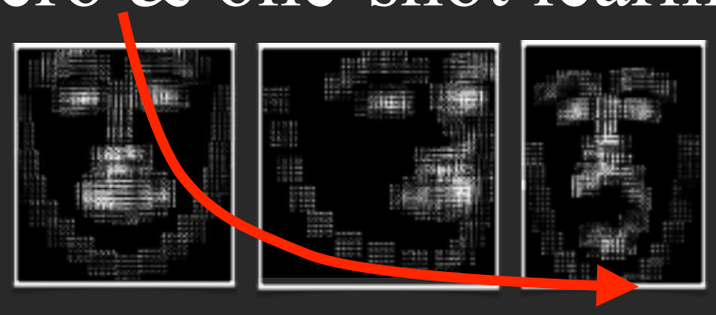


Why?

Recognition through reconstruction: latent-variable classification

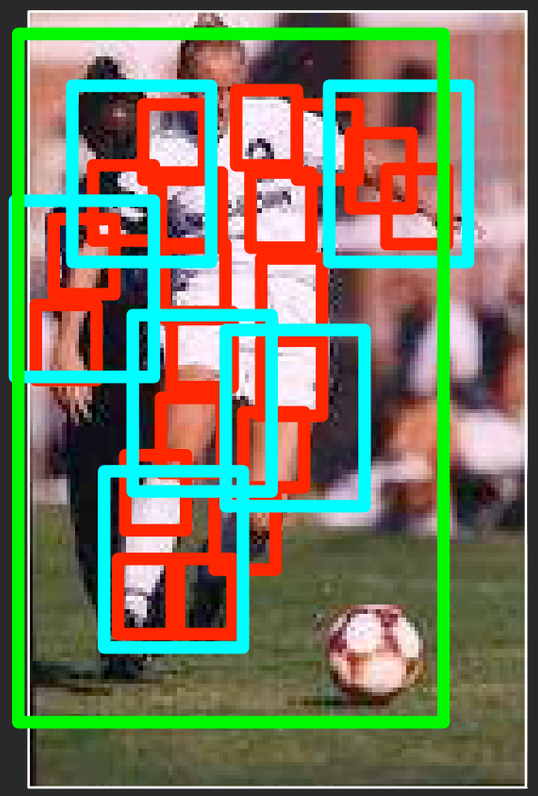


Sharing + synthesis: zero & one-shot learning for tails



Central challenge: part discovery

Latent hierarchical models



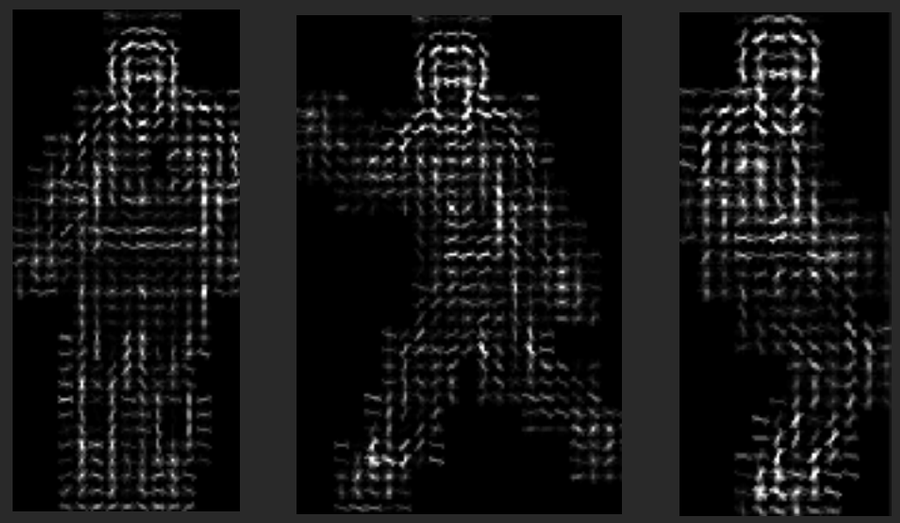
$$S(x,z) = \sum_{i \in V} w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

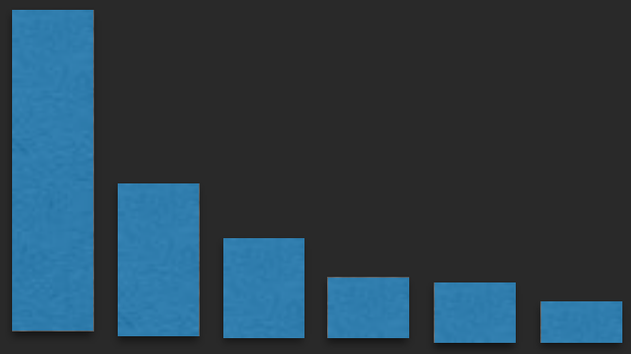
Can we write as a set of templates? $S(x,z) = w(z) \cdot \Phi(x)$

Shape models

$$S(x, z) = \sum_{i \in V} w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

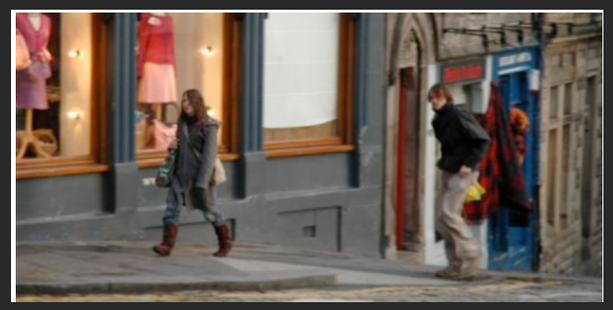
$$S(x, z) = w(z) \cdot \Phi(x) + b(z)$$

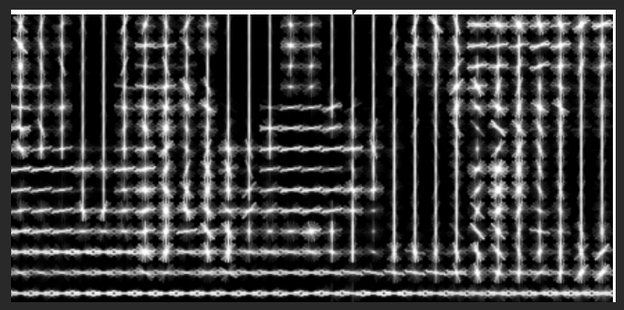


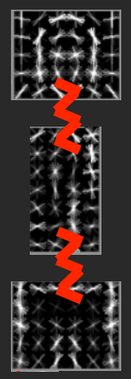


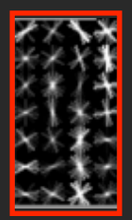
Inference: $\max_z S(x,z)$

implement with convolutions + max pooling (c.f. distance transform)









conv

conv

max pool

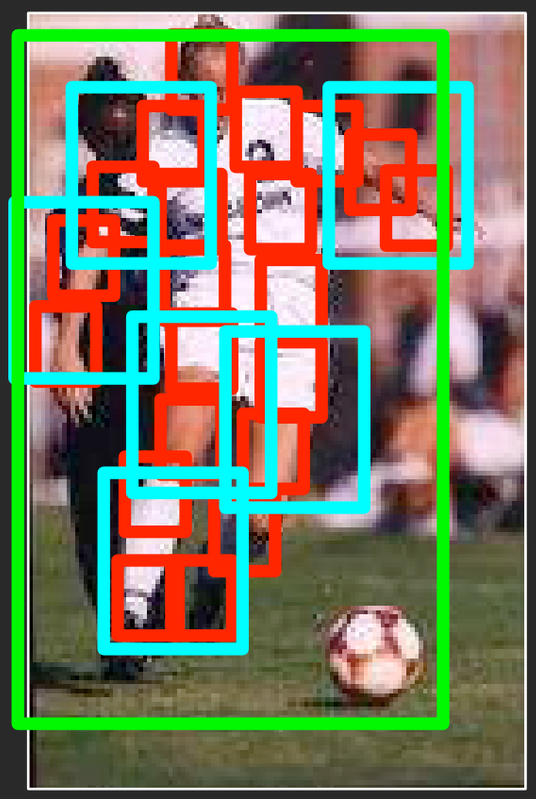
add

max pool

add

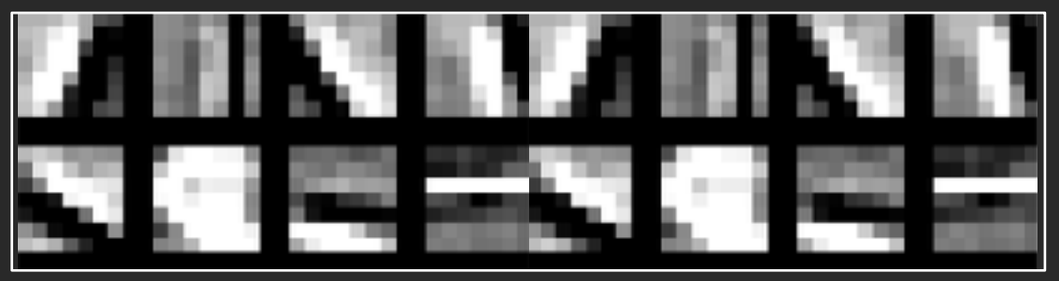
• • •

Can deep networks be viewed as hierarchical part models?



objects

parts



subparts

Often a common motivation (for a vision audience)

We'll look at detail in a bit…

Deep learning

Much of the field is in rapid motion

No standard textbooks (yet!)

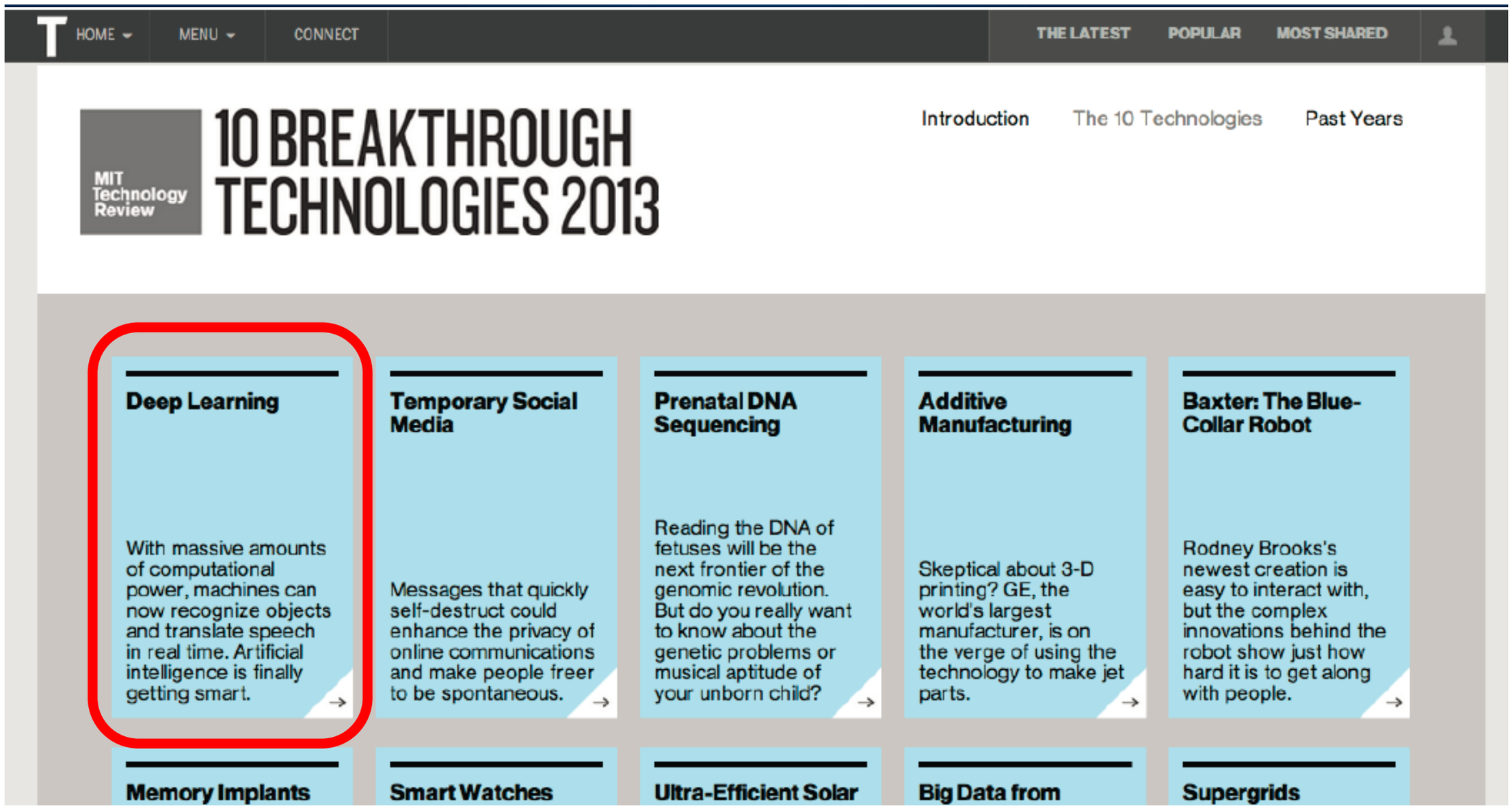
Some of my favorite references

1. https://sites.google.com/site/deeplearningsummerschool/
2. http://www.deeplearningbook.org/
3. http://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

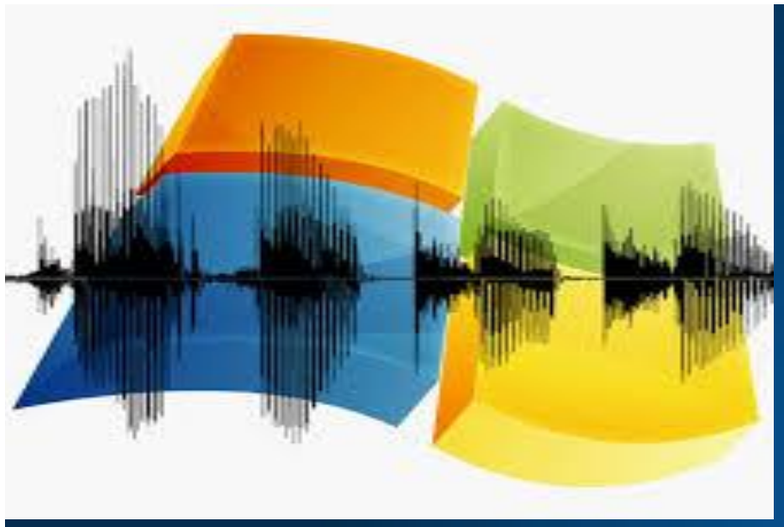
My favorite (Matlab) toolbox

http://www.vlfeat.org/matconvnet/

Motivation


HOME
MENU
CONNECT
THE LATEST
POPULAR
MOST SHARED
MIT Technology Review
10 BREAKTHROUGH TECHNOLOGIES 2013
Introduction
The 10 Technologies
Past Years
Deep Learning
With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.
Temporary Social Media
Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.
Prenatal DNA Sequencing
Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?
Additive Manufacturing
Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.
Baxter: The Blue-Collar Robot
Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.
Memory Implants
Smart Watches
Ultra-Efficient Solar
Big Data from
Supergrids

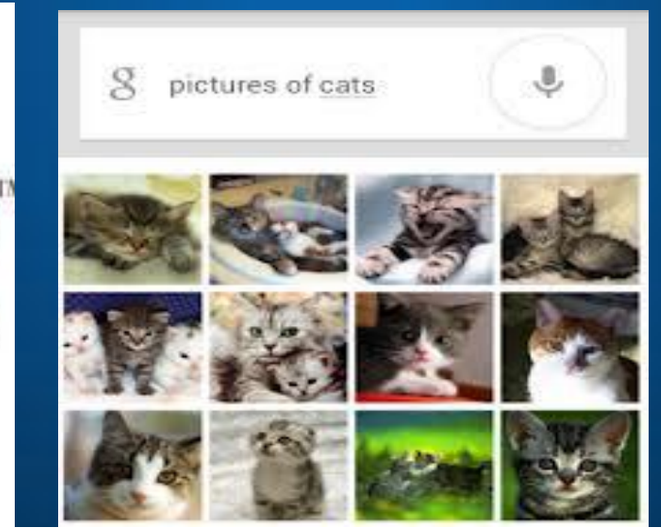
Products



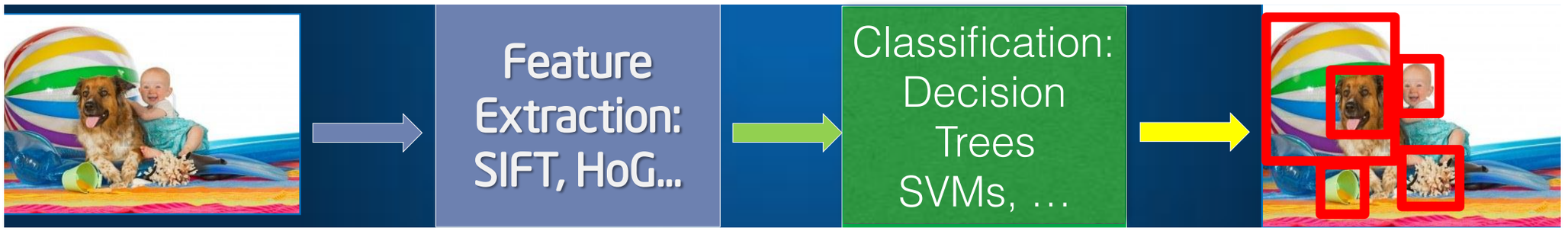

WEB
IMAGES
VIDEOS
MAPS
NEWS
MORE
bing
golden retriever
Size
Color
Type
Layout
People
Date
License
Funny
Golden Retriever
Miniature
Golden Retriever
Baby
Golden Retriever
White
Golden Retriever

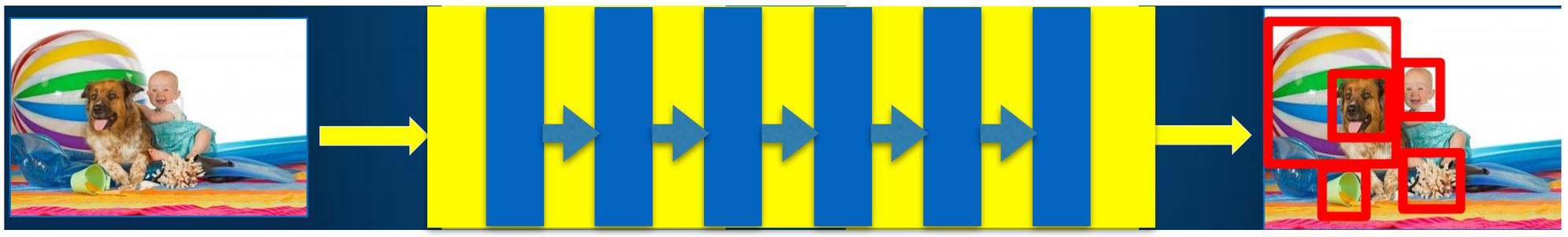

PhotoWonder
A magic photo beautifier, cool and easy to use!
Bai du 百度


Translate
Google


pictures of cats

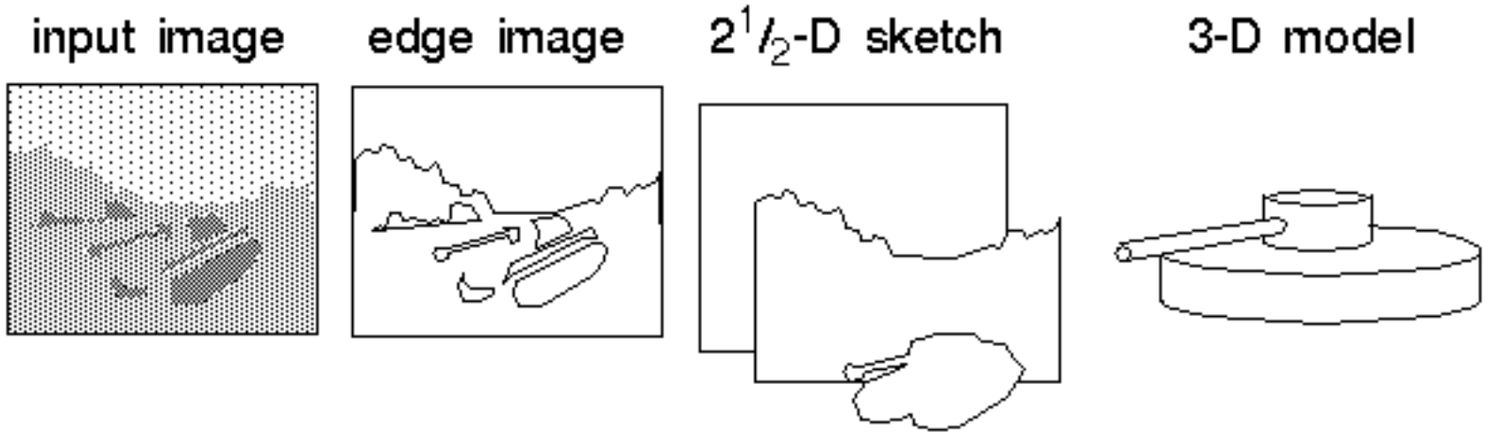

flickr

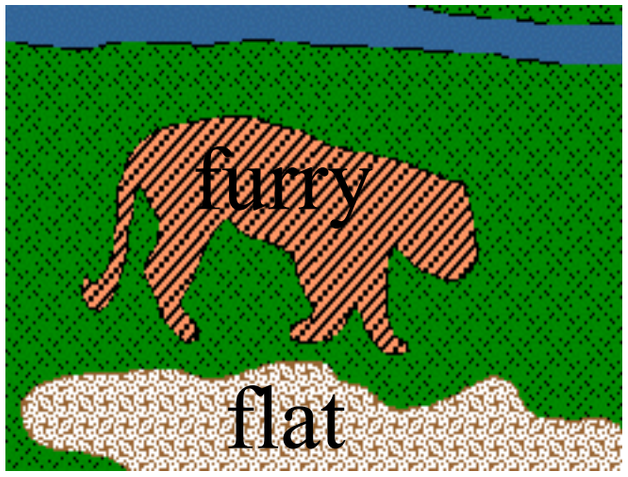


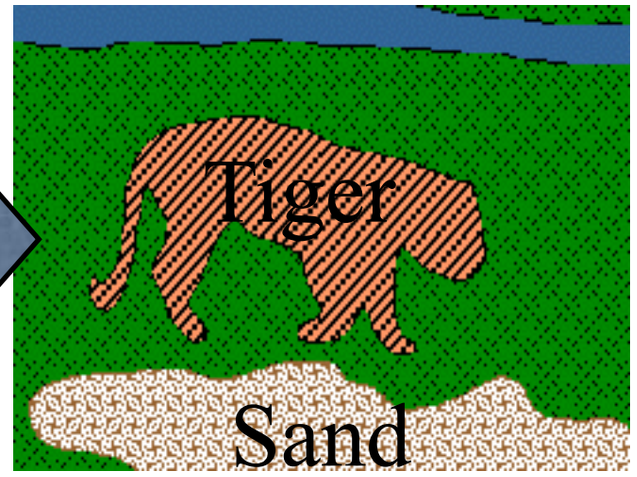


Deep network

Hierarchies in vision







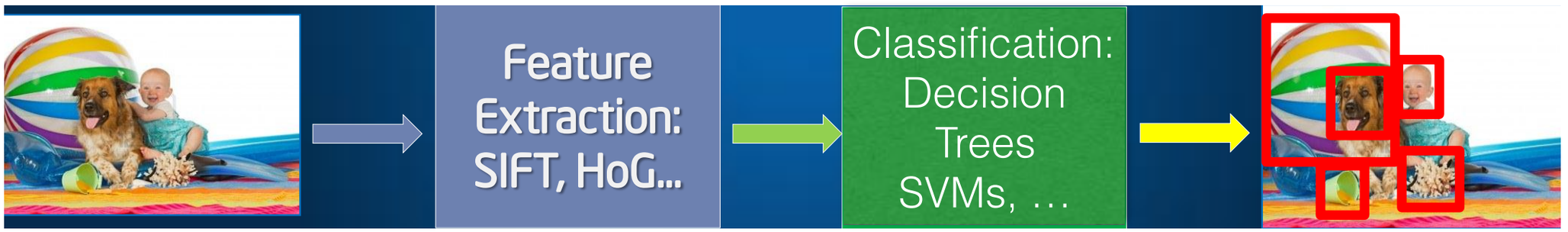
Low-level
Signal-processing

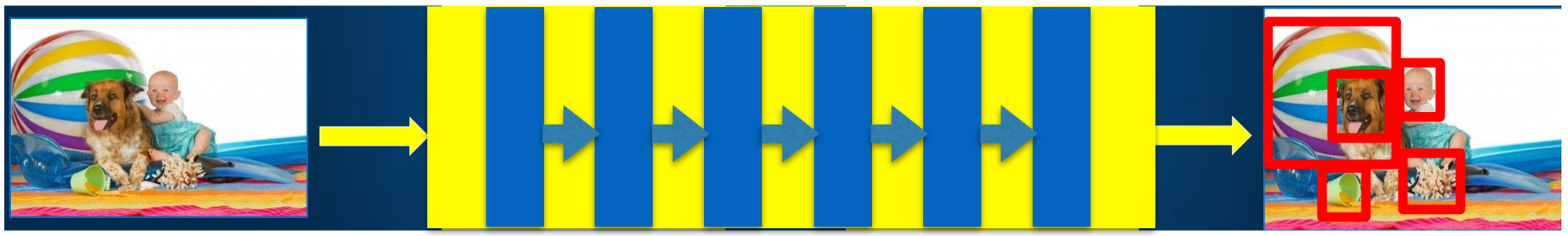
Mid-level
Grouping

High-level
Recognition

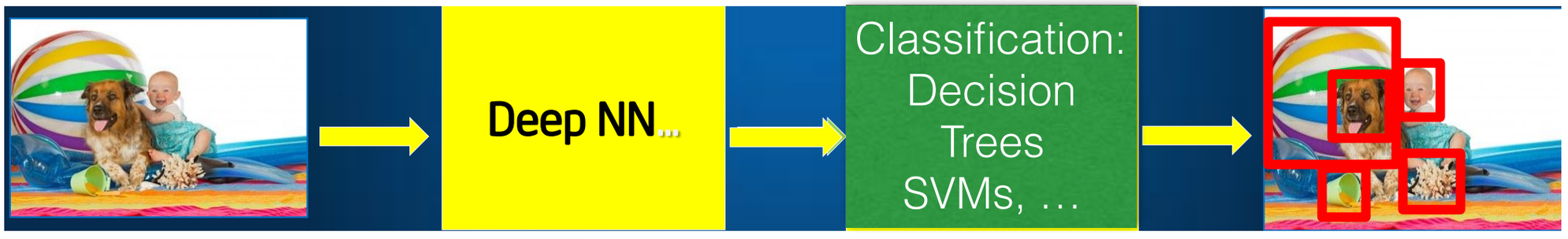
"Lesson" from deep learning: perhaps hiearchichies should be learned rather than hand-designed

Off-the-shelf baseline





End-to-end “fine-tuned”



“Off-the-shelf”

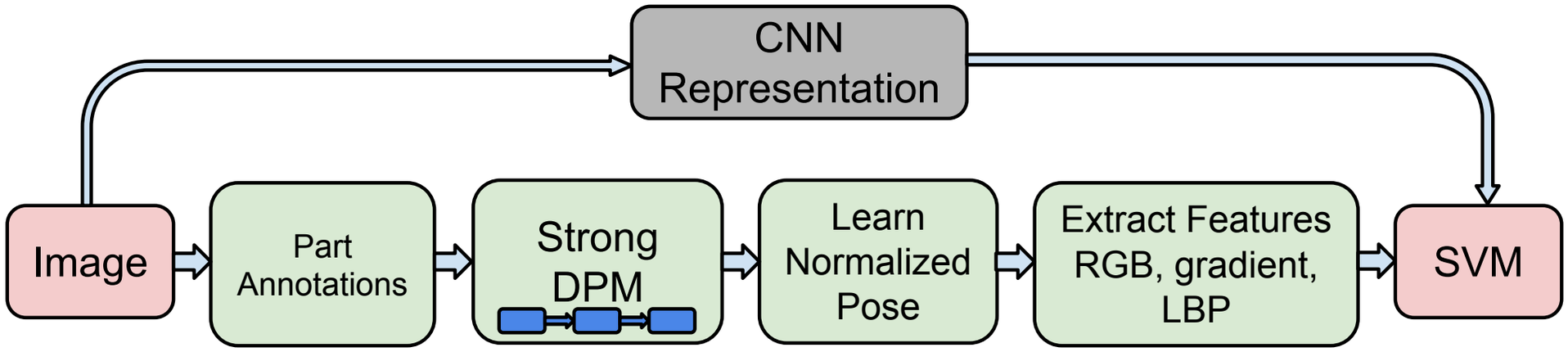
My personal thoughts: a universal feature extractor is within reach

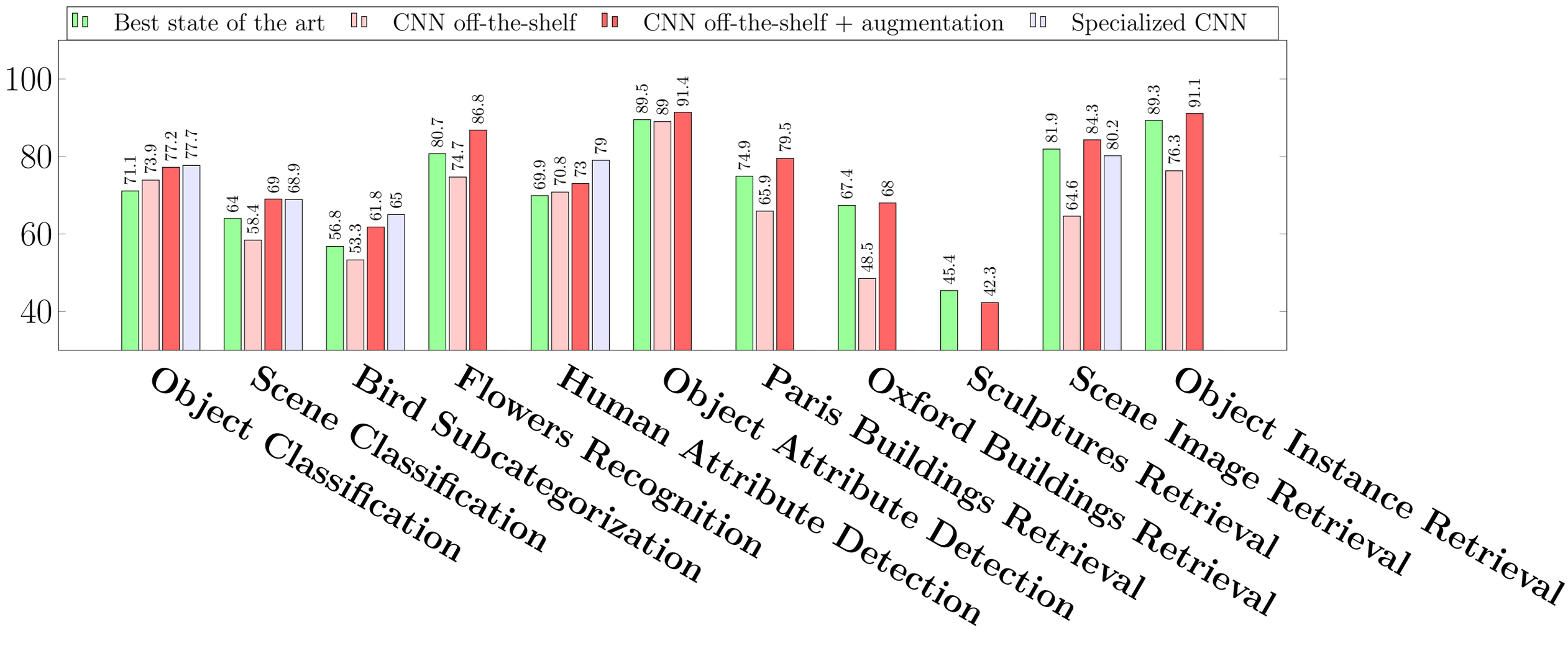
Implies that “version0” of any deep learning solution shouldn’t do any deep learning! (relevant for your projects)

CNN Features off-the-shelf: an Astounding Baseline for Recognition

(CVPR workshops, 2012)

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson





Where did this all start?

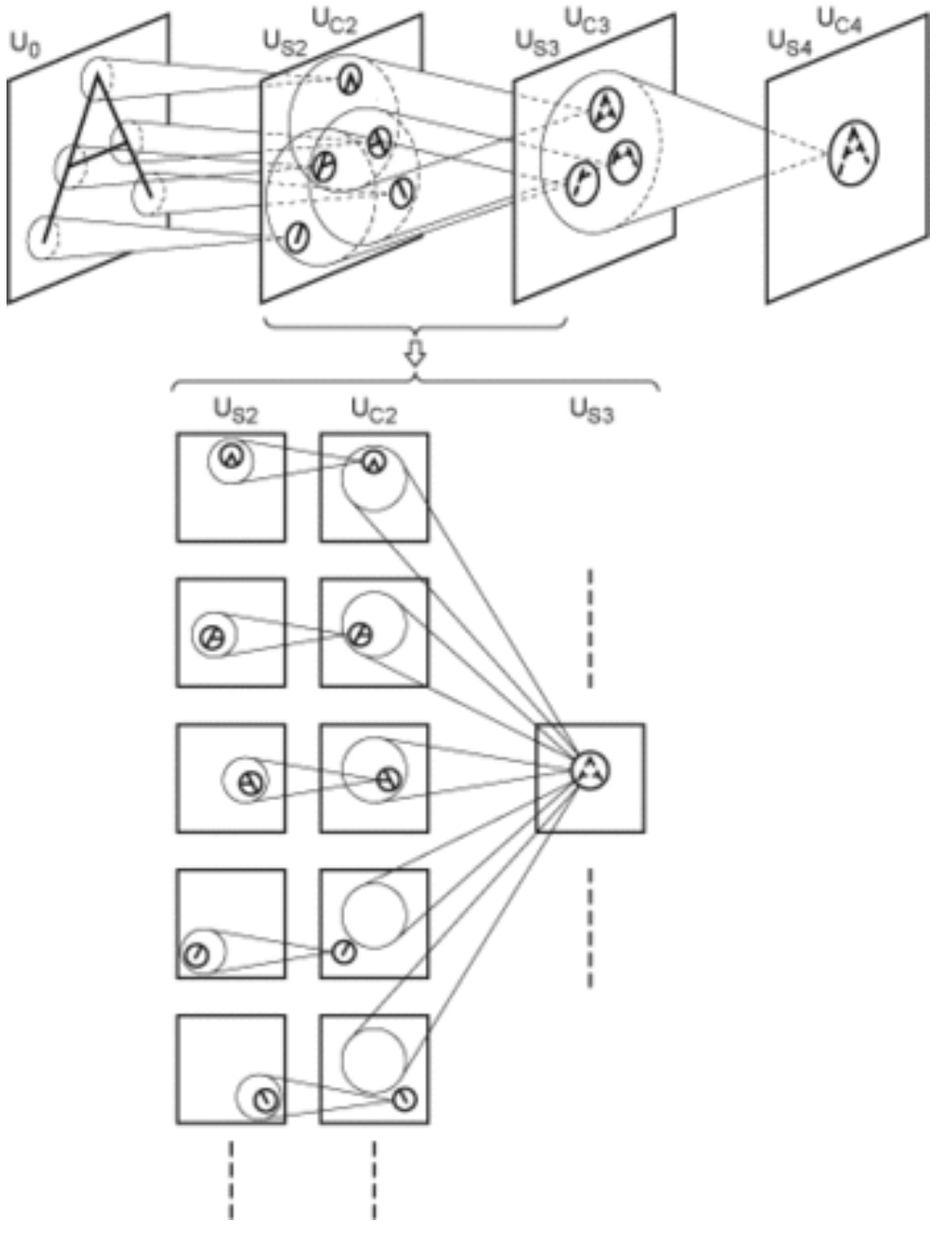
Hubel & Wiesel (1962)

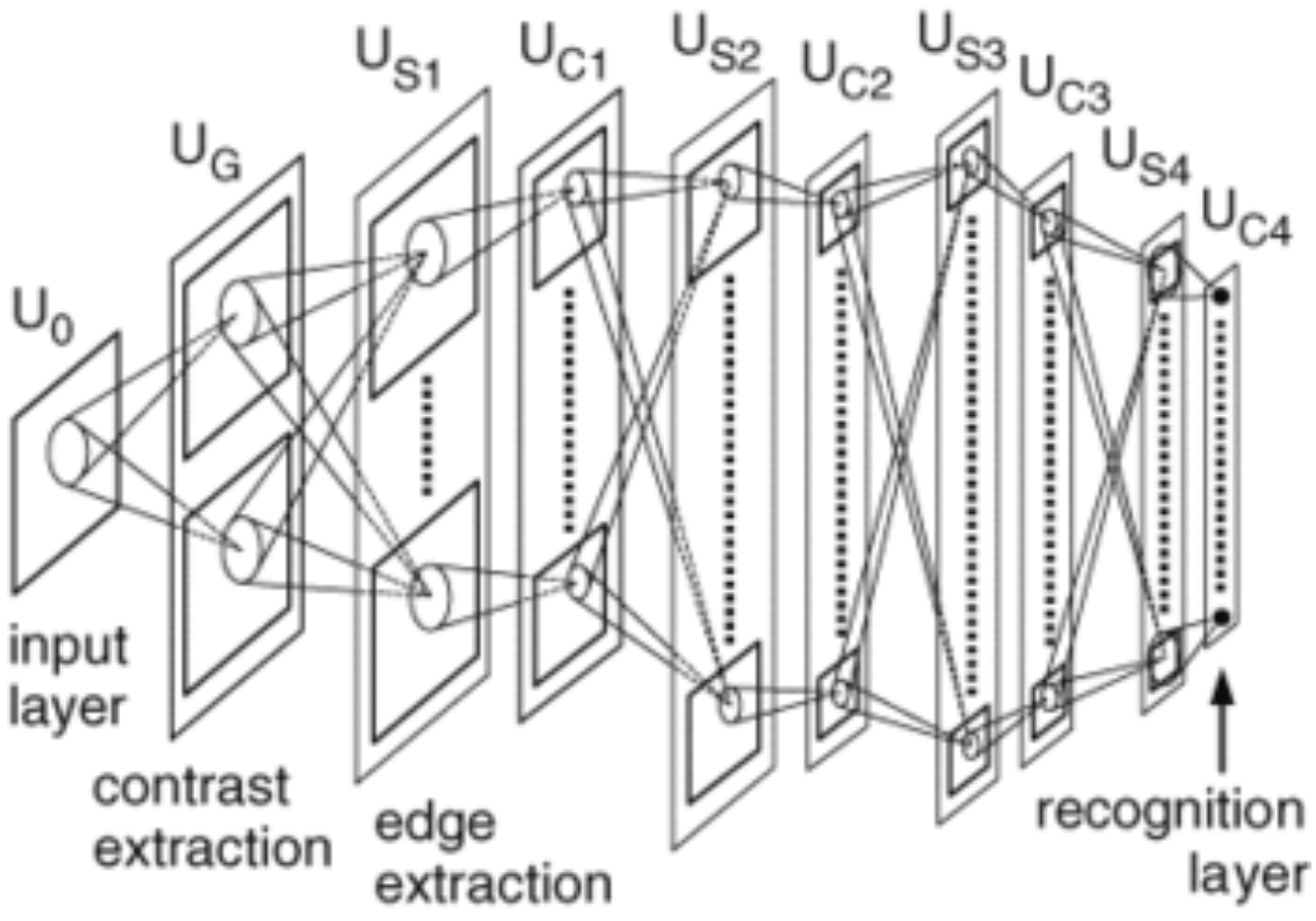
Insights about early image processing in the brain.

- Simple cells detect local features
- Complex cells pool local features in a retinotopic neighborhood

Earliest “deep” architecture

Neocognitron



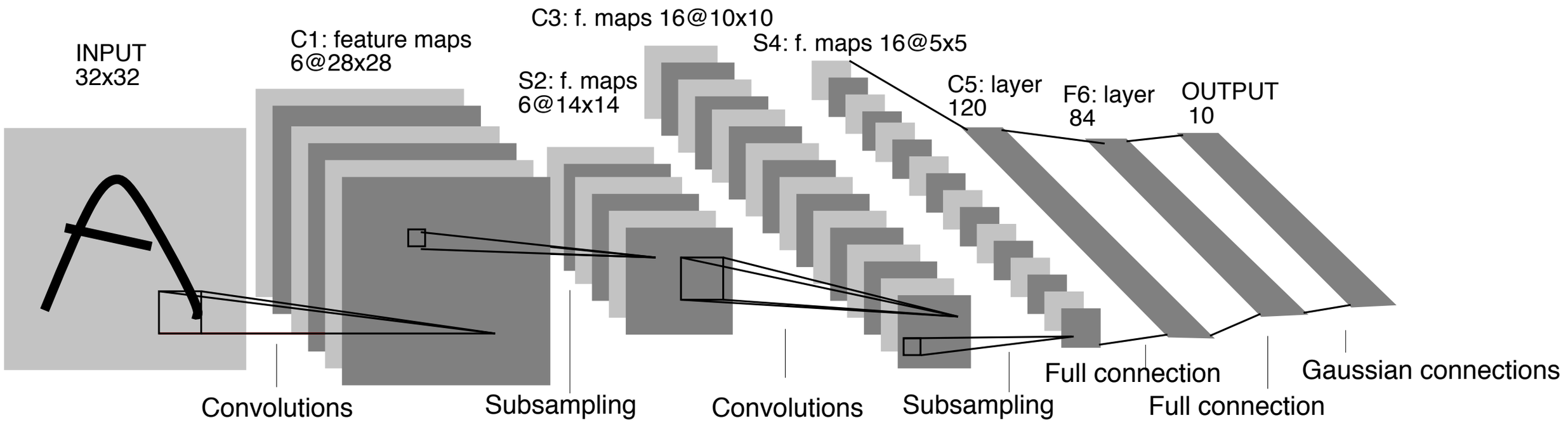


(Fukushima 1974-1982)

Original of current networks

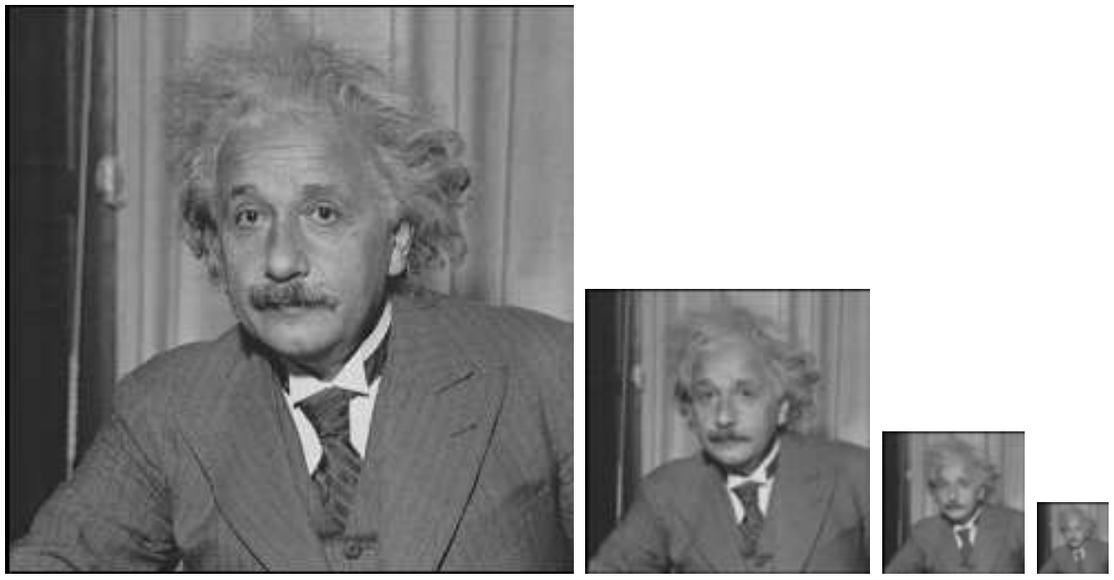
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner



Stack together convolution and pooling (avg + subsample) operations.
Why can’t this be whole story?

Recall: Gaussian pyramids



$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

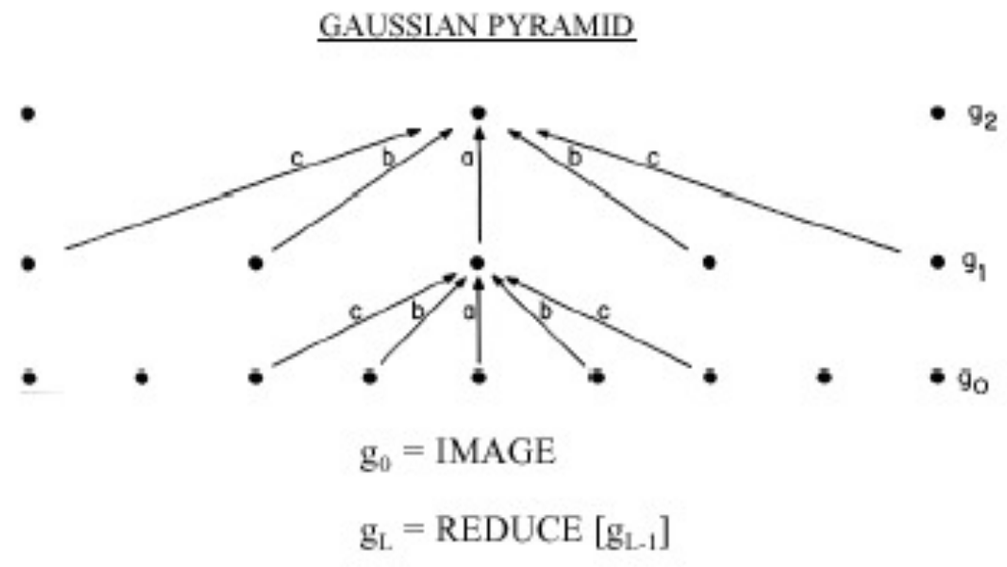
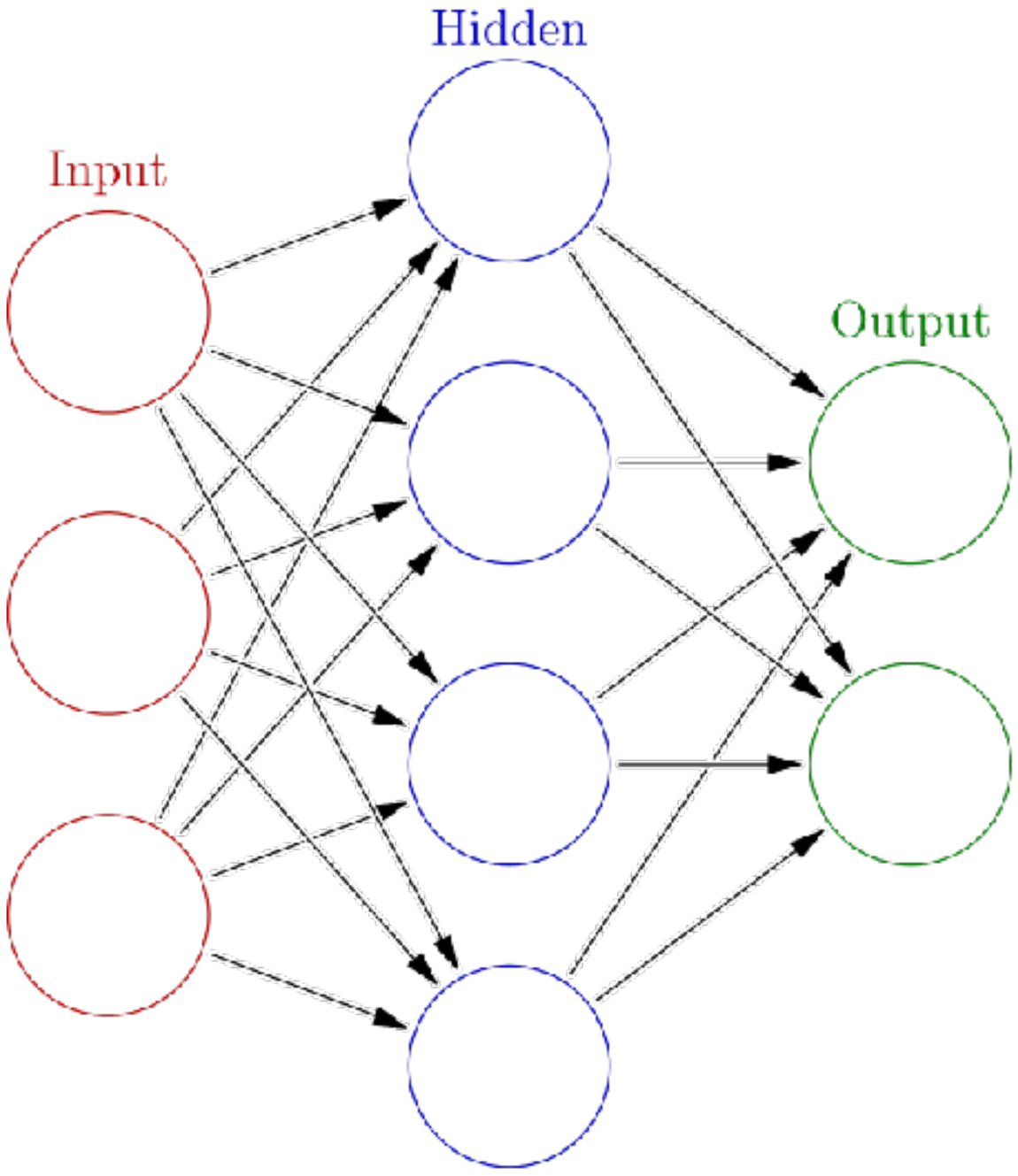
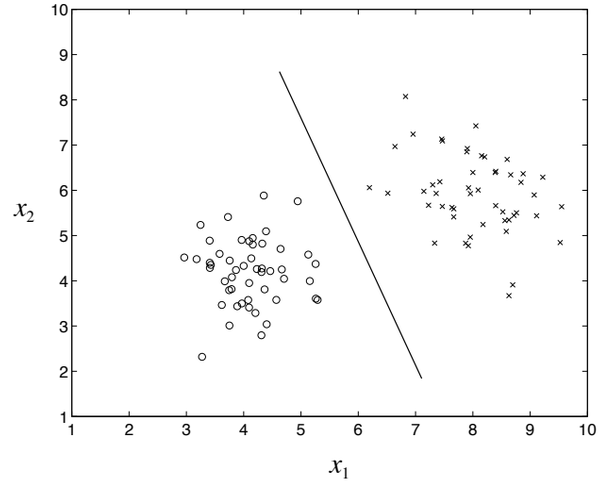


Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

Alternate perspective: neural networks



Recall: class-conditional Gaussians



$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1) + p(x|y=0)p(y=0)}$$

Plug in the following and simplify:

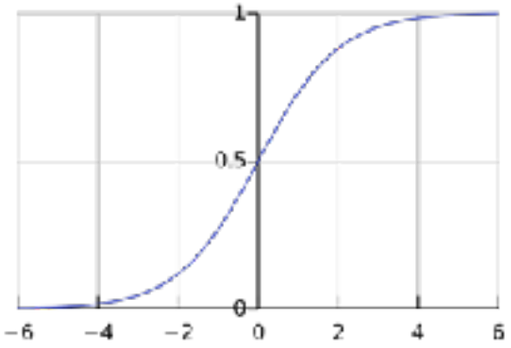
$$p(y=1) = .5$$

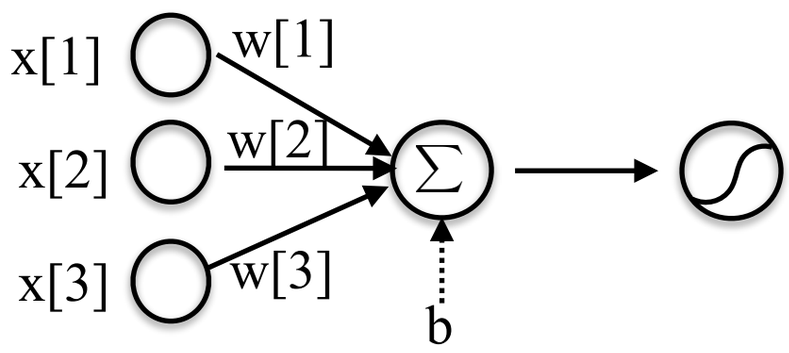
$$p(x|y) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)}$$

Recall: class conditional Gaussians

$$p(y=1|x) = sigmoid(w \cdot x + b)$$

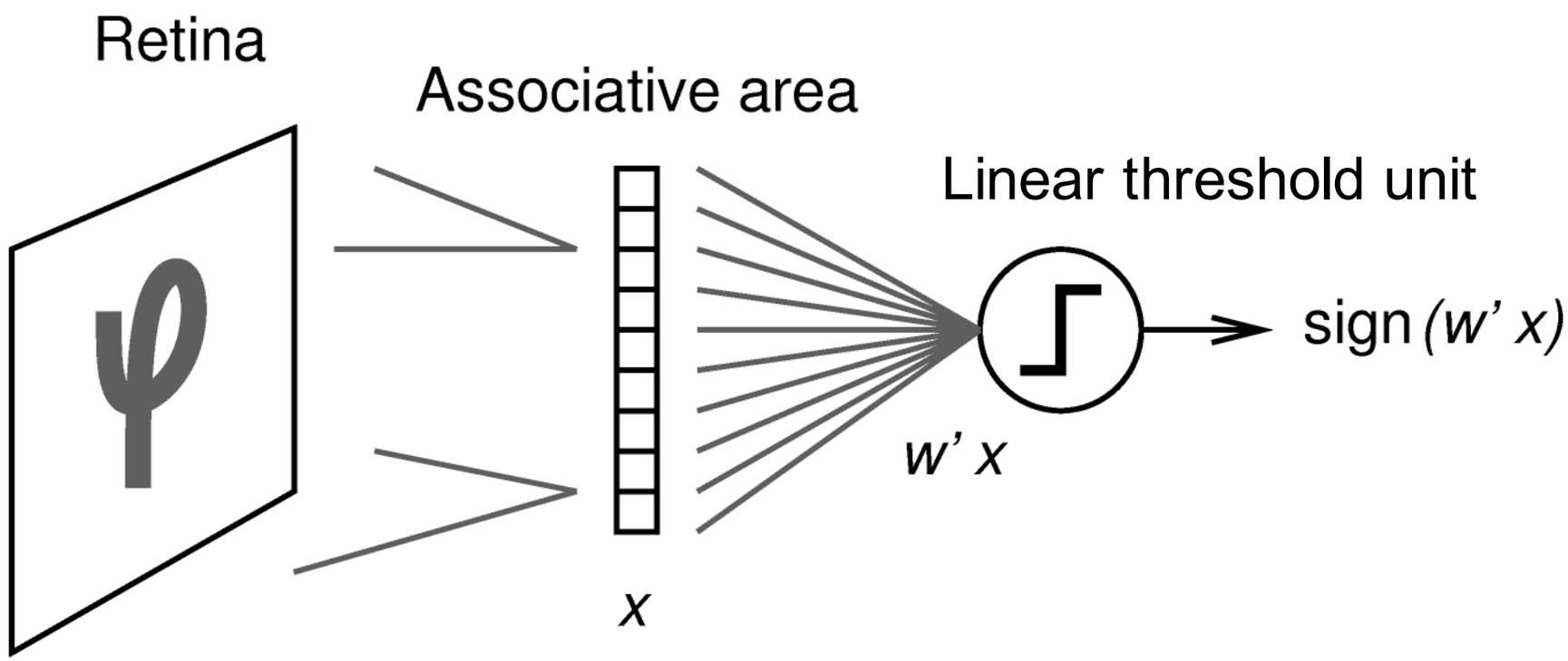
$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

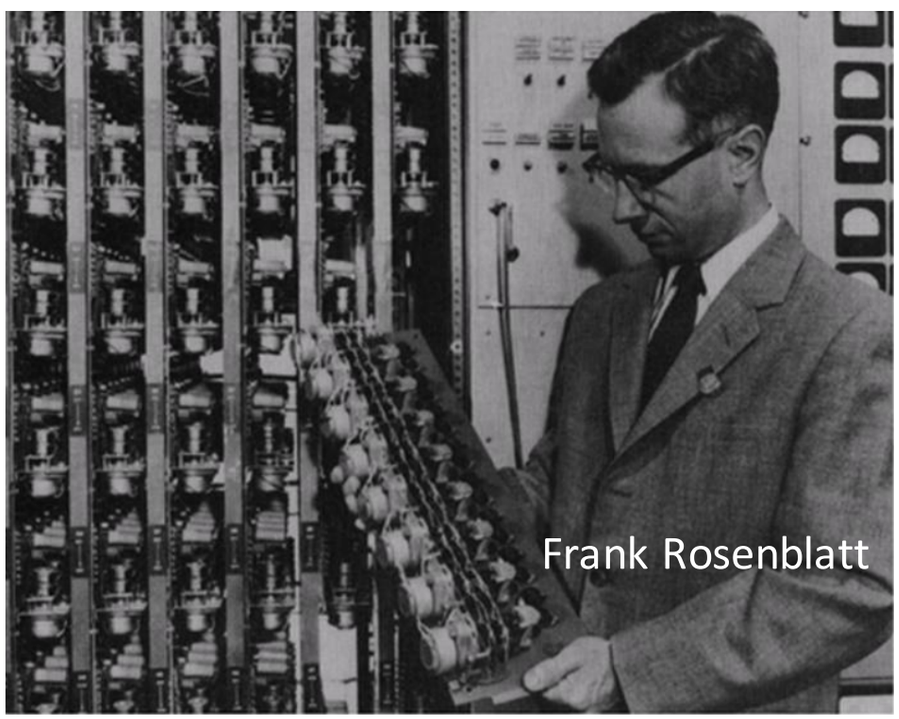




$$p(y=1|x) > .5 \quad \text{when} \quad f_w(x) = w \cdot x + b > 0, \quad w = \Sigma^{-1}(\mu_1 - \mu_0)$$

The perceptron

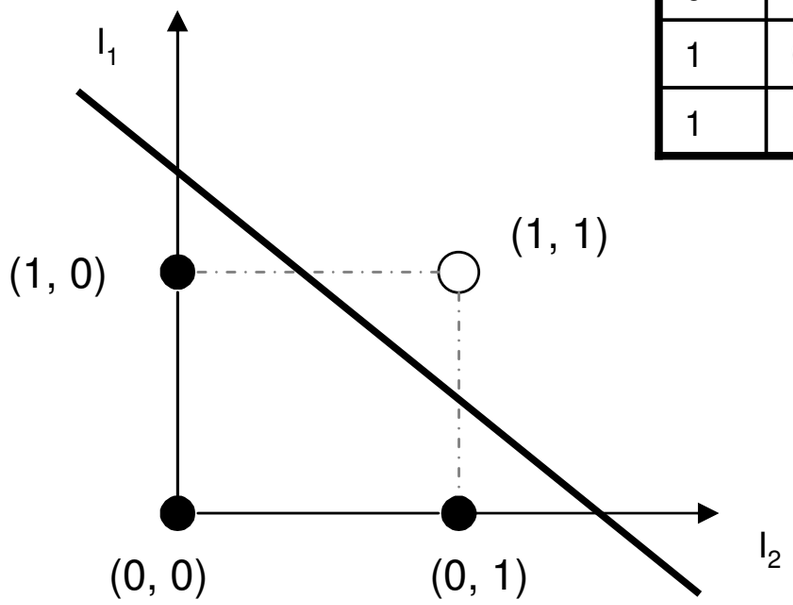




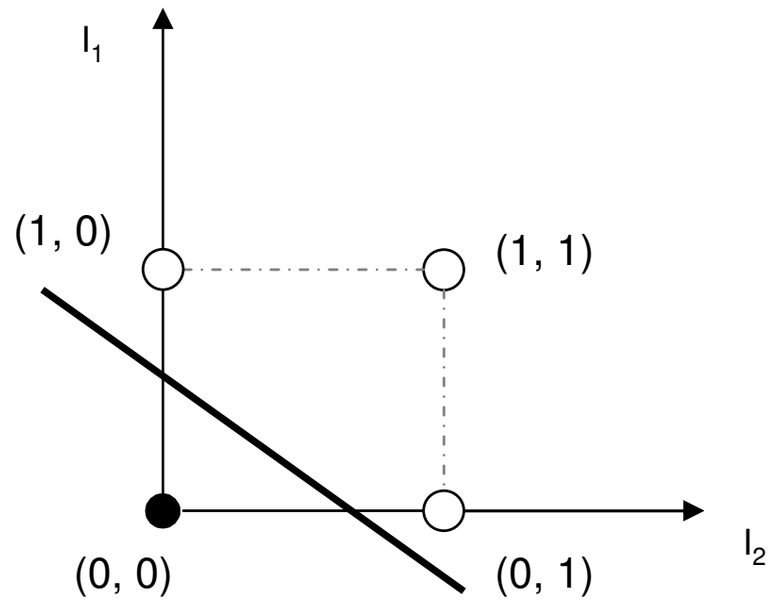
Percepton

If input features are binary, can model logical “and”s and “or”s

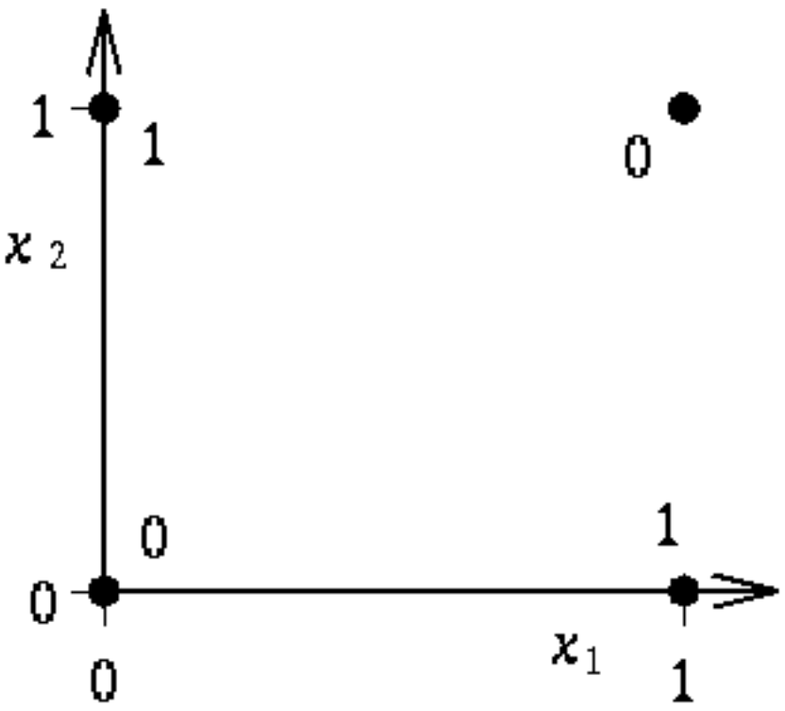
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



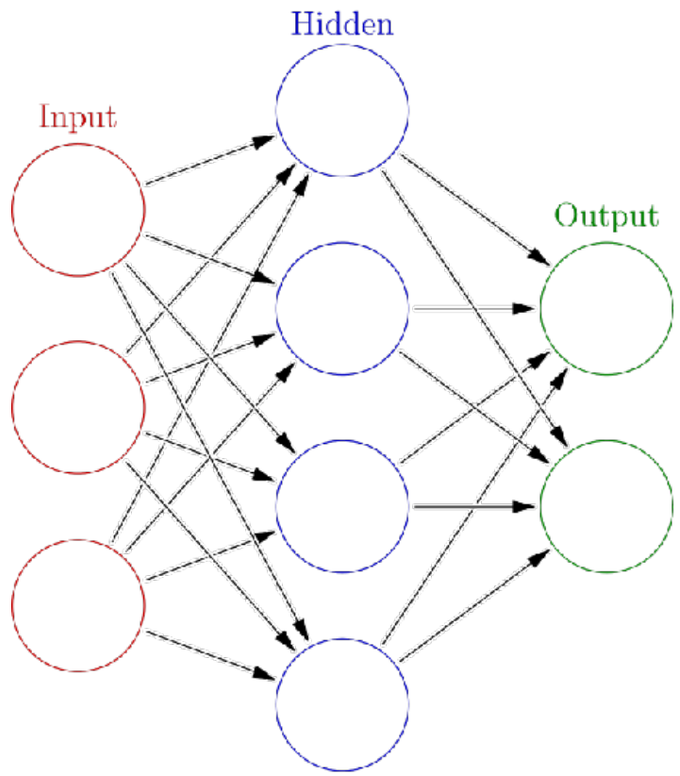
OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



What about “xors”?



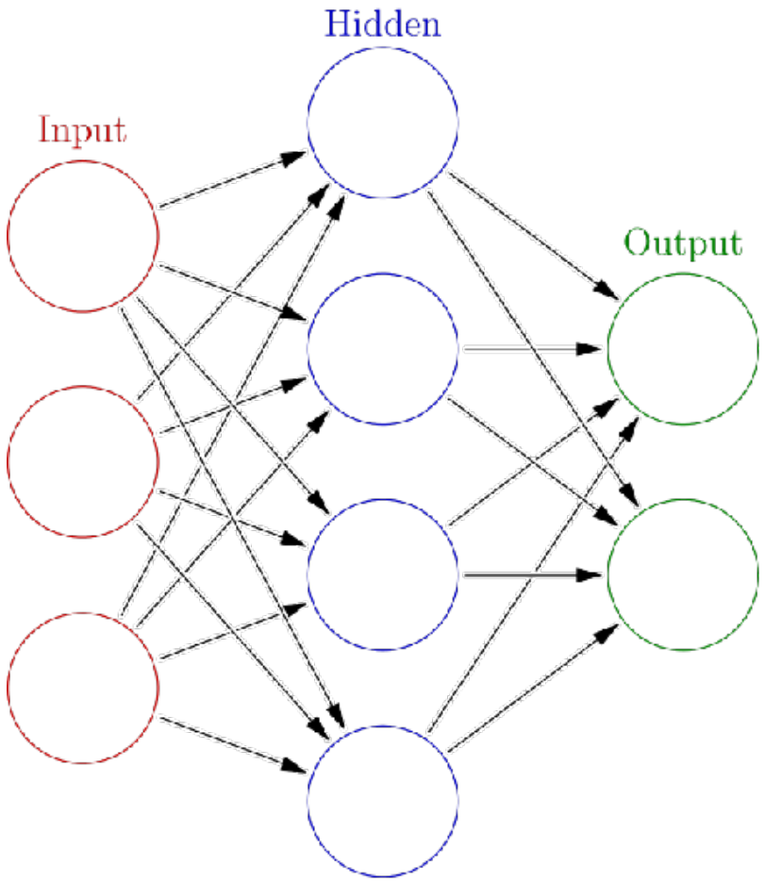
Multilayer perceptons

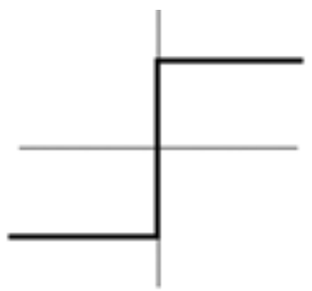


Can model more complex “circuits”

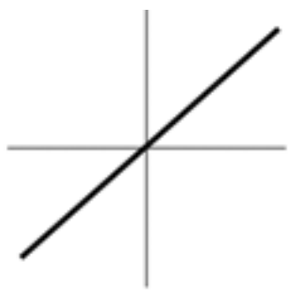
Activation functions

$$z_i = \phi(b_i + \sum_{j \neq i} w_{ij} z_j)$$

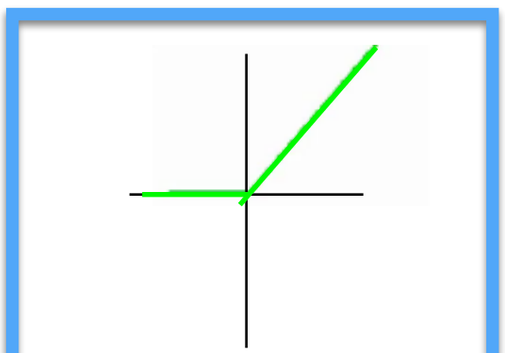




Step Function

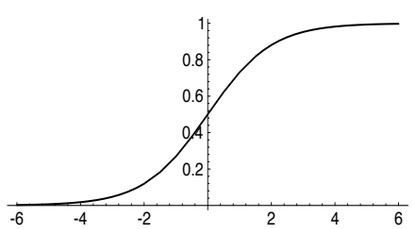


Linear Function

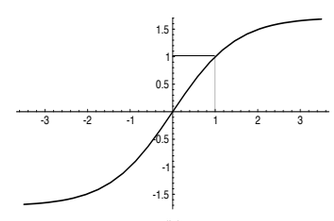


rectified linear (RelU)

Threshold Logic



sigmoid

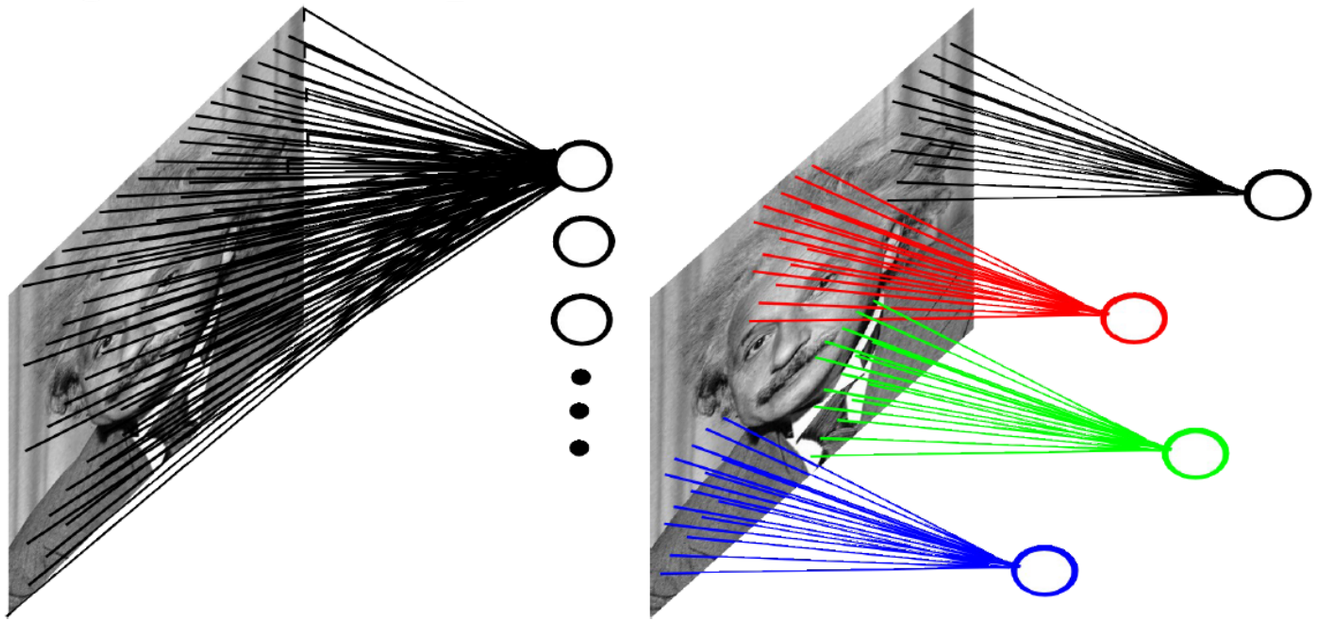


tanh

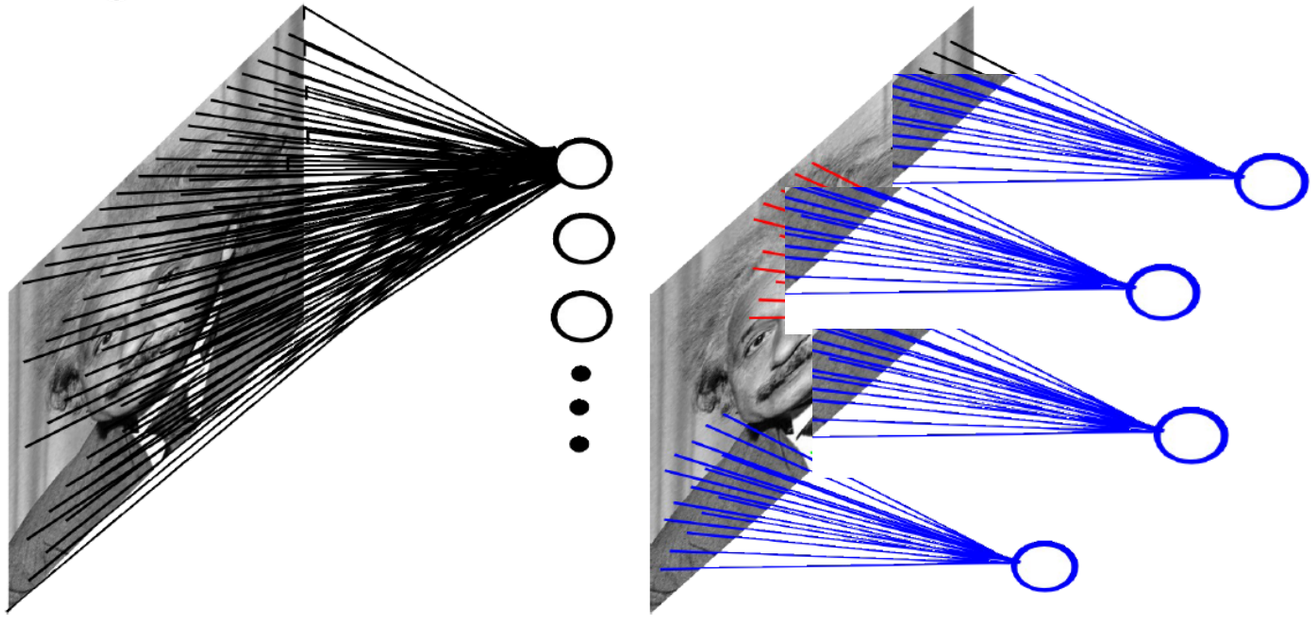
$f(x) = 1/(1 + e^{-x})$.

What's the difference between a standard "neural network" and a "convolutional neural network"?

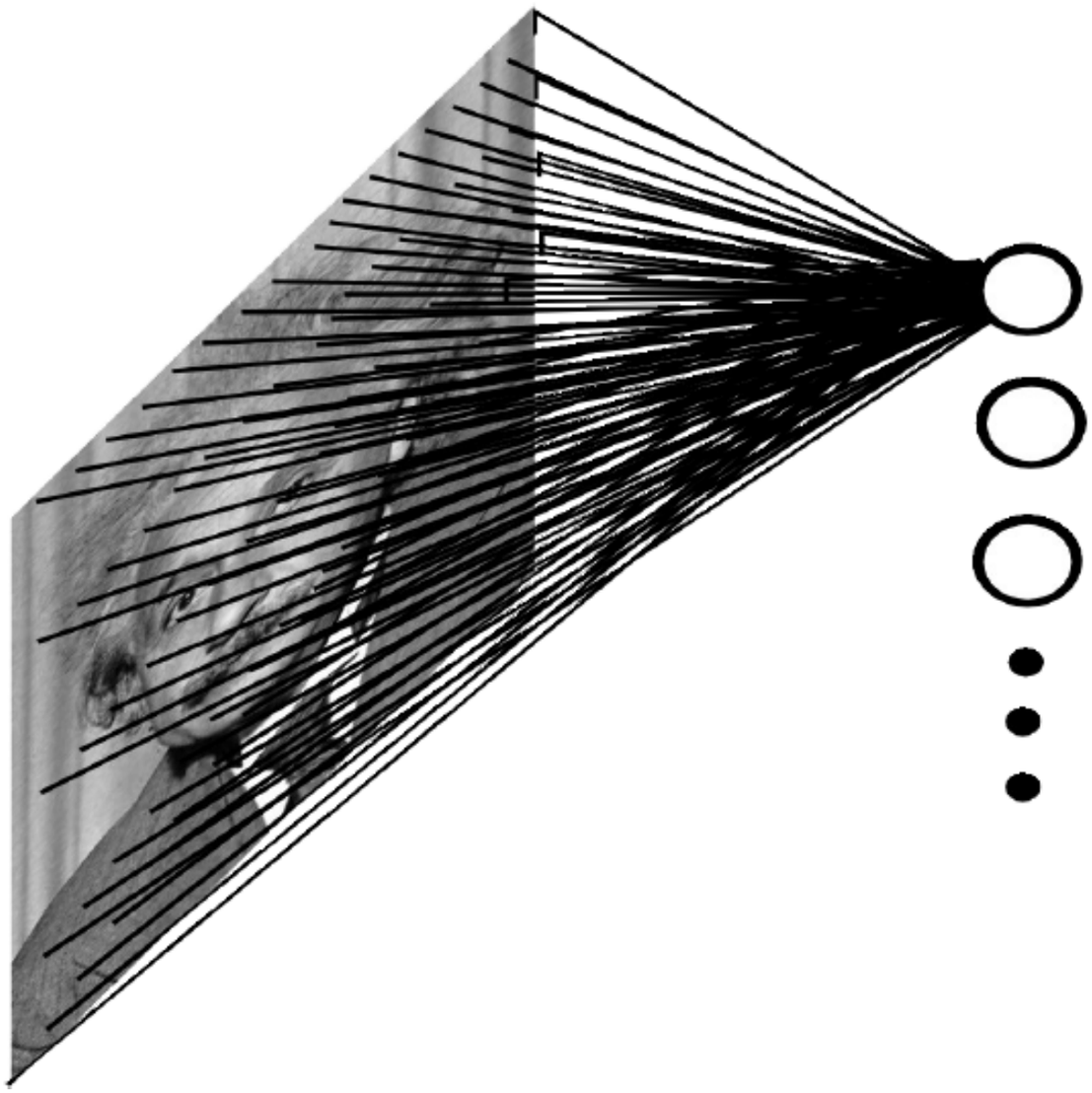
1. Sparsity through local receptive feilds

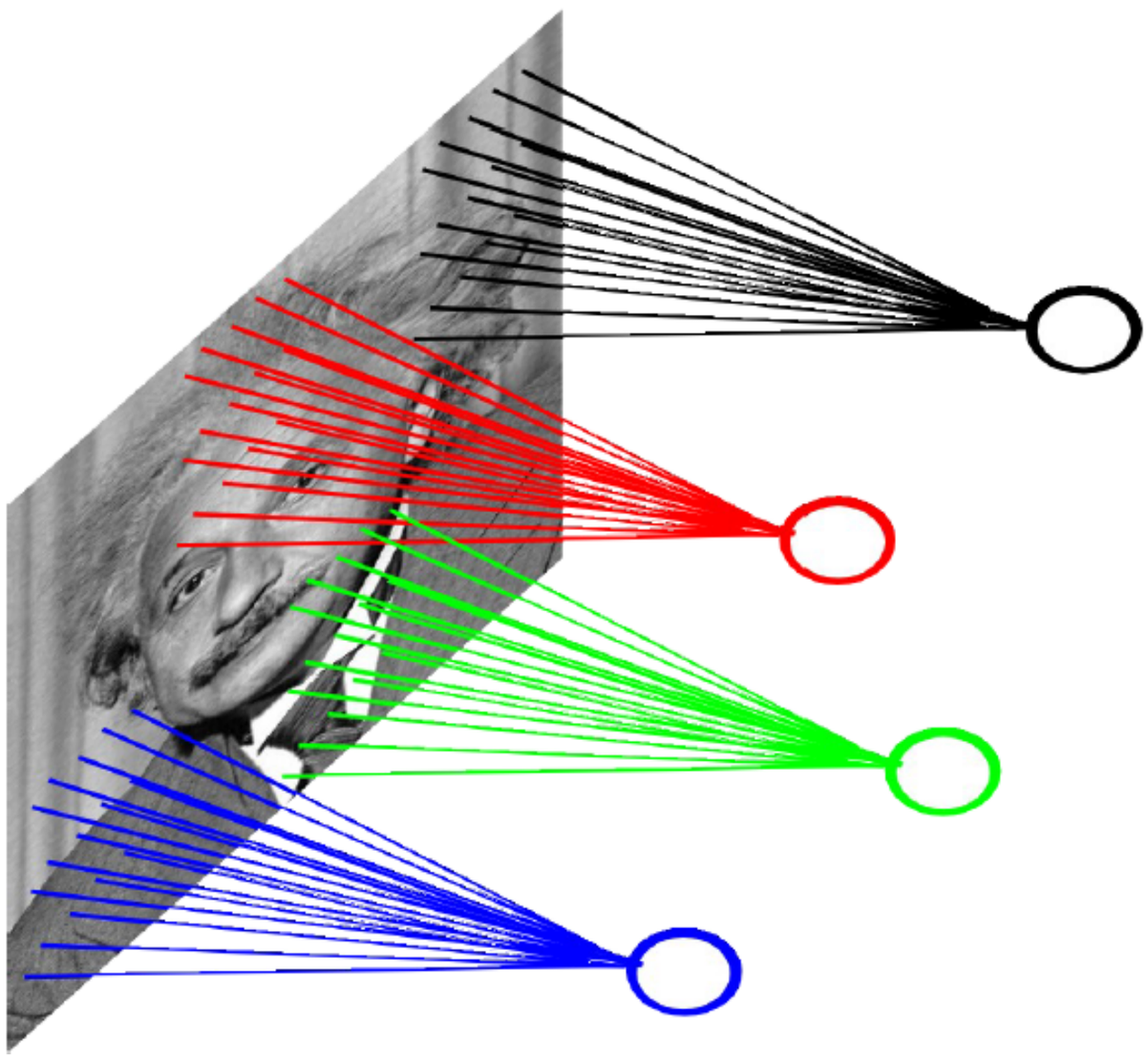


2. Weight sharing across locations

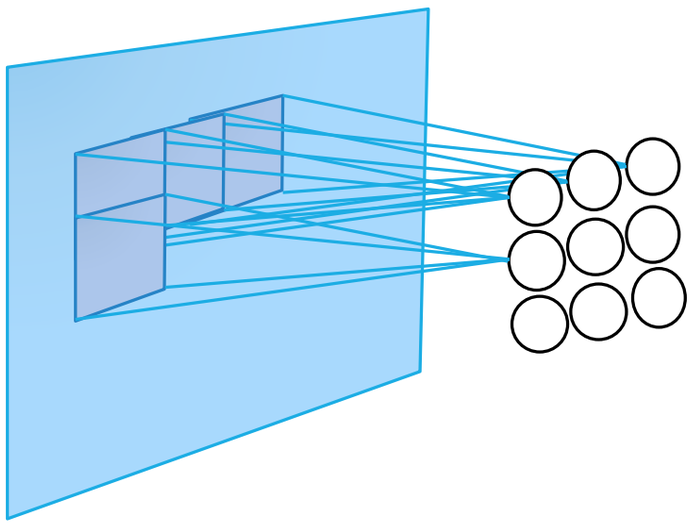


Local *receptive feilds*



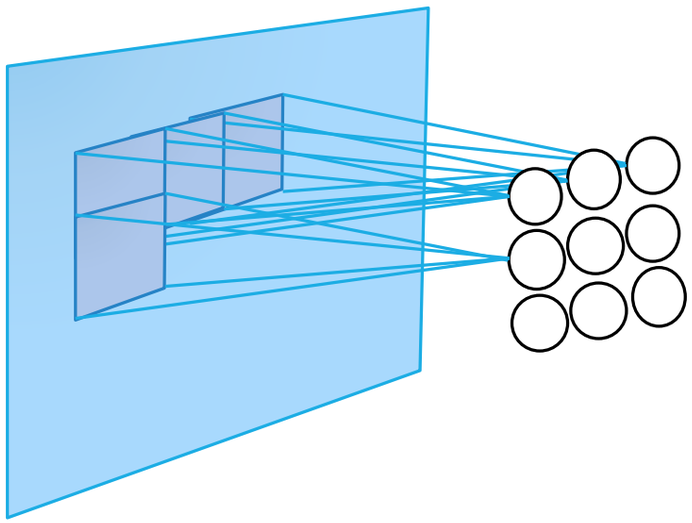


Last puzzle peices: pooling



Name	Pooling formula
Average pool	$\frac{1}{s^2}\sum x_i$
Max pool	$\max\{x_i\}$

Last puzzle peices: pooling + normalization



Name	Pooling formula
Average pool	$\frac{1}{s^2}\sum x_i$
Max pool	$\max\{x_i\}$
L2 pool	$\sqrt{\frac{1}{s^2}\sum x_i^2}$
L_p pool	$\left(\frac{1}{s^2}\sum \lvert x_i\rvert^p\right)^{\frac{1}{p}}$

Contrast normalization

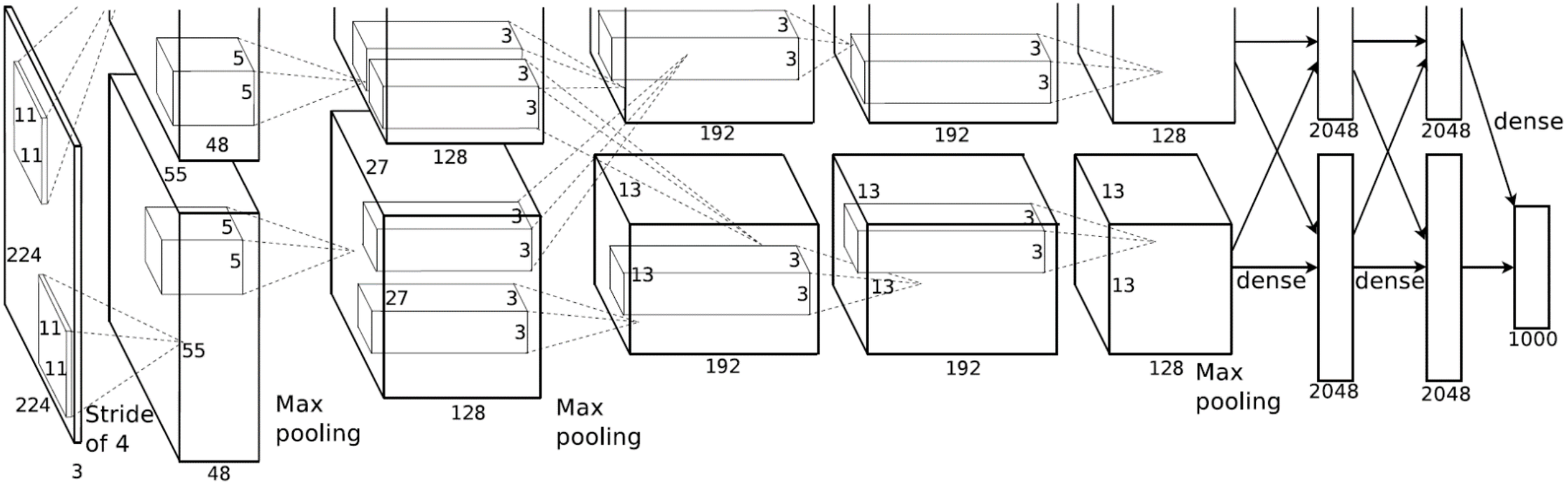
- Subtracting a low-pass smoothed version of the layer
- Just another convolution in fact (with fixed coefficients)
- Lots of variants (per feature map, across feature maps, …)
- Divisive normalization

Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale DAGs, recurrence, LSTMs
- Why does it work so well?

Some popular networks

AlexNet

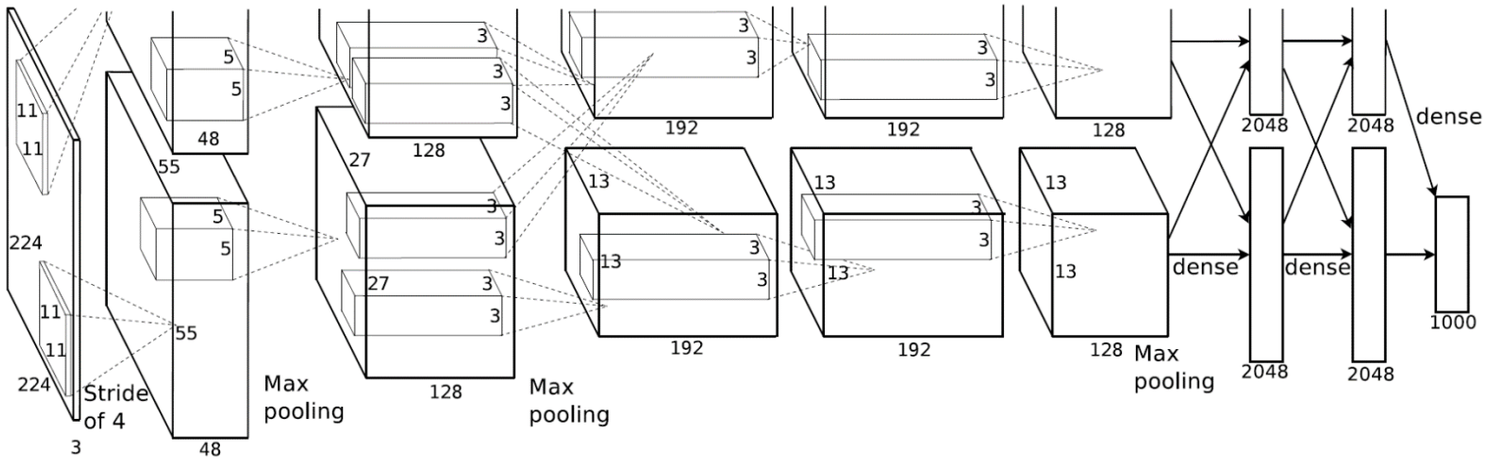


- Structure (conv-relu-maxpool-norm)^3-linear-relu-linear-relu-linear
- Very good implementation, running on two GPUs.
- ReLU transfer function. Dropout trick.
- Also trains on full ImageNet (15M images, 15000 classes)

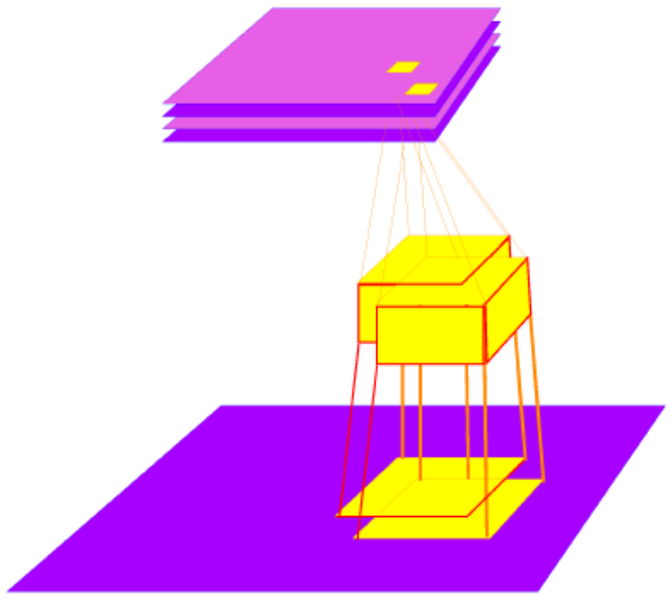
(Kirzhevsky, Sutskever, Hinton, 2012)

Aside: scanning-window CNNs

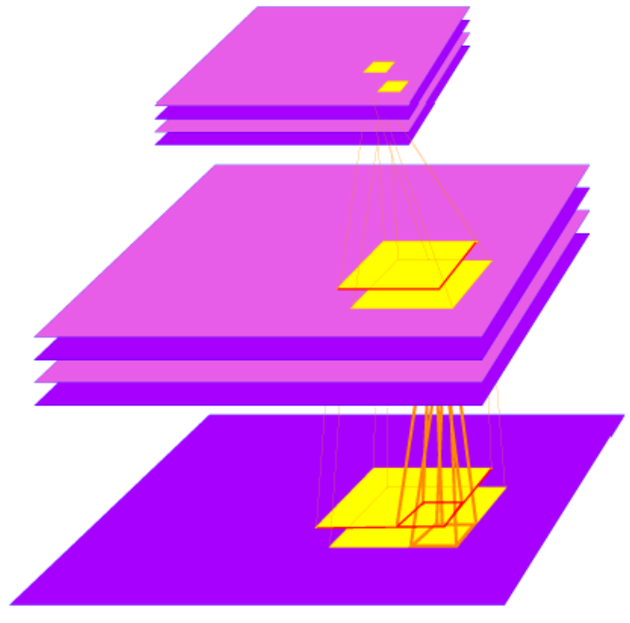
Yan Lecun: “There is no fully-connected layer, only a 1x1xN convolutional layer!”



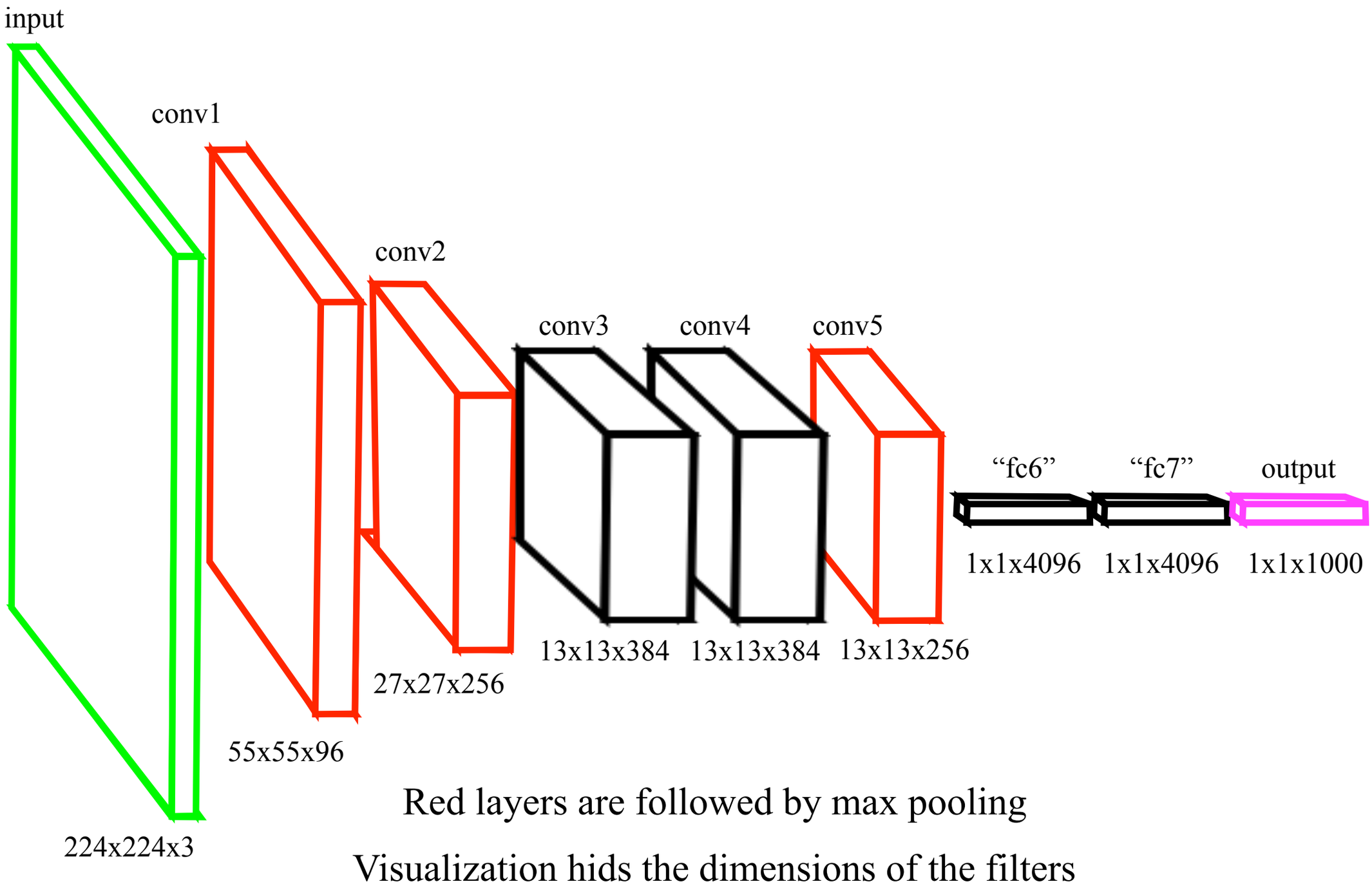
Wrong way



Right way



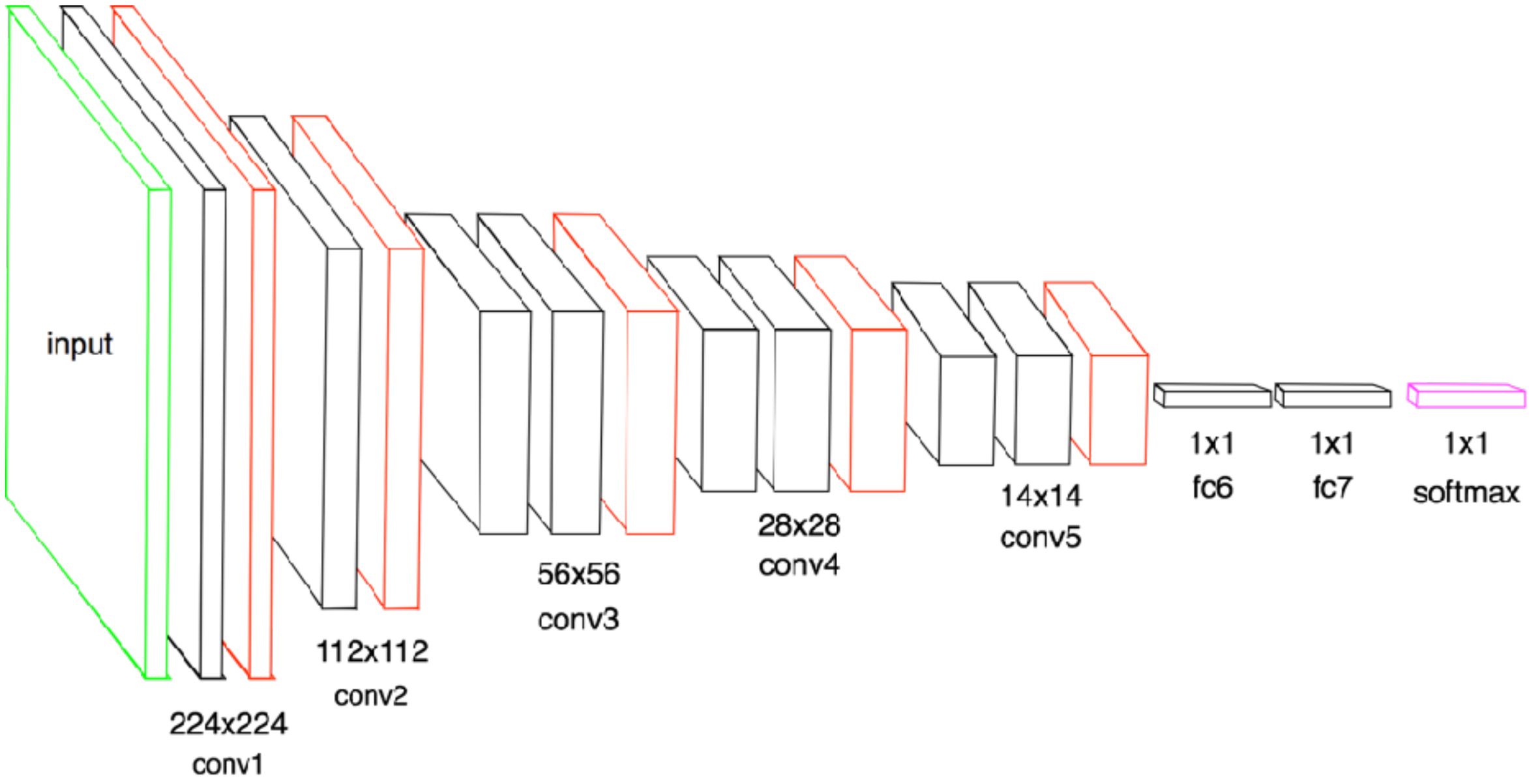
A better visualization of AlexNet



Red layers are followed by max pooling

Visualization hids the dimensions of the filters

VGG19



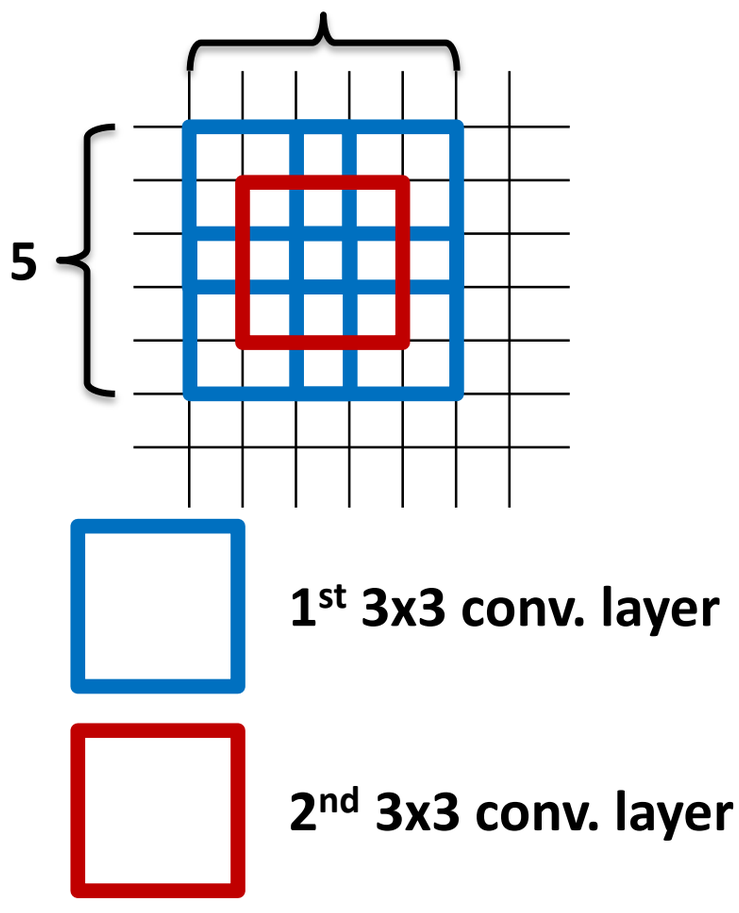
All filter dimensions 3x3 except fc6 (which uses 7x7)

People *still* misunderstand this

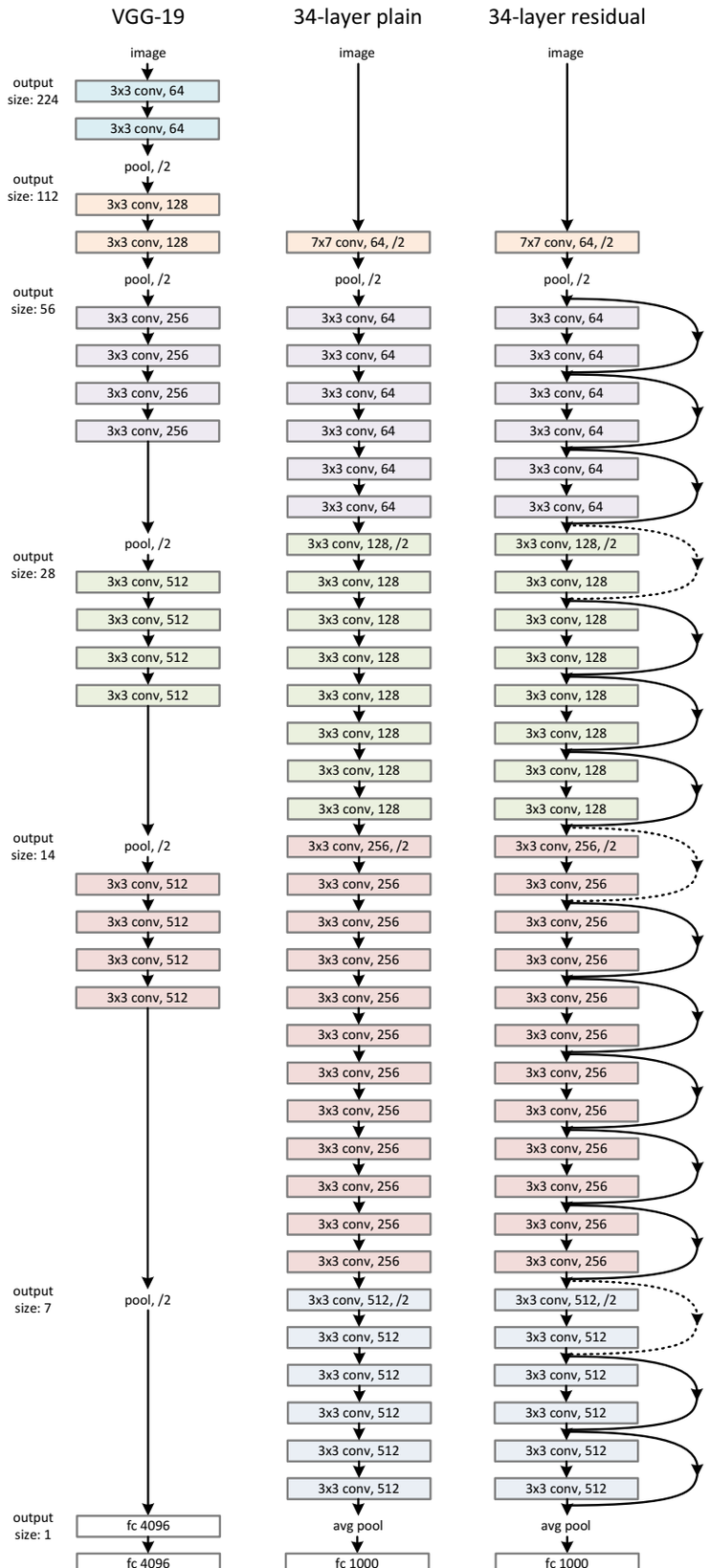
(Loosely) exploit associate property of convolutions

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net



Residual Net



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale DAGs, recurrence, LSTMs
- Why does it work so well?

Supervised training

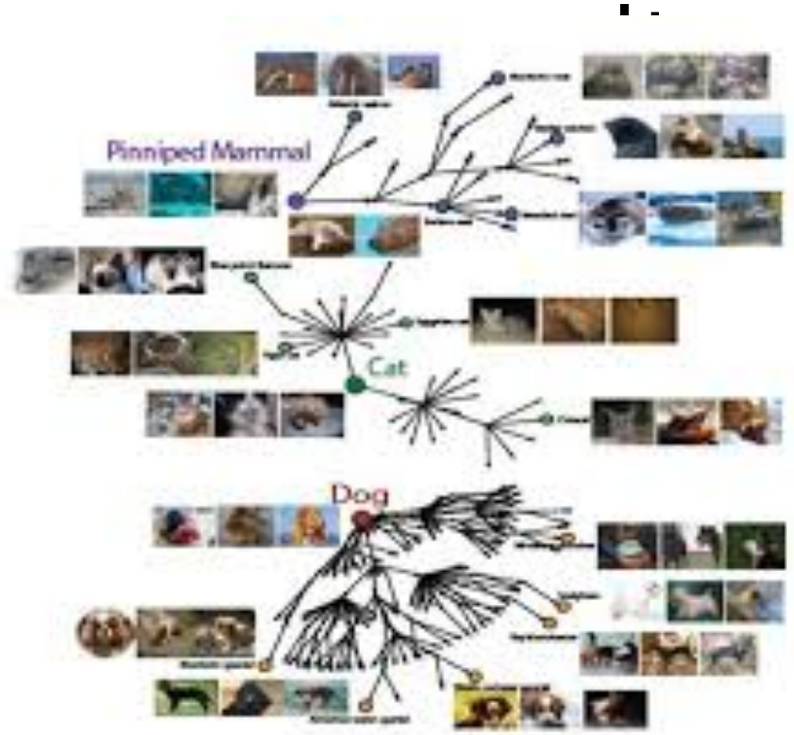
$$\{x_i, y_i\}$$

$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

Imagenet large-scale visual recognition challenge

1000 classes, ~1000 examples per class

Imagenet 2014 image classification



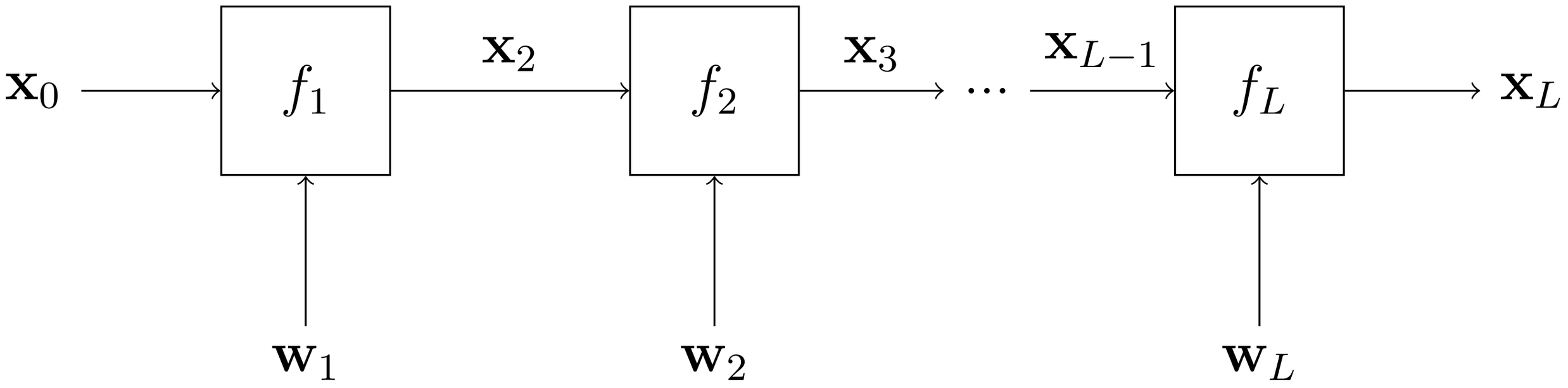
Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

Human top-5 error: 5.1 %

MatConvNet

Convolutional Neural Networks for MATLAB

Andrea Vedaldi Karel Lenc



http://www.vlfeat.org/matconvnet/#pretrained

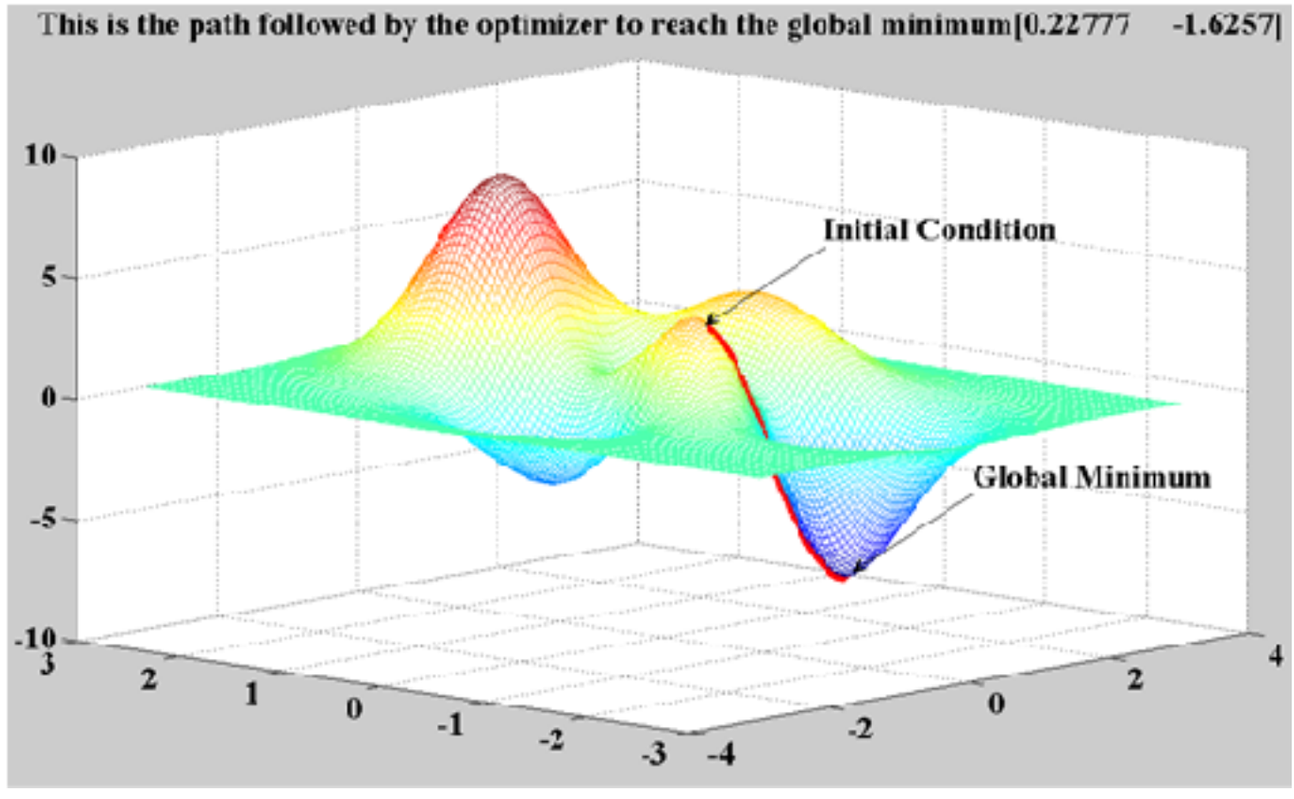
Great source of “off-the-shelf” state-of-the-art features for Matlab

User manual is cleanest “hands-on” explanation of backprop I’ve seen

Gradient descent

$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

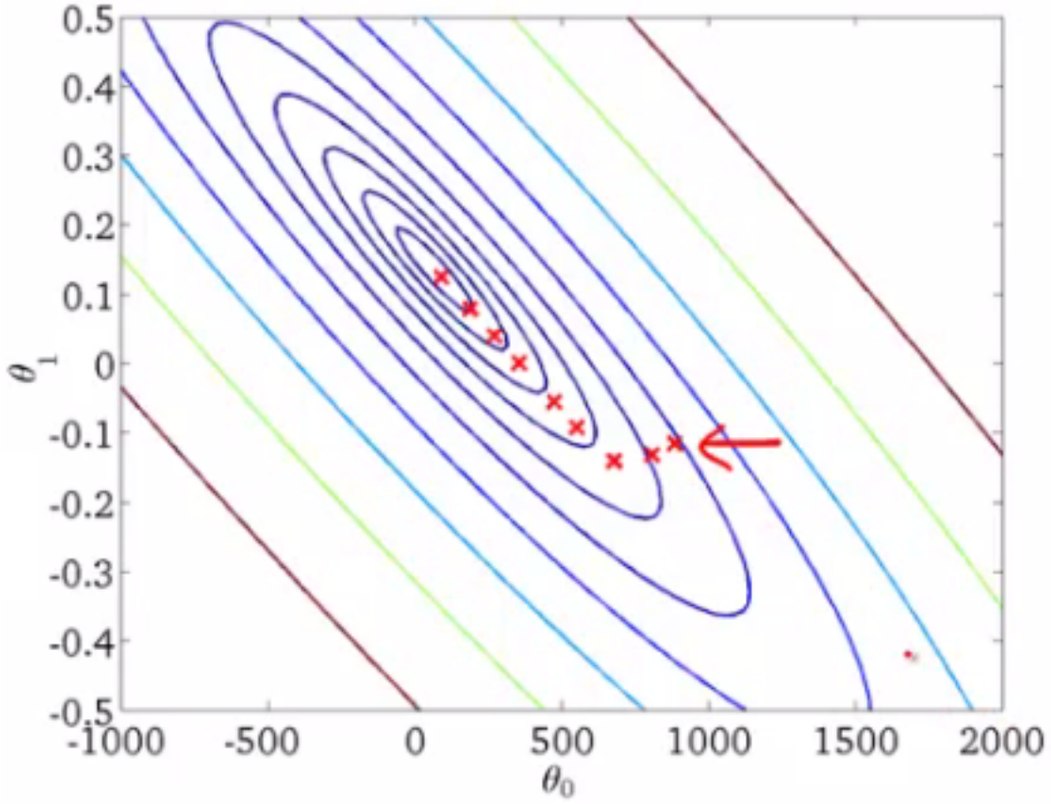
$$w := w - step * (w + \sum_i (f_w(x_i) - y_i) \frac{\partial f_w(x_i)}{\partial w})$$

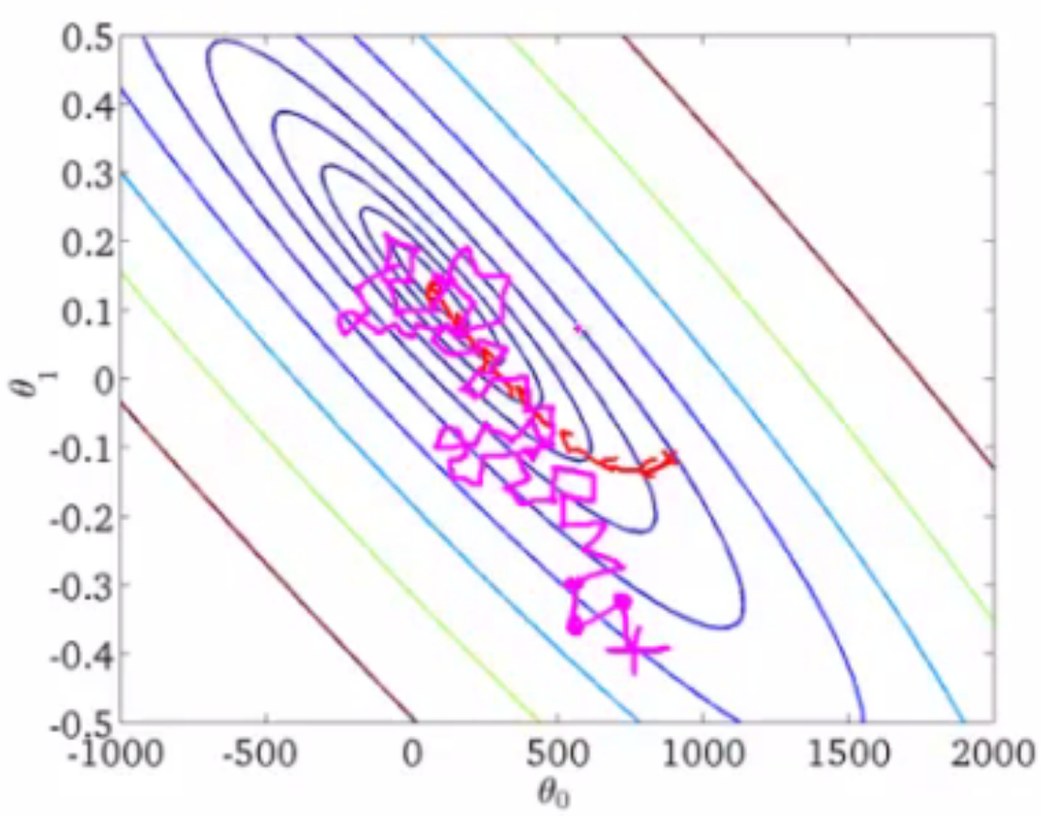


Stochastic gradient descent

$$\min_w \frac{\lambda}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

$$w := w - step * \left(\lambda w + (f_w(x_i) - y_i)\frac{\partial f_w(x_i)}{\partial w}\right)$$





Crucial parameters for tuning: learning rate (step) and weight decay (lambda)

Optimization

Appears to be a significant hurdle for training

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

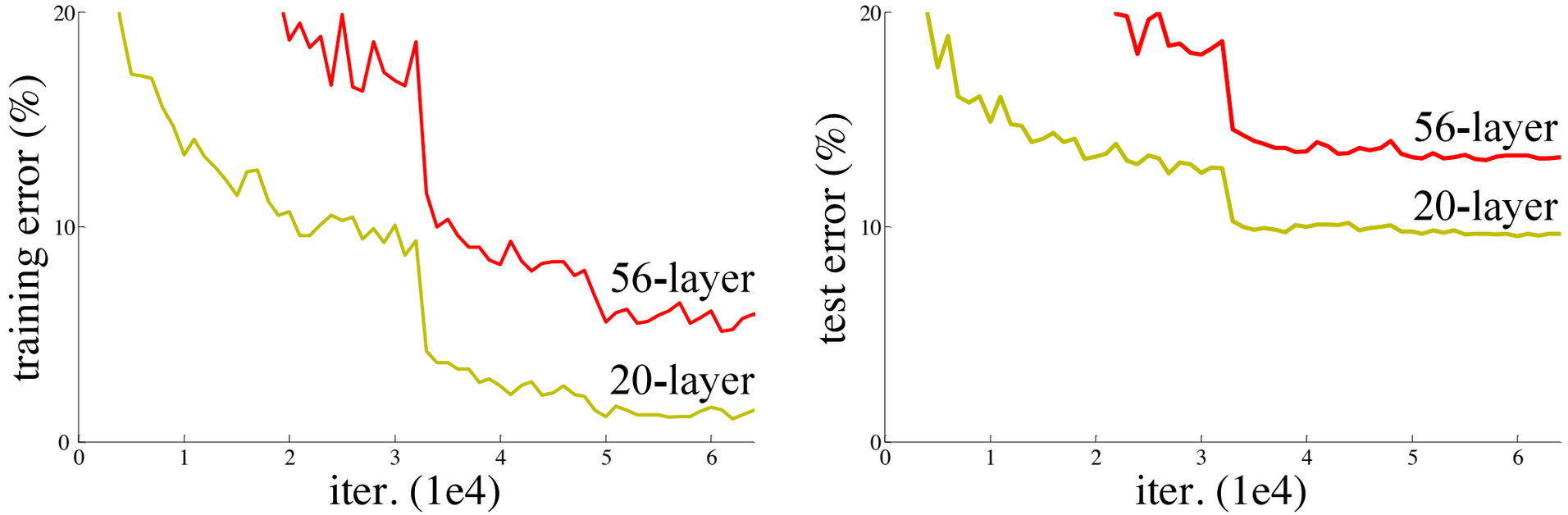
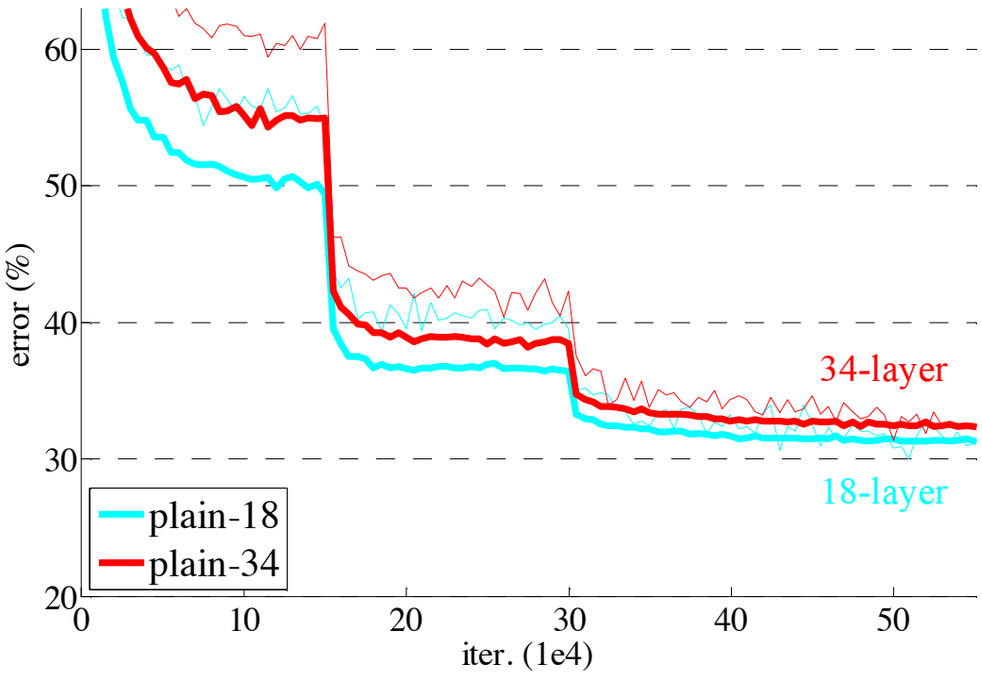


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Intuition: bias deep models to behave like shallow models during learning



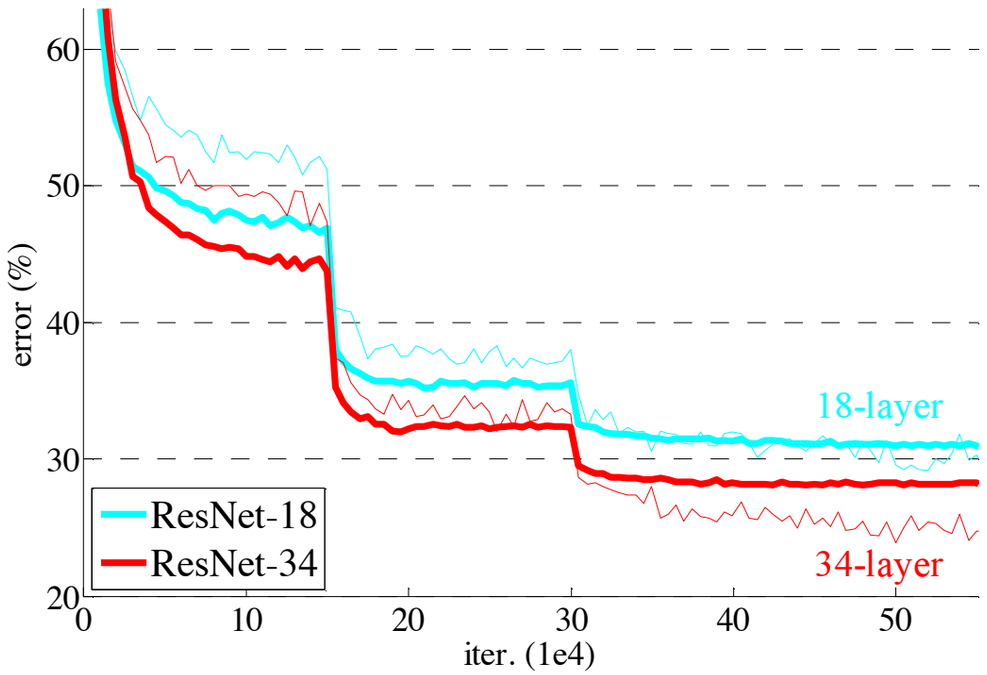
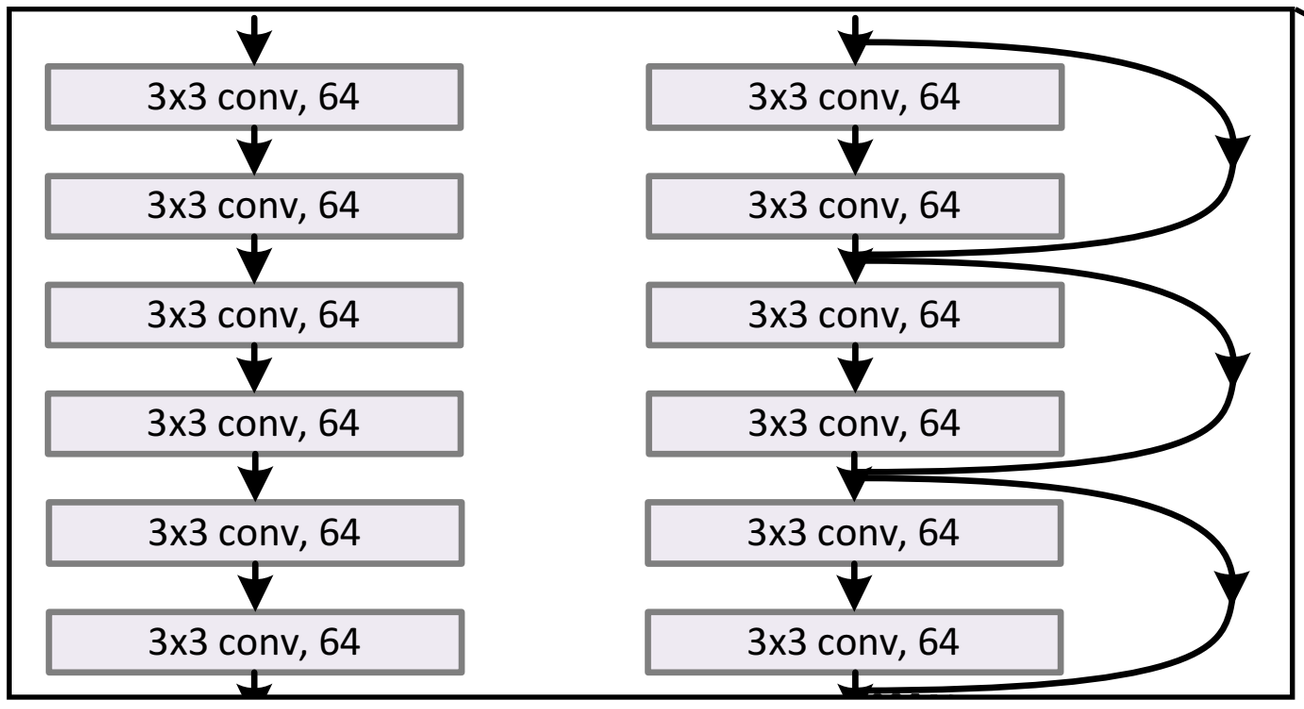
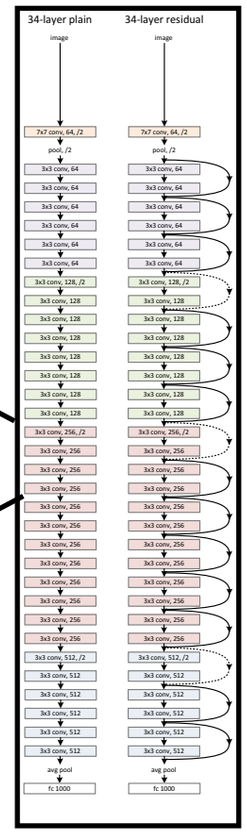


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.





Efficient BackProp

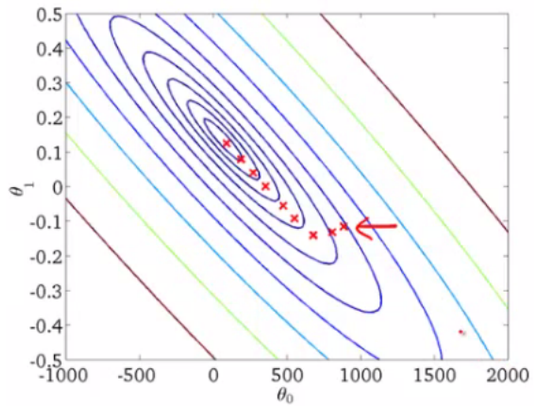
Yann LeCun[1], Leon Bottou[1], Genevieve B. Orr[2], and Klaus-Robert Müller[3]

[Nice discussion of optimization issues in above paper]

$$\min_w E(w)$$

1. First order method (gradient descent)

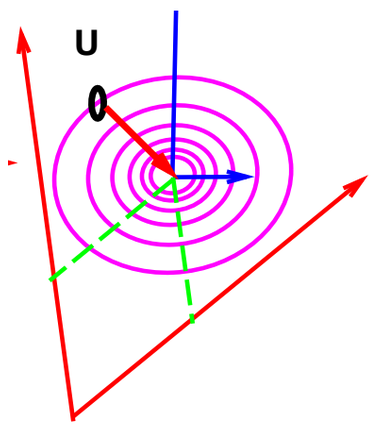
$$w := w - \alpha g, \quad g = \nabla_w E$$



2. Second order method (Newton's method)

https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

$$w := w - H^{-1} g$$



Intuition: build second-order behaviour into SGD by normalizing variables (zero-mean, identity covariance)

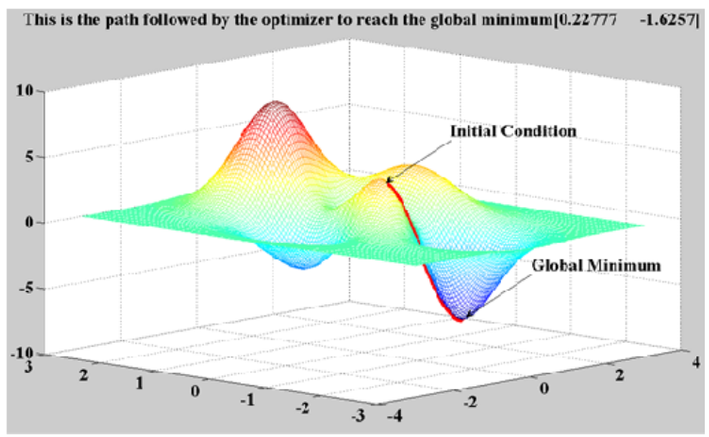
[cf. Batch Normalization, Ioffe et al]

(Mini) batch learning

$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

$$w := w - step * \left(w + \sum_{i \in MiniB} (f_w(x_i) - y_i) \frac{\partial f_w(x_i)}{\partial w} \right)$$

Learn from batches of training data (rather than a single example or full dataset)



Crucial observation: updates are more statistically reliable when data $\{x_i\}$ in batch are uncorrelated

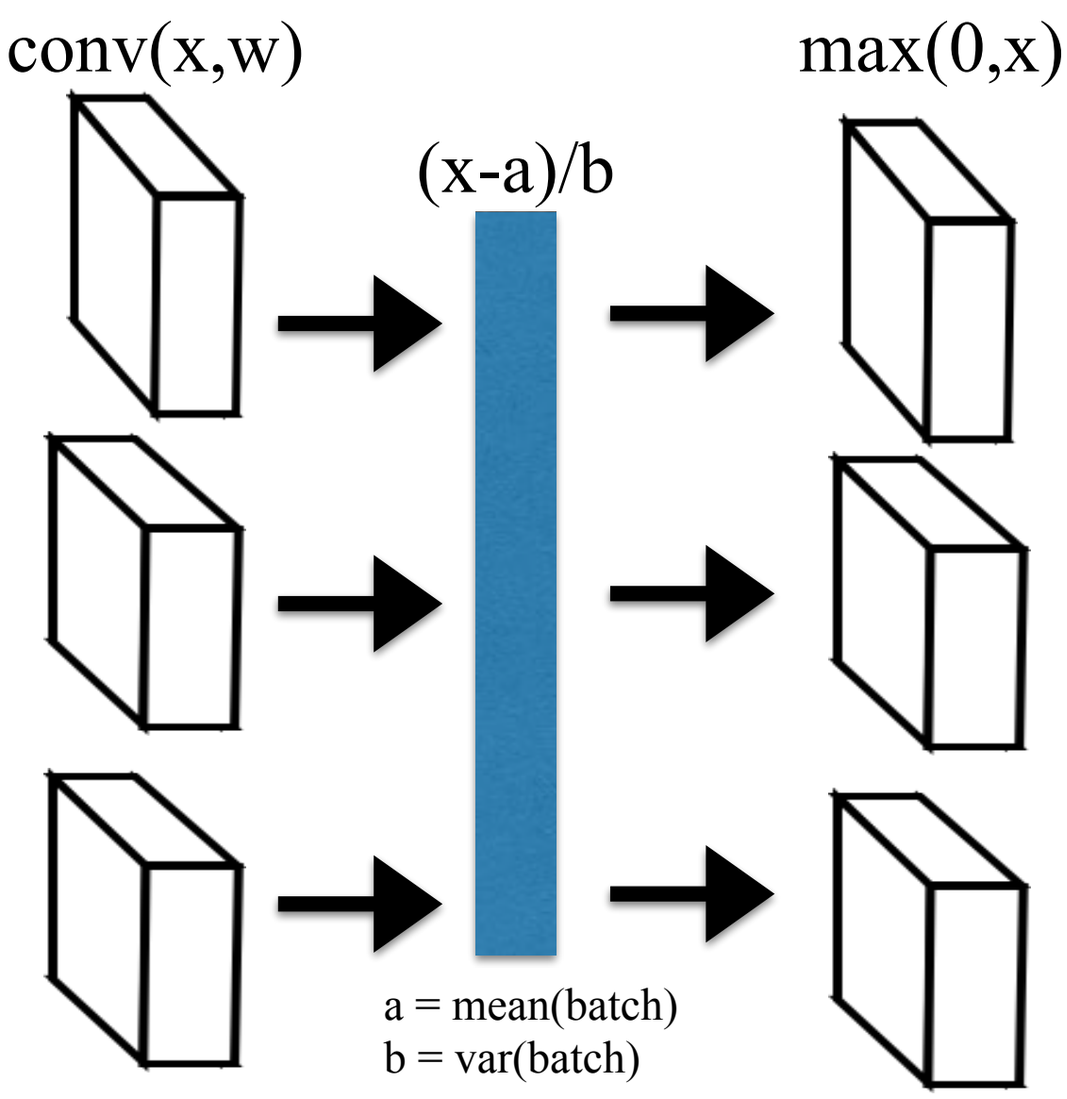
In practice, *randomly permute* training examples

Challenging to do when learning from patches in images or frames from videos

Batch normalization

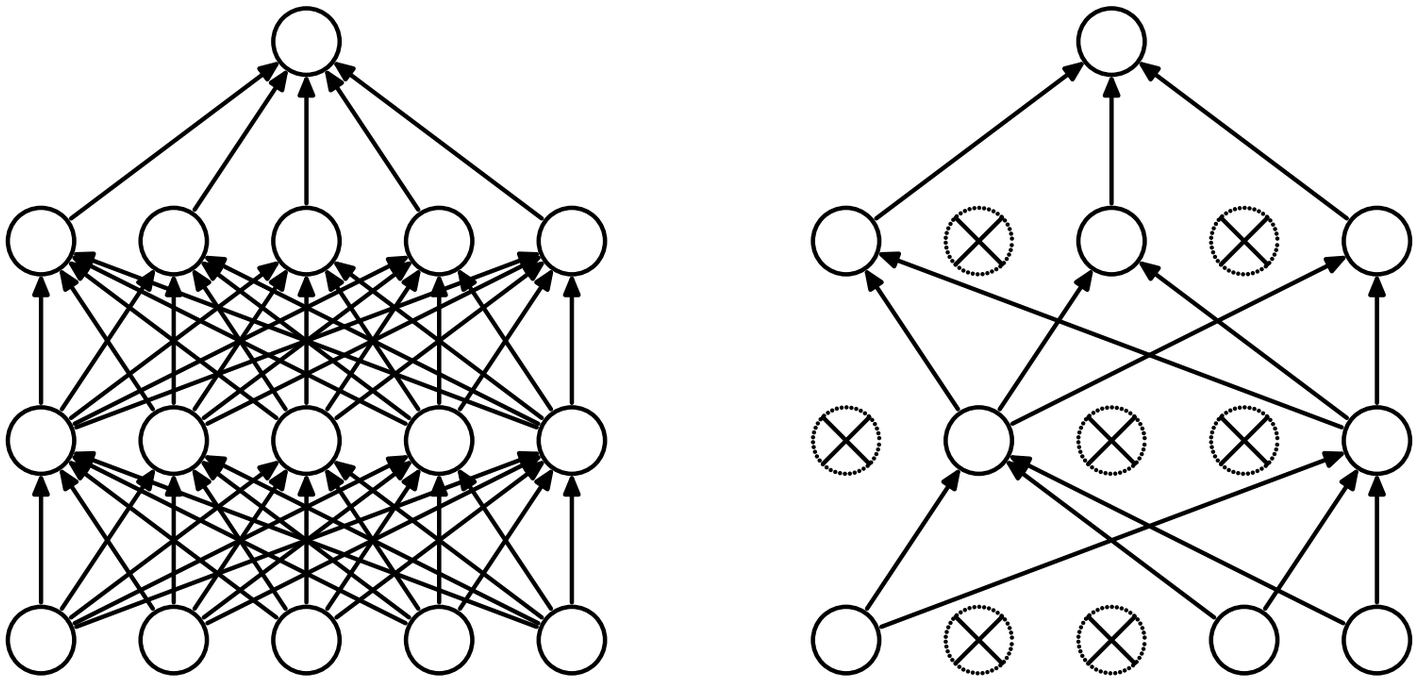
[Ioffe et al]

Intuition: build second-order behaviour into SGD by normalizing variables (zero-mean, identity covariance) before nonlinearity



Many (if not most) contemporary networks make use of this

Drop-out regularization



(a) Standard Neural Net

(b) After applying dropout.

Intuition: we should really train a family of models with different architectures and average their predictions (c.f. model averaging from machine learning)

Practical implementation: learn a single “superset” architecture that randomly removes nodes (by randomly zero’ing out activations) during gradient updates

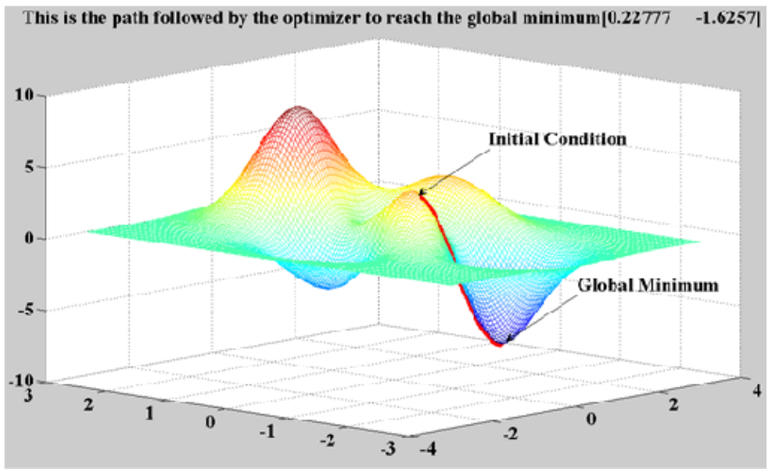
Bottom line: optimization matters!

Seems to be the limiting factor in performance right now

… so let's dig into the gritty details

$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

$$w := w - step * (w + \sum_i (f_w(x_i) - y_i) \frac{\partial f_w(x_i)}{\partial w})$$



Outline

- Motivation
- Popular networks
- Optimization
  - Backprop
  - Extensions: multiscale DAGs, recurrence, LSTMs
- Why does it work so well?