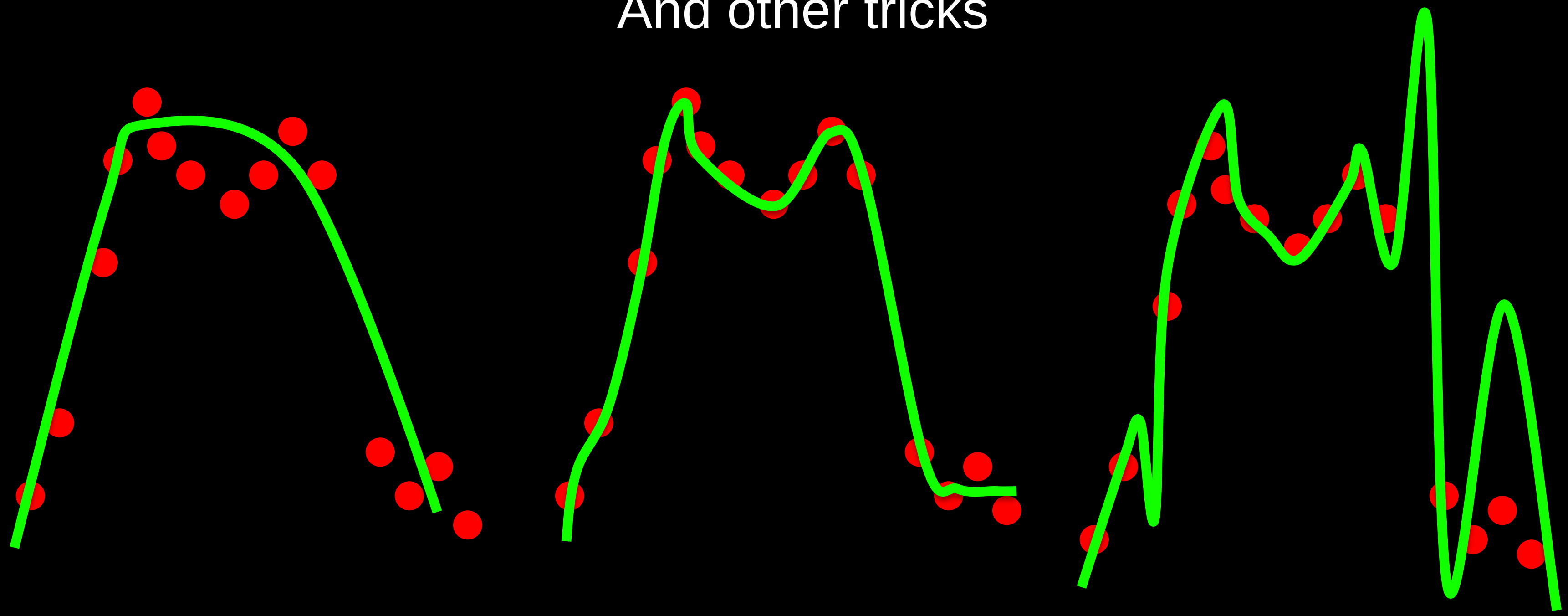
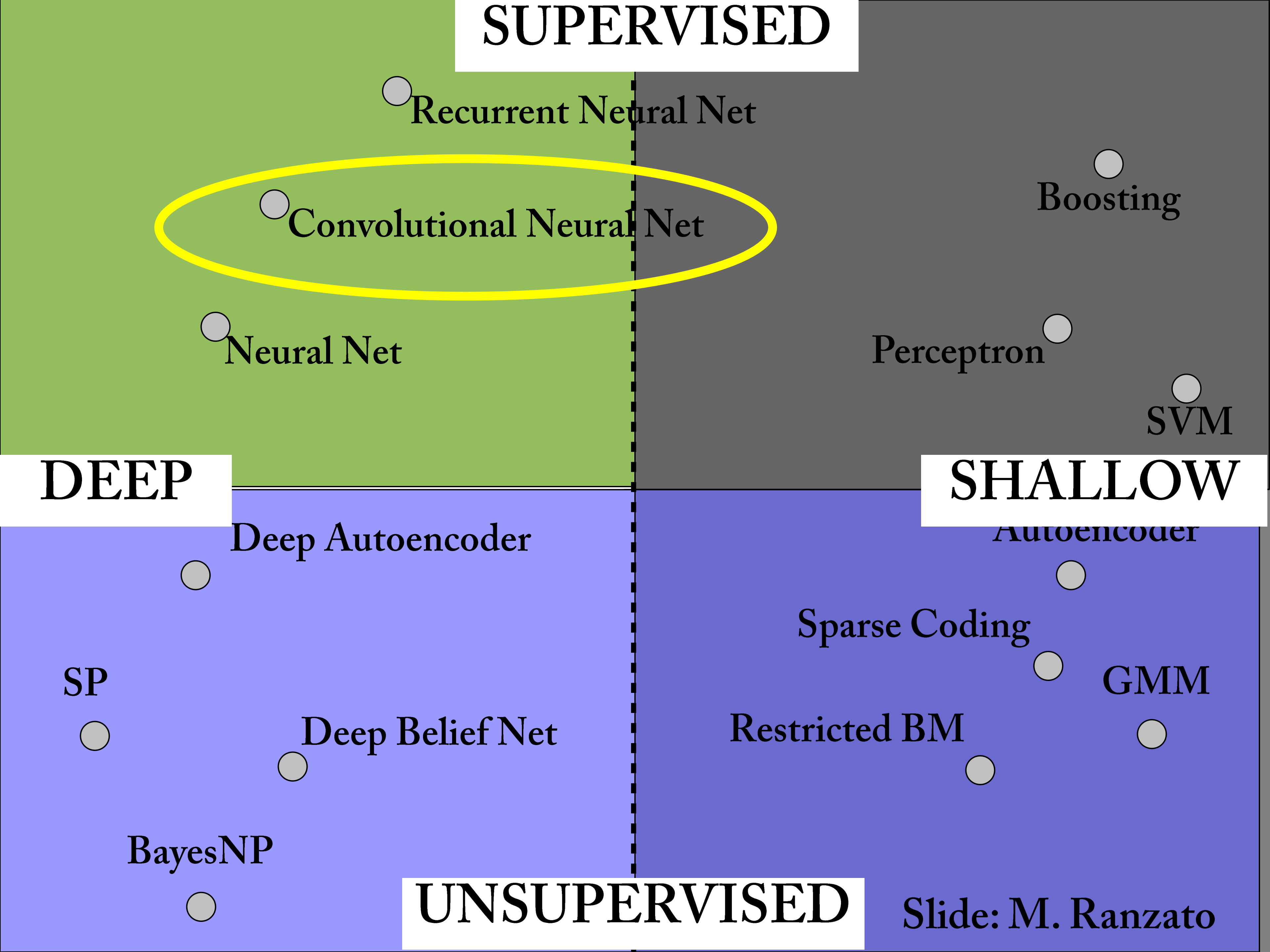
Introduction to Deep Learning

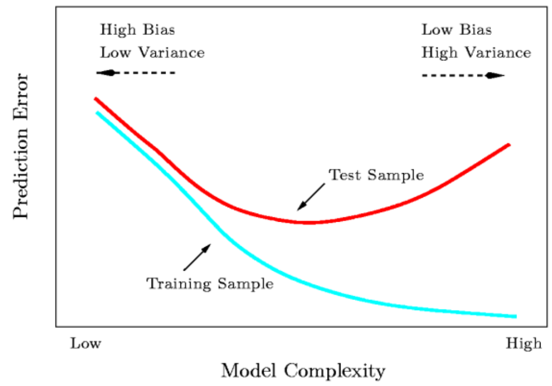
Convolutional Networks, Dropout
And other tricks

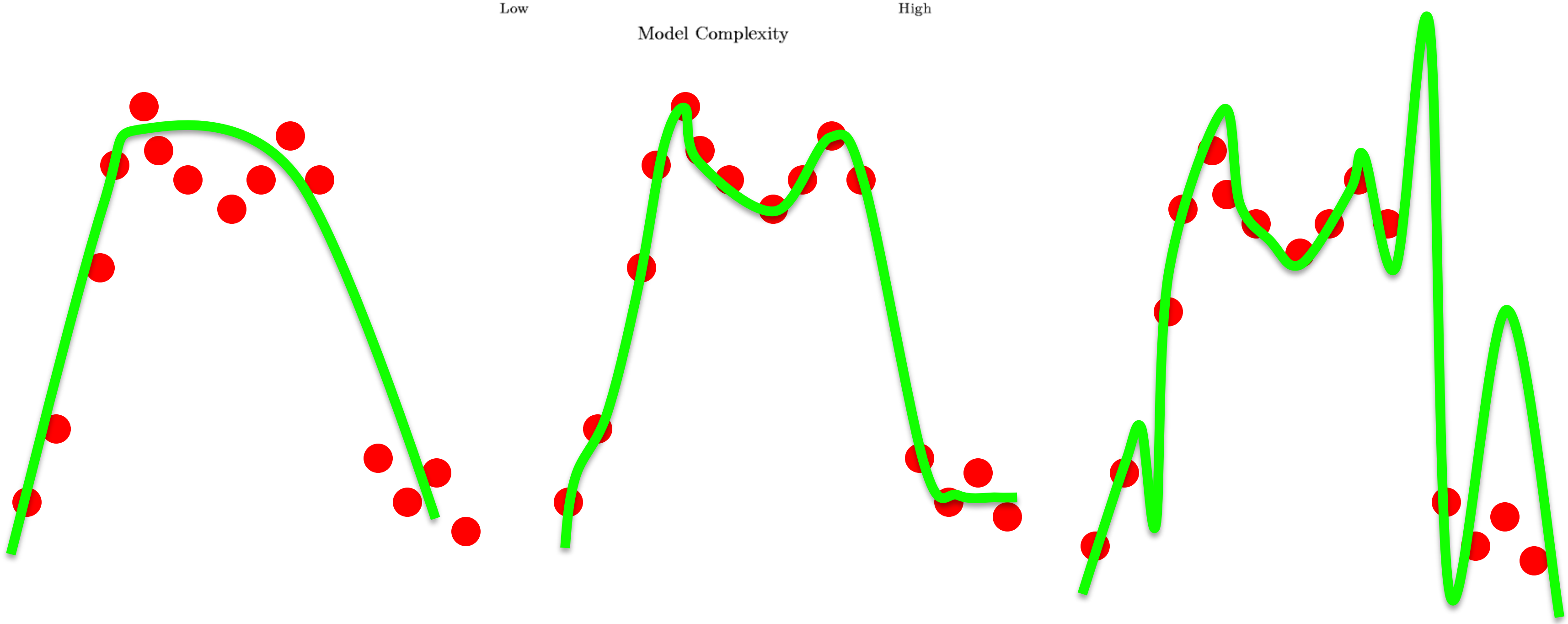


Dimitris Samaras
Stony Brook University


SUPERVISED
Recurrent Neural Net
Convolutional Neural Net
Boosting
Neural Net
Perceptron
SVM
DEEP
SHALLOW
Deep Autoencoder
Autoencoder
Sparse Coding
SP
GMM
Deep Belief Net
Restricted BM
BayesNP
UNSUPERVISED
Slide: M. Ranzato

MLCV, Lecture 1: Regularization problem

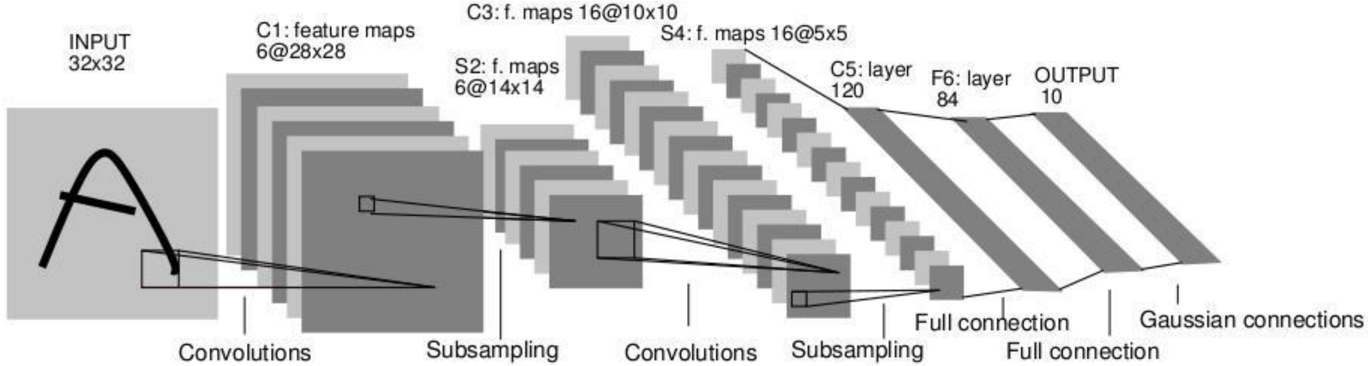




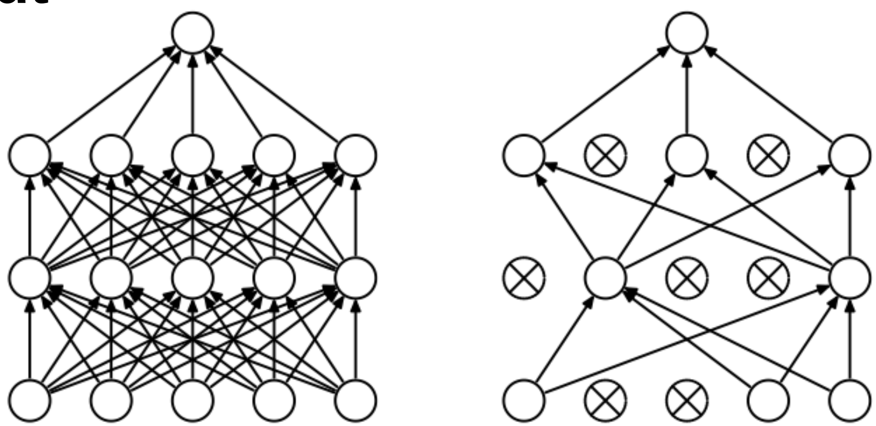
underfitting **just right** **overfitting**

Today: two regularization methods

Convolutional Networks



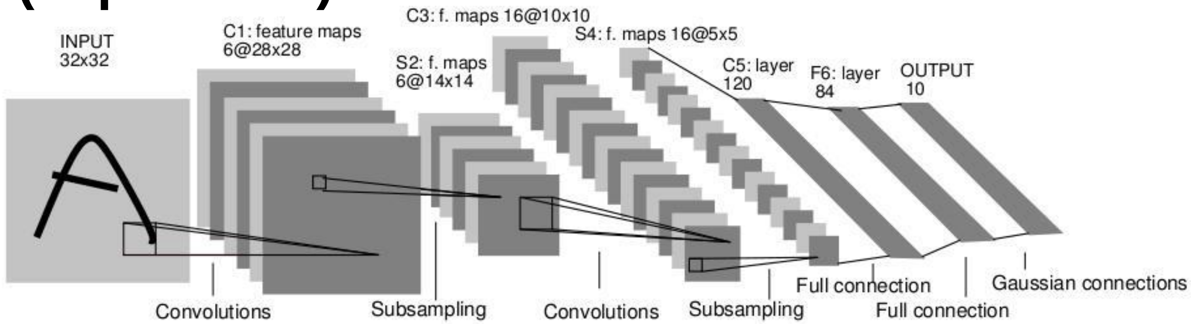
Dropout



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

Convolutional Networks

Discriminative (supervised)



Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541-551, Winter 1989

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Generative (unsupervised)

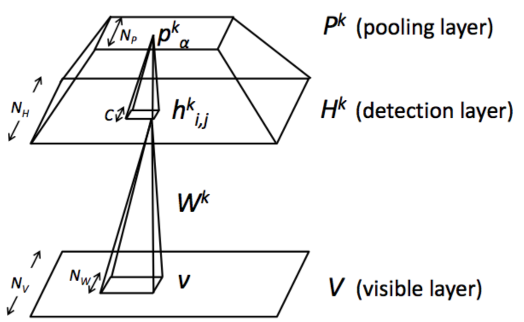


Figure 1. Convolutional RBM with probabilistic max-pooling. For simplicity, only group k of the detection layer and the pooing layer are shown. The basic CRBM corresponds to a simplified structure with only visible layer and detection (hidden) layer. See text for details.

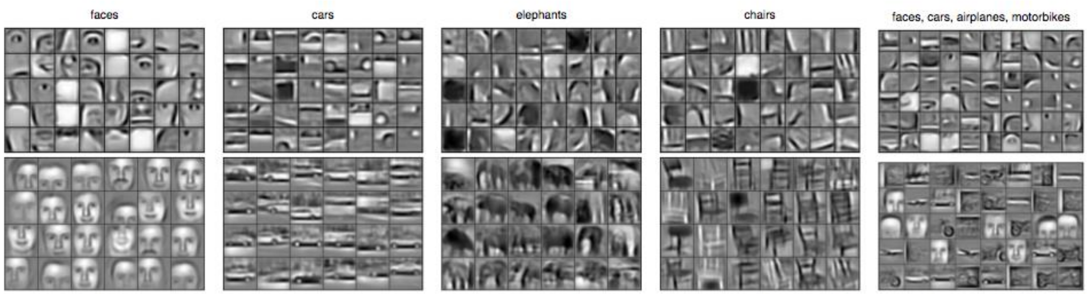


Figure 3. Columns 1-4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, ICML 2010

Minimal Convolutional Network

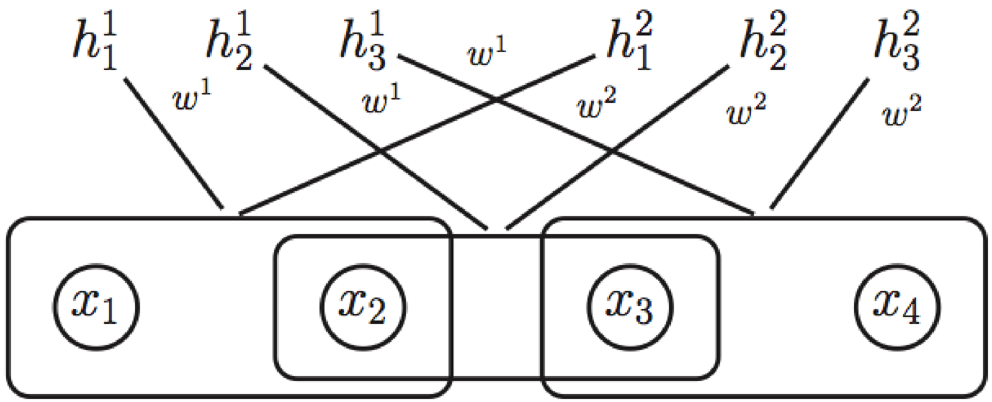


Figure 28.7 A small 1d convolutional RBM with two groups of hidden units, each associated with a filter of size 2. h_1^1 and h_1^2 are two different "views" of the data in the first window, (x_1, x_2). The first view is computed using the filter $\mathbf{w}^1$, the second view using filter $\mathbf{w}^2$. Similarly, h_2^1 and h_2^2 are the views of the data in the second window, (x_2, x_3), computed using $\mathbf{w}^1$ and $\mathbf{w}^2$ respectively.

# of Parameters

RBM: 4 x 6

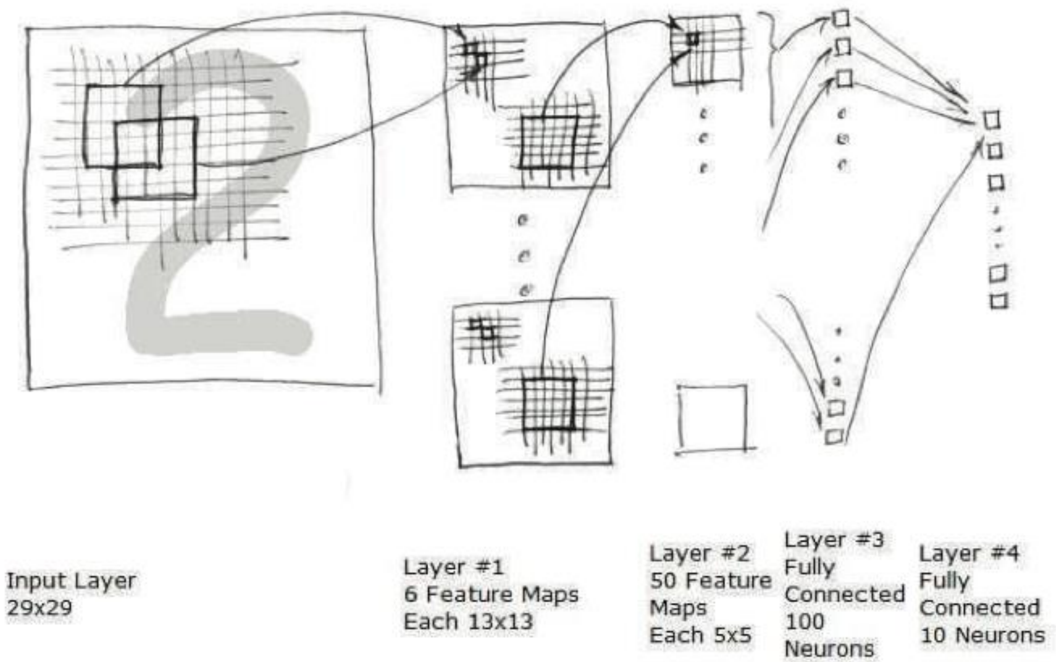
inputs x outputs

C-RBM: 2 x 2

block size x # of output types

K. Murphy, Probabilistic Machine Learning

Weights and connections in a CNN



(Simard et. al. 2003)

In layer 1, we have 6 feature maps each of which has size 13 × 13. Each hidden node in one of these feature maps is computed by convolving the image with a 5×5 weight matrix, adding a bias, and passing the result through some form of nonlinearity.
There are 13×13×6 = 1014 neurons and (5×5+1)×6 = 156 weights.
In layer 2, we have 50 feature maps, each of which is obtained by convolving each feature map in layer 1 with a 5 × 5 weight matrix, adding them up, adding a bias, and passing through a nonlinearity.
There are 5 × 5 × 50 = 1250 neurons, (5 × 5 + 1) × 6 × 50 = 7800 weights, and 1250 × 26 = 32, 500 connections.
Layer 3 is fully connected to layer 2, and has 100 neurons and 100 × (1250 + 1) = 125, 100 weights.
Layer 4 is also fully connected, and has 10 neurons, and 10 × (100 + 1) = 1010 weights.

Total: 3,215 neurons, 134,066 adjustable weights, and 184,974 connections.

K. Murphy, Probabilistic Machine Learning

Convolutional Neural Networks (aka ‘LeCun’ nets)

http://yann.lecun.com/exdb/lenet/index.html

Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]

- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 20

- But (until recently) less good at more complex datasets
  - E.g. Caltech-101/256 (few training examples)

Convolutional Neural Networks

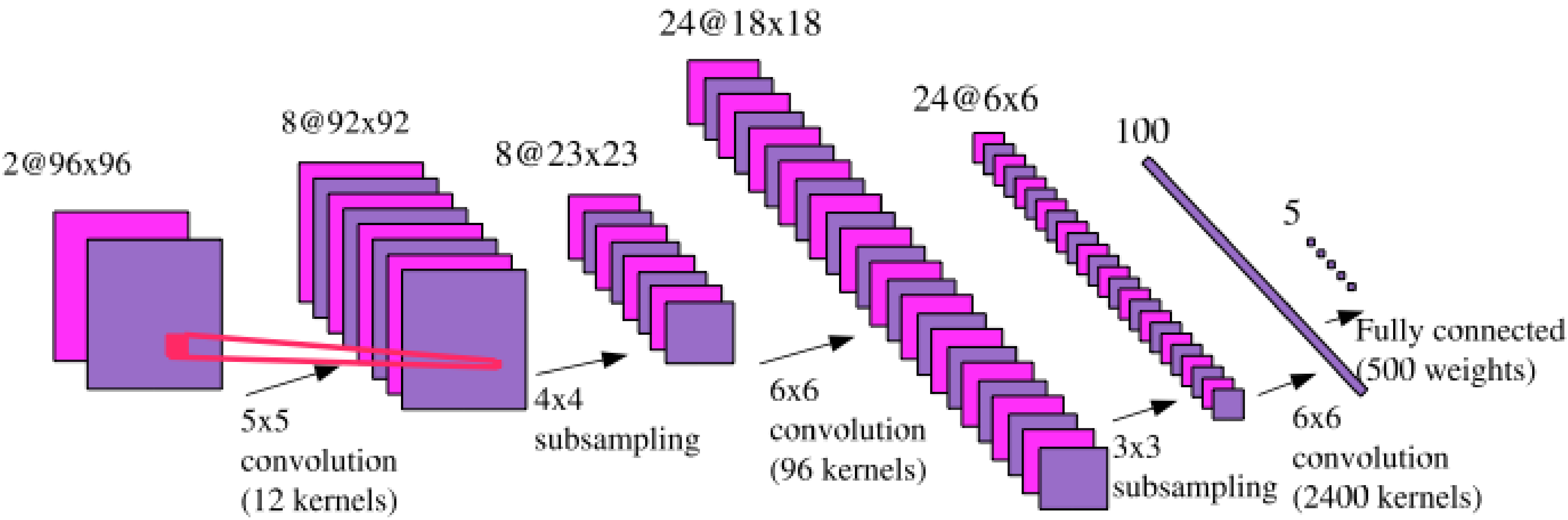
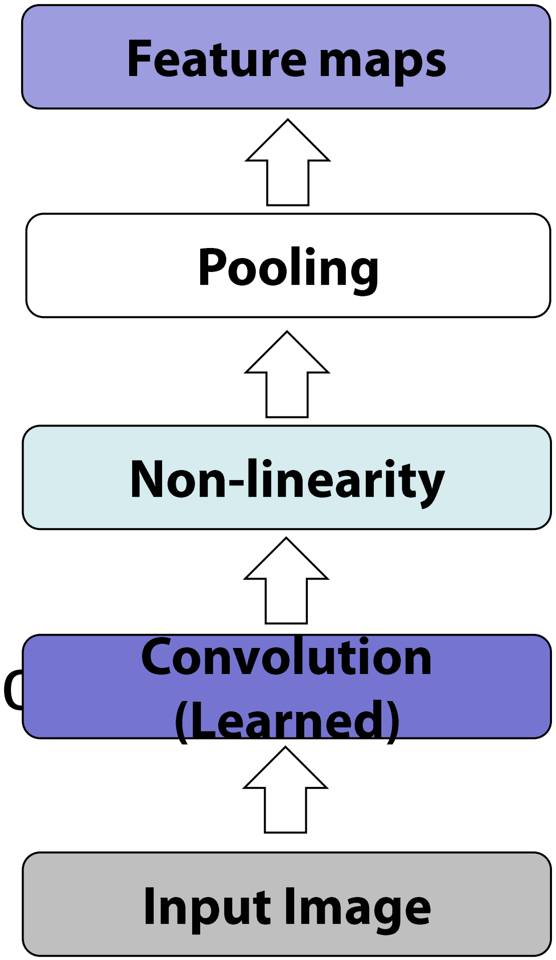


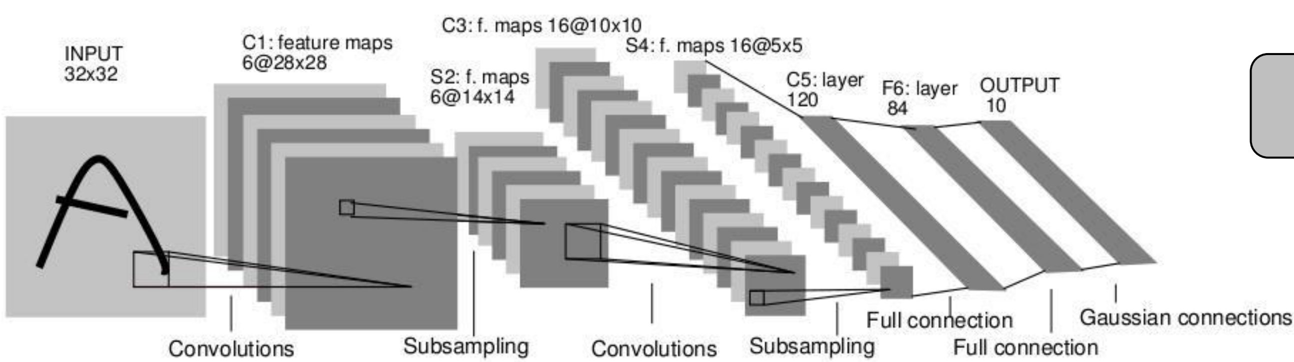
Figure 5: The architecture of the convolutional net used for the NORB experiments. The input is an image pair, the system extracts 8 feature maps of size 92×92, 8 maps of 23×23, 24 maps of 18×18, 24 maps of 6×6, and 100 dimensional feature vector. The feature vector is then transformed into a 5-dimensional vector in the last layer to compute the distance with target vectors.

Bengio & Le Cun

Recap of Convnets

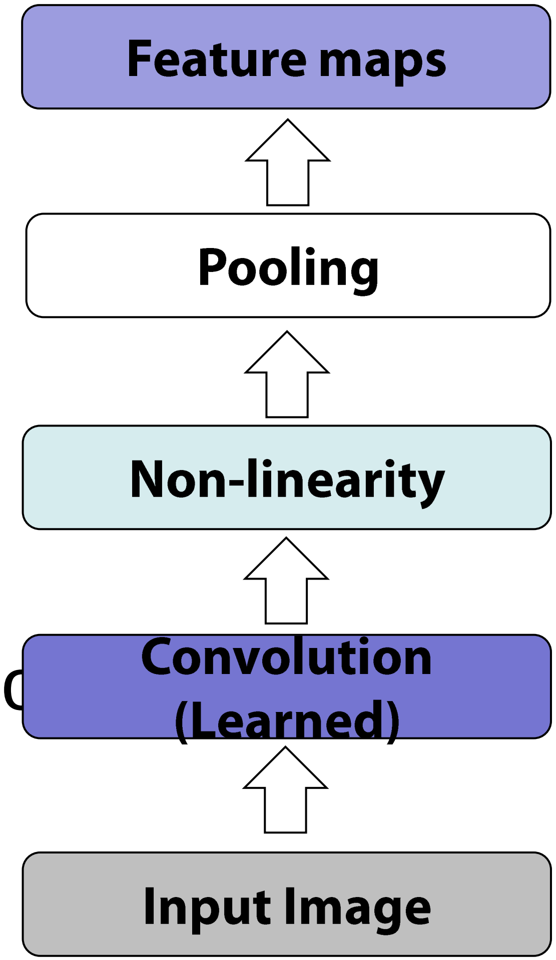
- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification erro

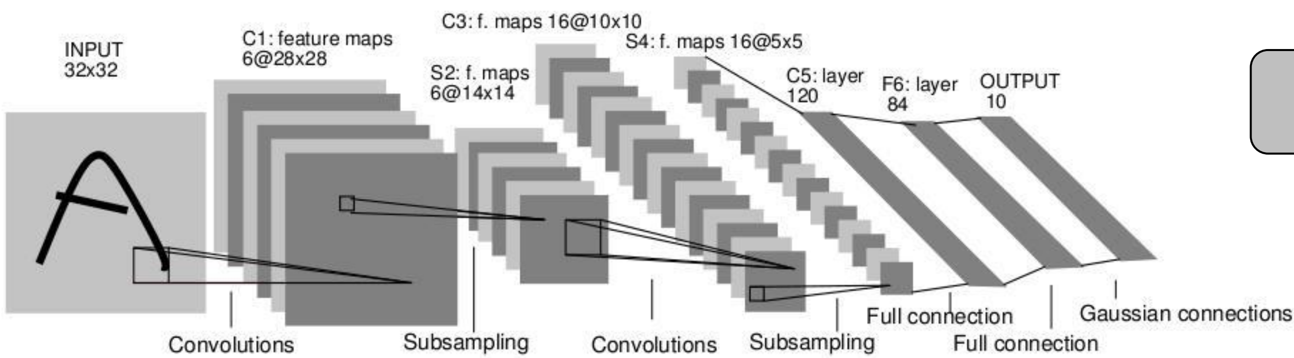




Recap of Convnets

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification errc





Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]

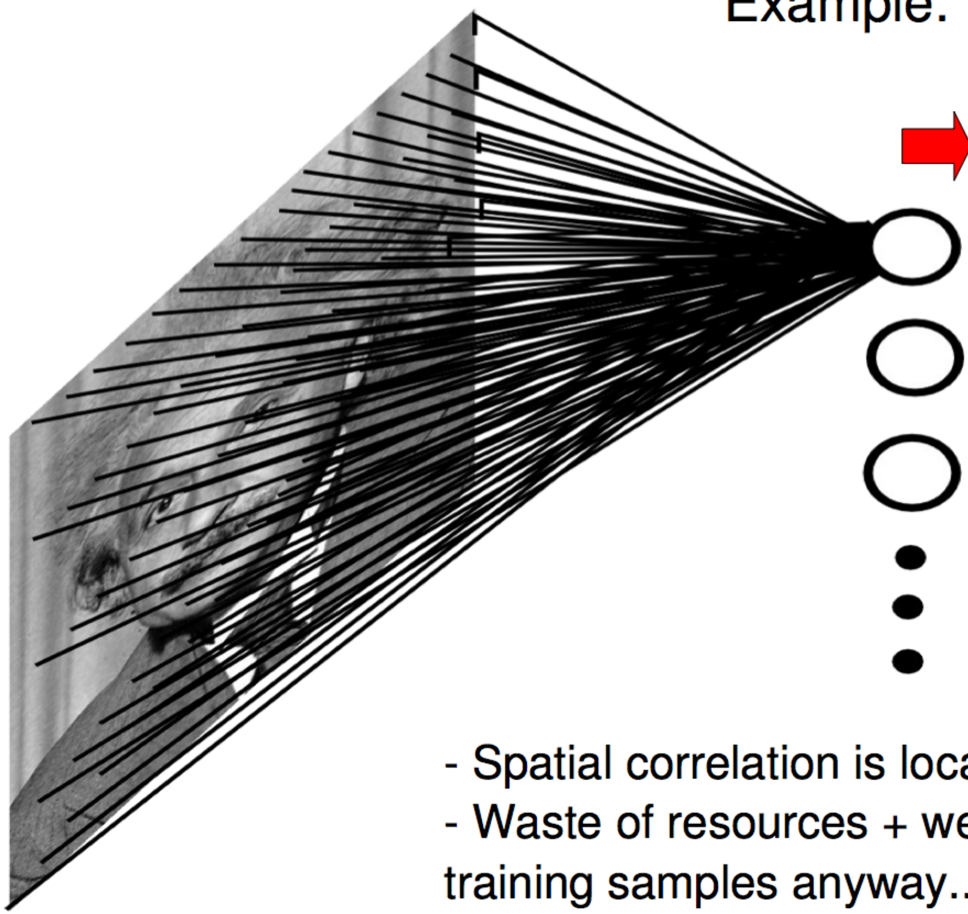
- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 20

- But (until recently) less good at more complex datasets
  - E.g. Caltech-101/256 (few training examples)

Network connectivity

Fully Connected Layer

Example: 200x200 image
40K hidden units
→ **~2B parameters**!!!



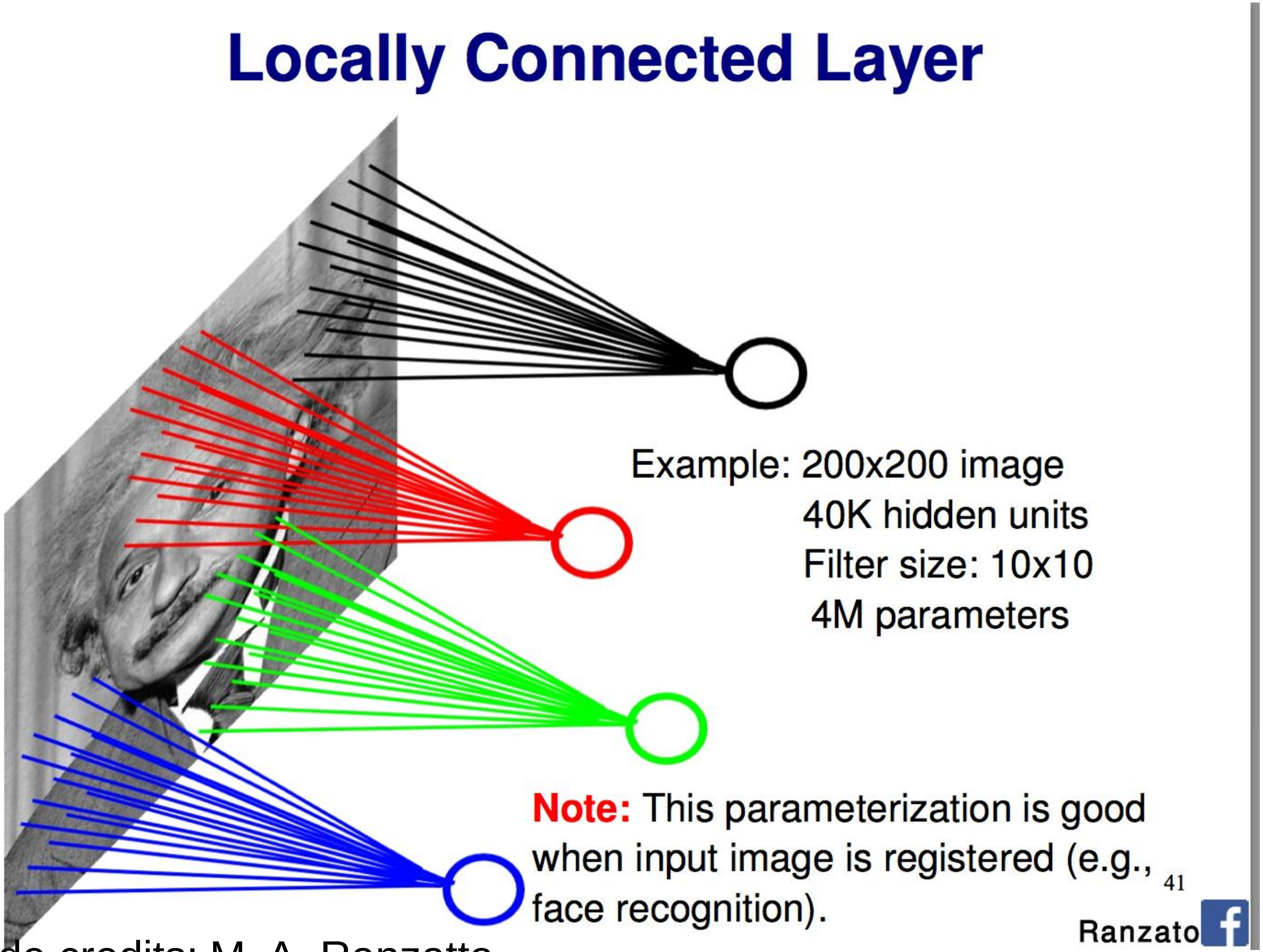
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

40

Ranzato

Slide credits: M. A. Ranzatto

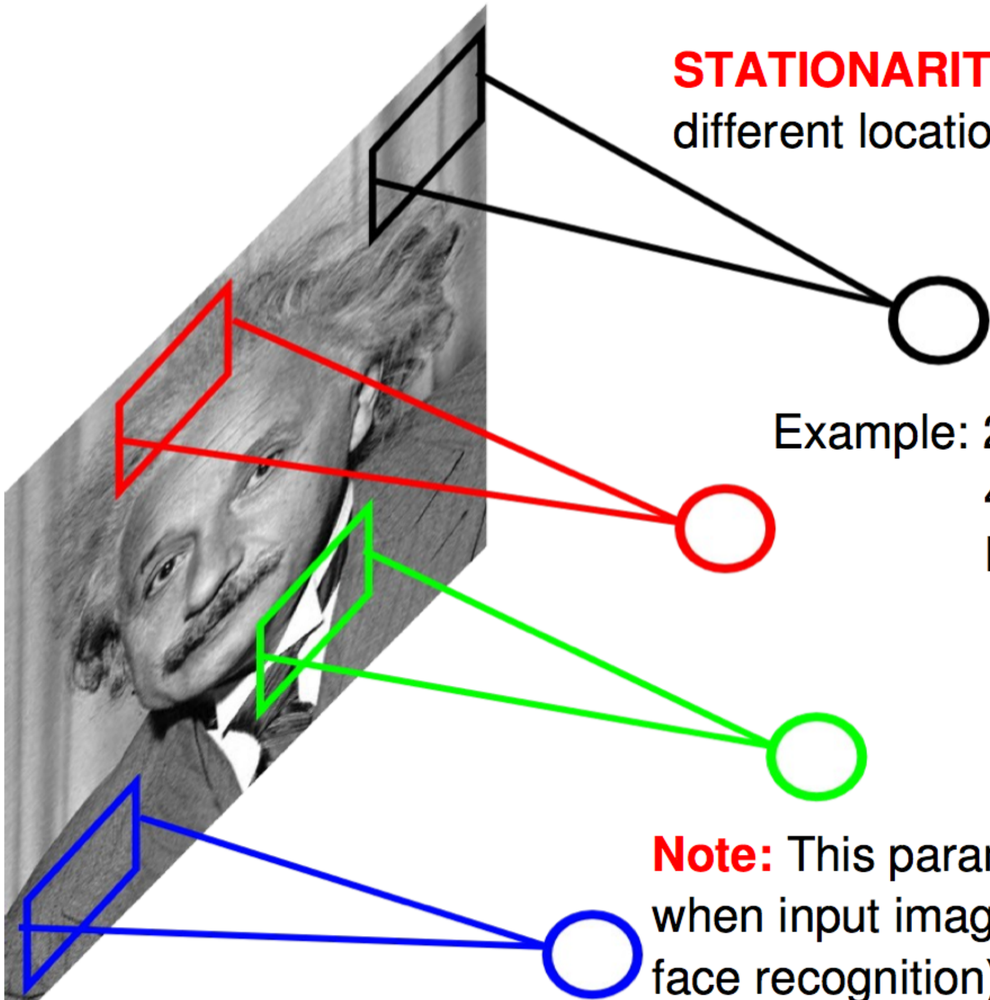
Network connectivity



Slide credits: M. A. Ranzatto

Network connectivity

Locally Connected Layer



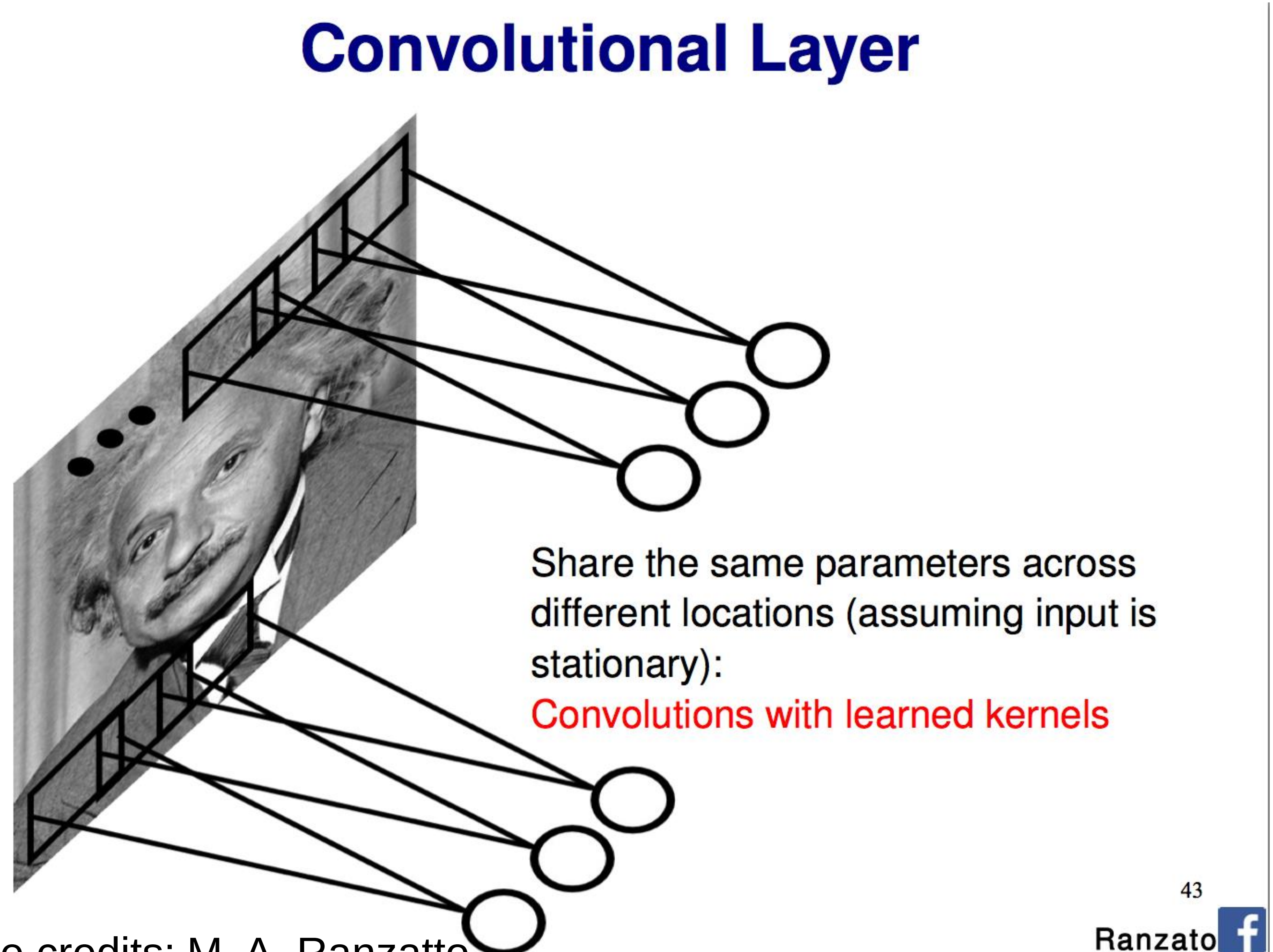
STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Slide credits: M. A. Ranzatto

Network connectivity



Slide credits: M. A. Ranzatto

Network connectivity

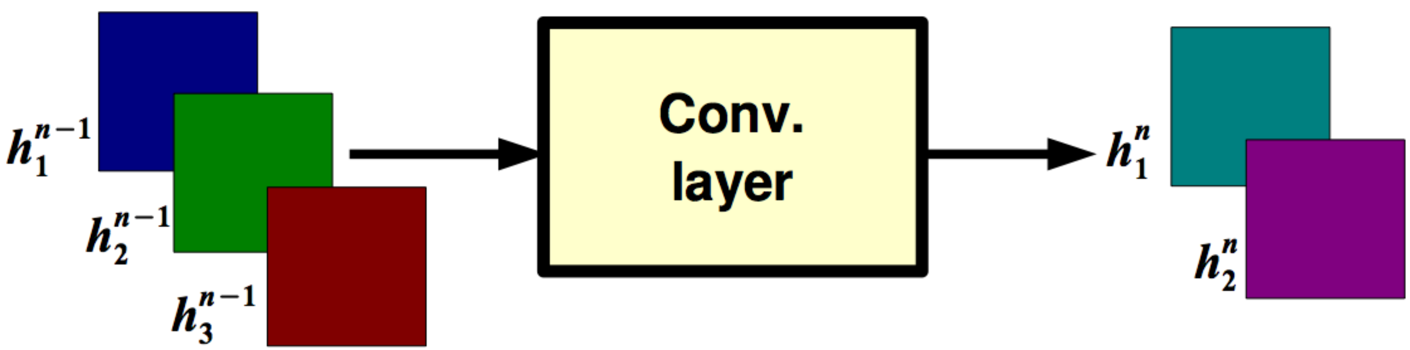
Convolutional Layer

$$h_j^n = max\left(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n\right)$$

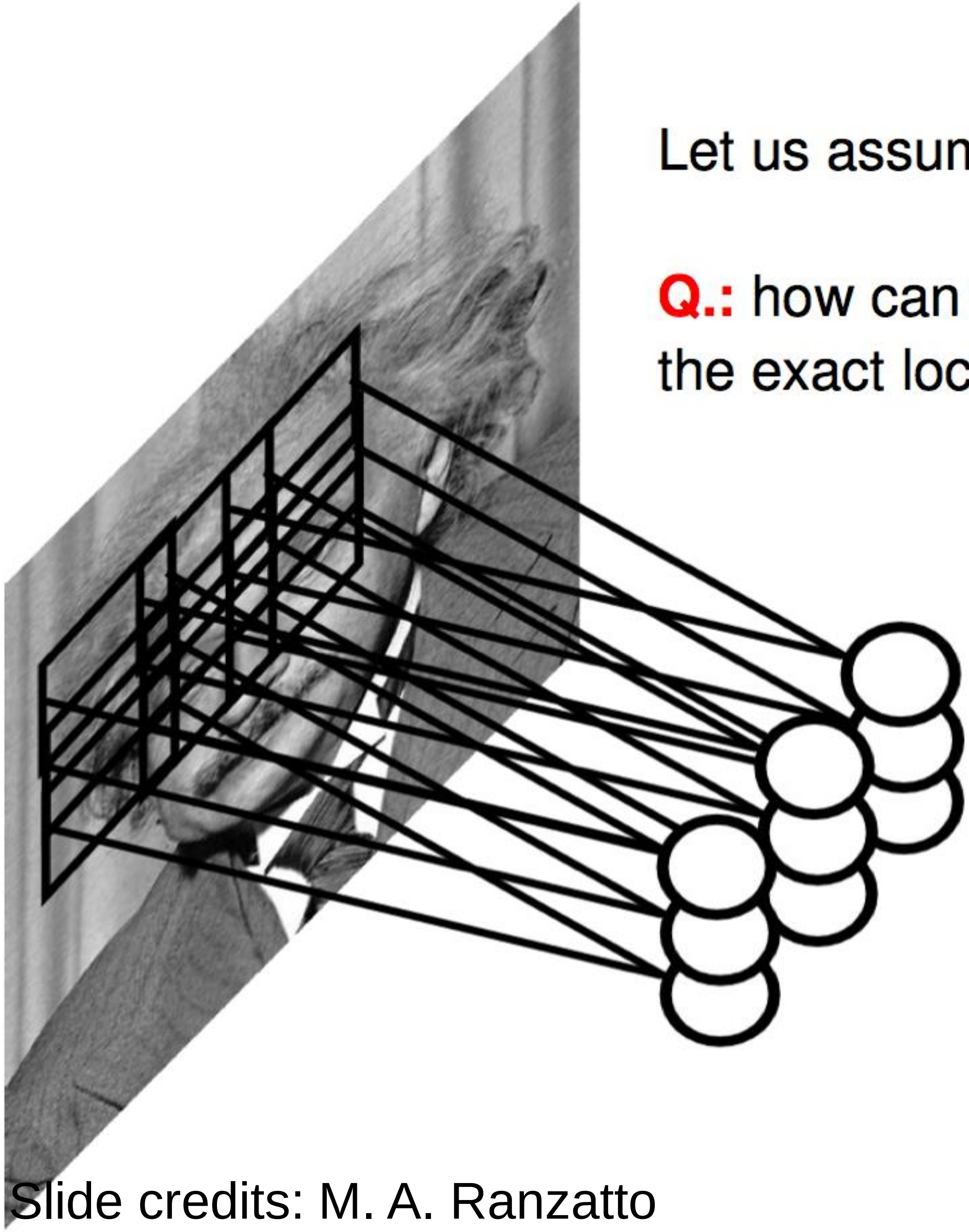
output feature map

input feature map

kernel



Pooling Layer



Let us assume filter is an "eye" detector.

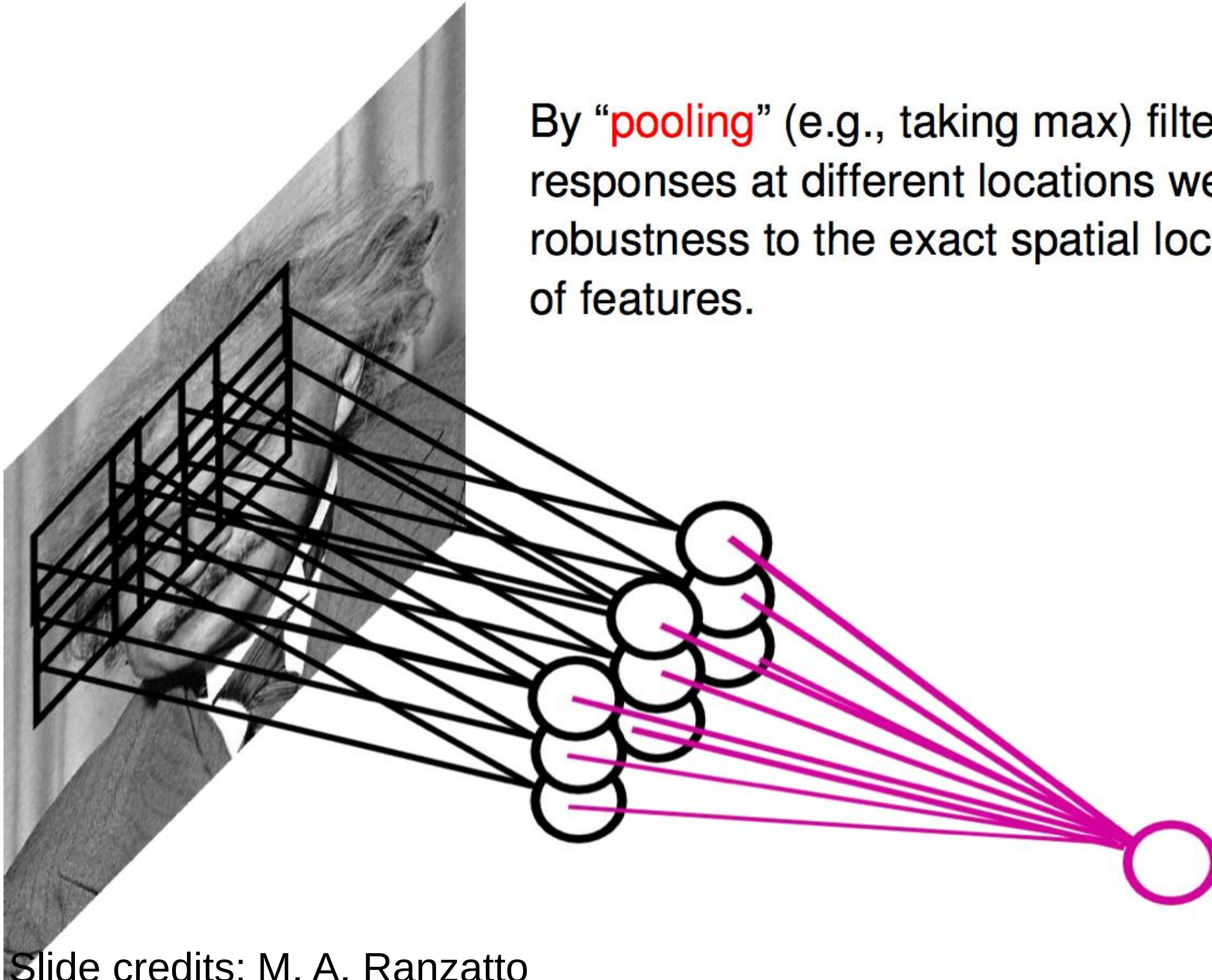
Q.: how can we make the detection robust to the exact location of the eye?

Slide credits: M. A. Ranzatto

Ranzato

Pooling Layer

By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.



Slide credits: M. A. Ranzatto

Ranzato

Pooling Layer: Examples

Max-pooling:

$$h_j^n(x,y)=max_{\bar{x}\in N(x),\,\bar{y}\in N(y)}\,h_j^{n-1}(\bar{x},\bar{y})$$

Average-pooling:

$$h_j^n(x,y)=1/K\sum_{\bar{x}\in N(x),\,\bar{y}\in N(y)}h_j^{n-1}(\bar{x},\bar{y})$$

L2-pooling:

$$h_j^n(x,y)=\sqrt{\sum_{\bar{x}\in N(x),\,\bar{y}\in N(y)}h_j^{n-1}(\bar{x},\bar{y})^2}$$

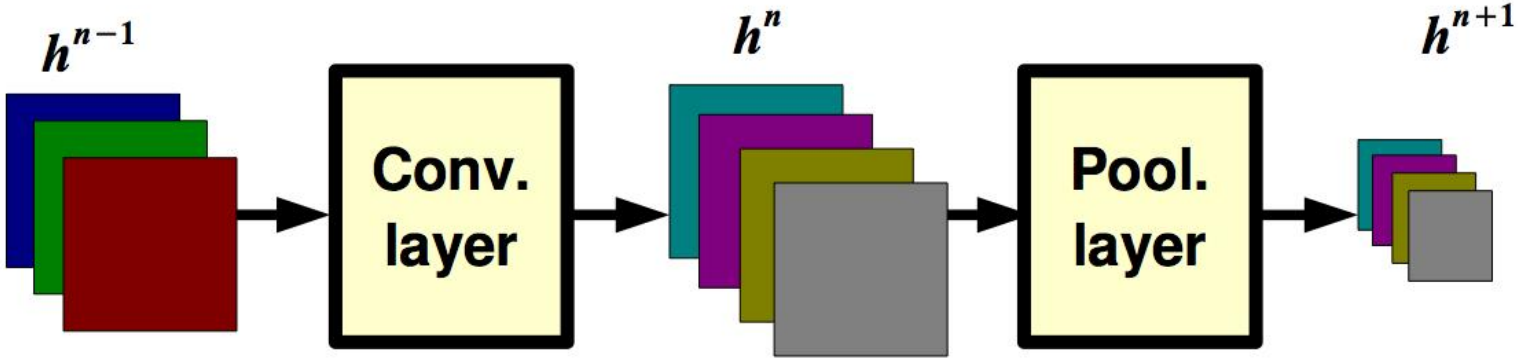
L2-pooling over features:

$$h_j^n(x,y)=\sqrt{\sum_{k\in N(j)}h_k^{n-1}(x,y)^2}$$

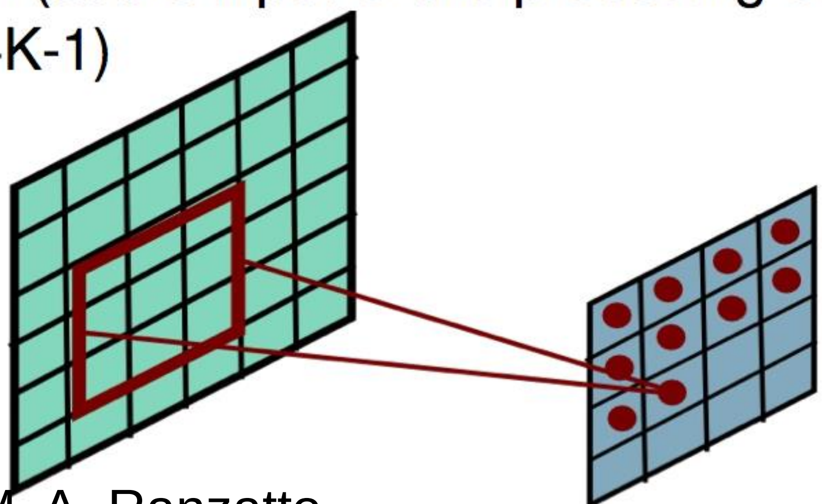
Slide credits: M. A. Ranzatto

Ranzato

Pooling Layer: Receptive Field Size



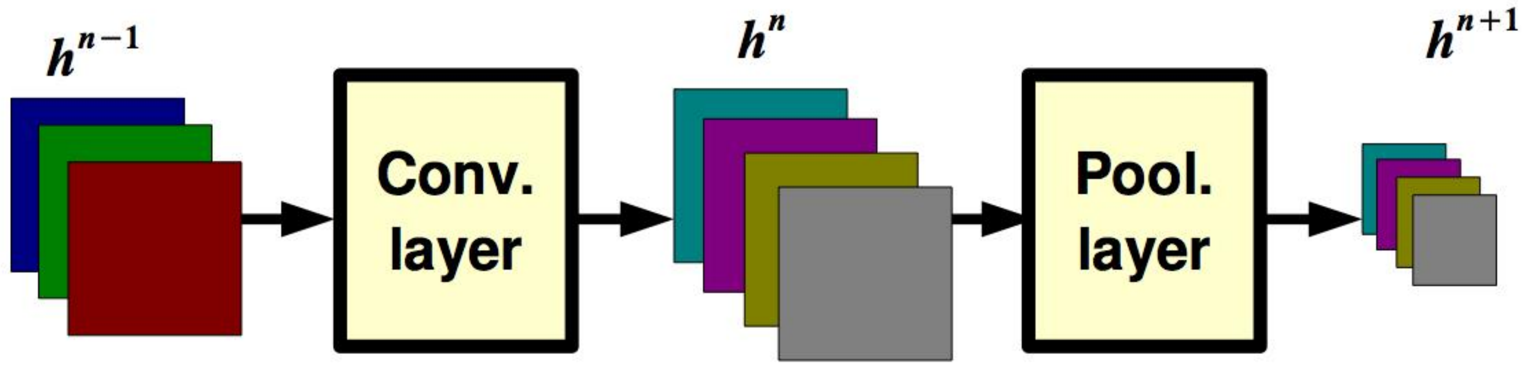
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



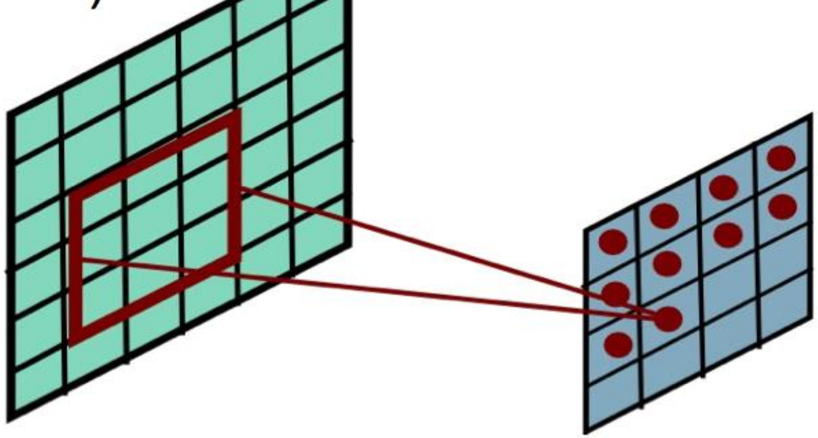
Ranzato

Slide credits: M. A. Ranzatto

Pooling Layer: Receptive Field Size

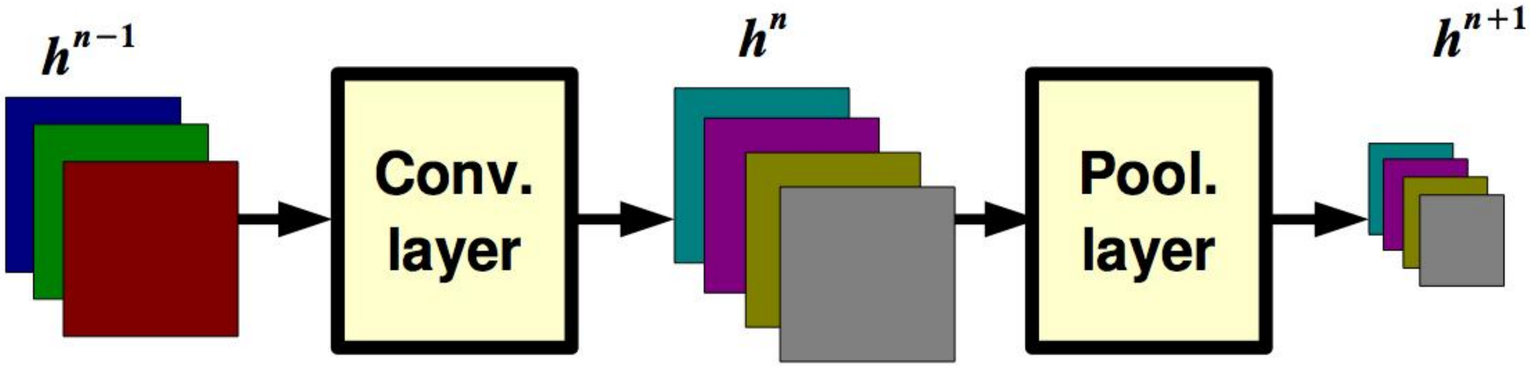


If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)

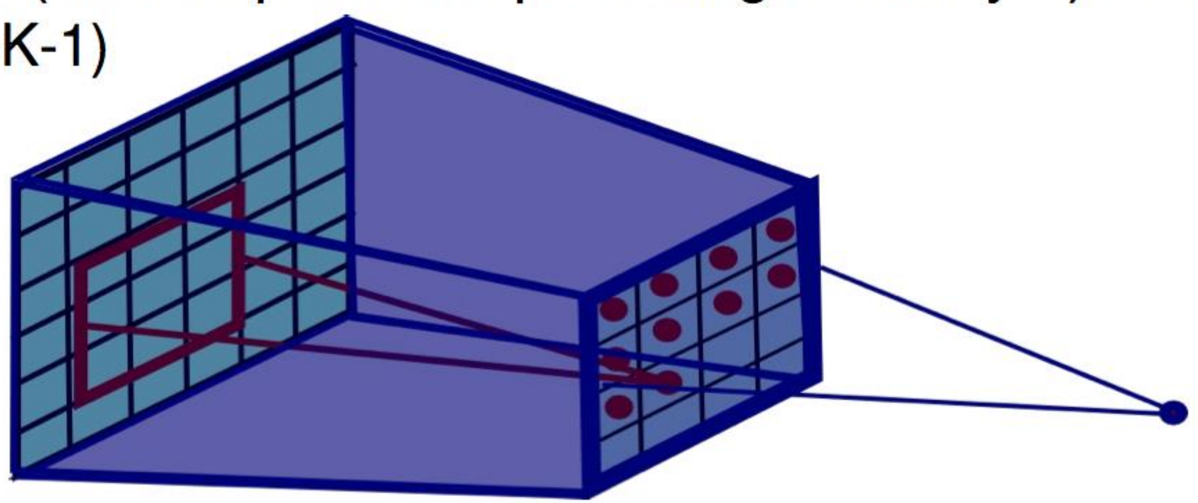


Ranzato

Pooling Layer: Receptive Field Size



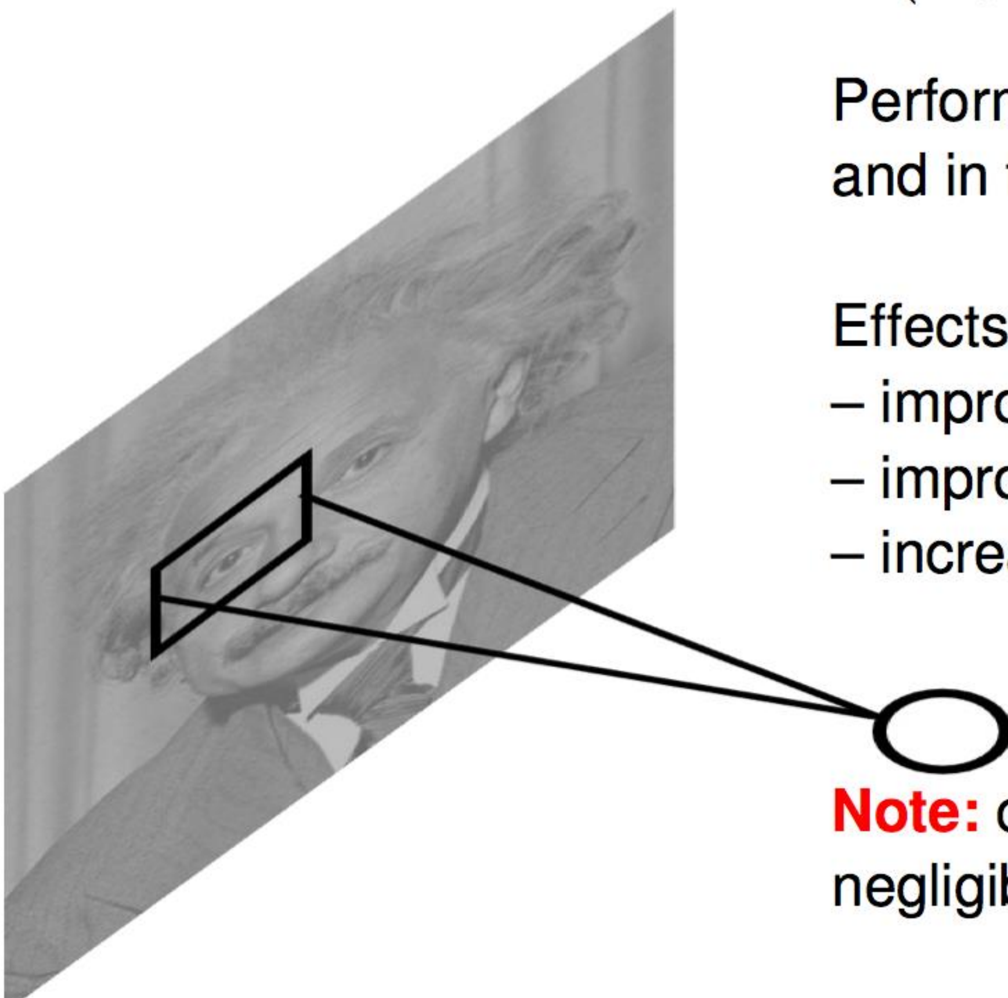
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



Ranzato

Local Contrast Normalization

$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{max(\epsilon, \sigma^i(N(x,y)))}$$



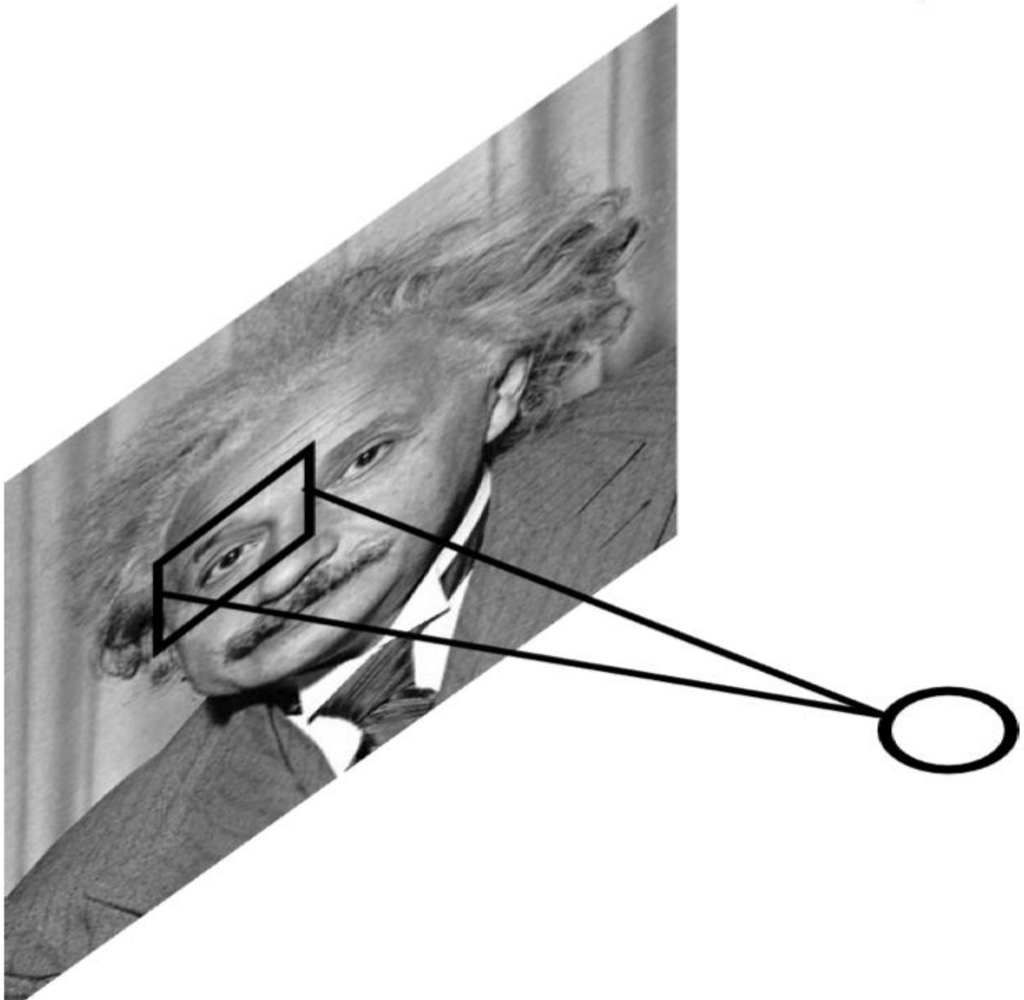
Performed also across features and in the higher layers..

Effects:
– improves invariance
– improves optimization
– increases sparsity

Note: computational cost is negligible w.r.t. conv. layer.

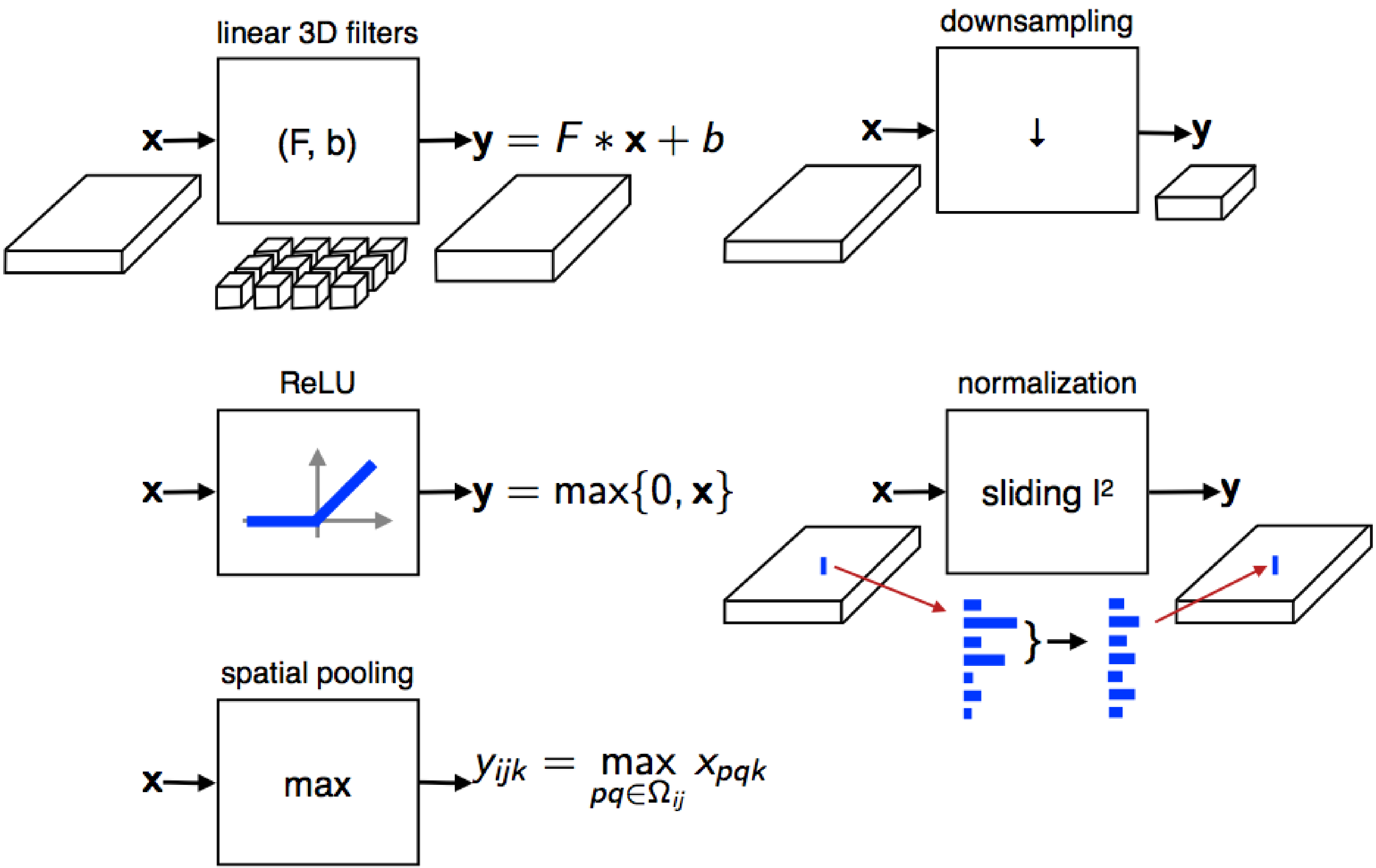
Local Contrast Normalization

$$h^{i+1}(x,y) = \frac{h^i(x,y) - m^i(N(x,y))}{max(\epsilon, \sigma^i(N(x,y)))}$$



Ranzato

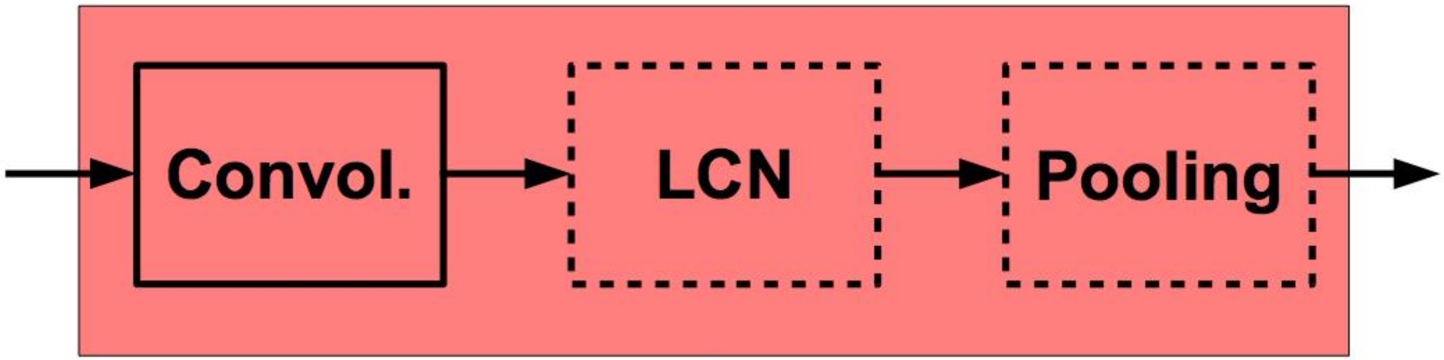
CNN components

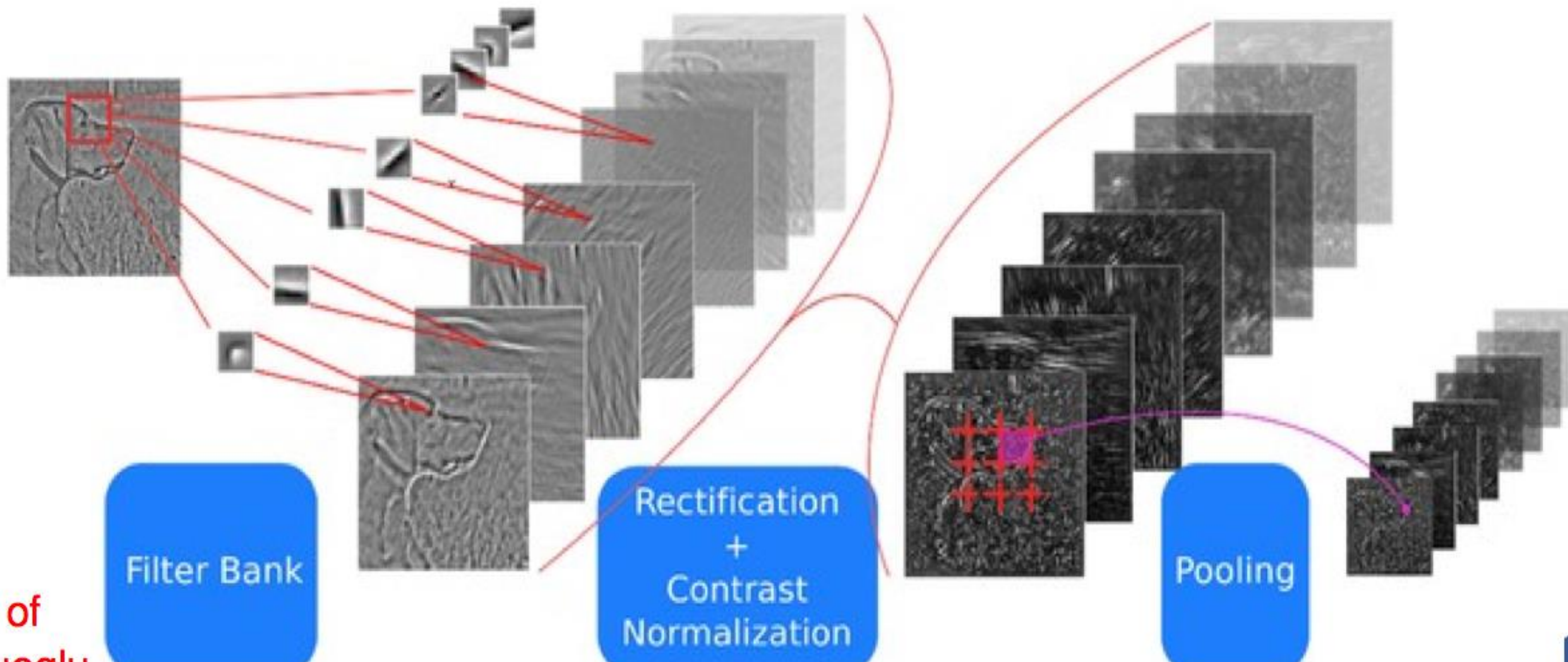


Slide credits: A. Vedaldi

ConvNets: Typical Stage

One stage (zoom)



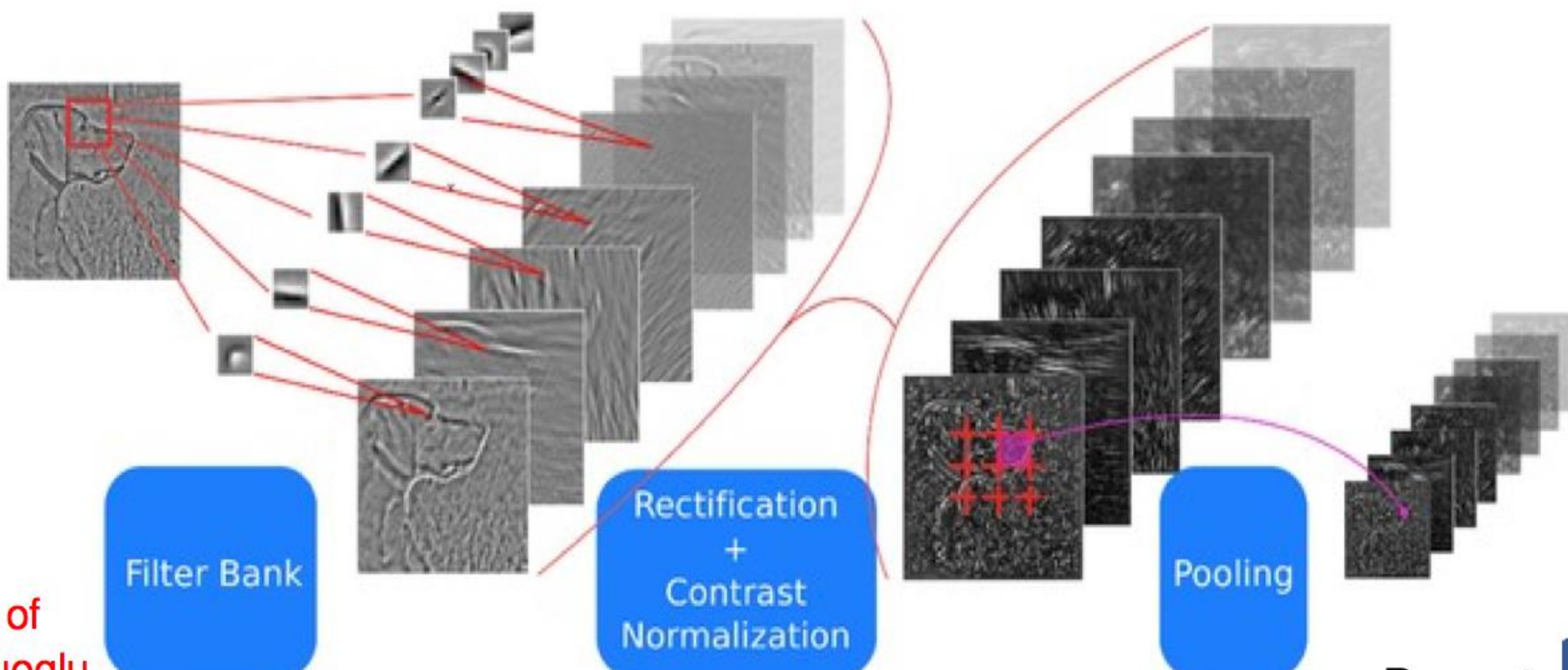


courtesy of
K. Kavukcuoglu

Ranzato

Note: after one stage the number of feature maps is usually increased (conv. layer) and the spatial resolution is usually decreased (stride in conv. and pooling layers). Receptive field gets bigger.

Reasons:
- gain invariance to spatial translation (pooling layer)
- increase specificity of features (approaching object specific units)



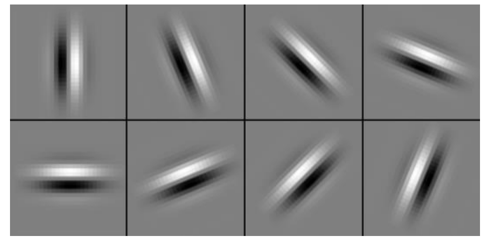
courtesy of
K. Kavukcuoglu

Ranzato

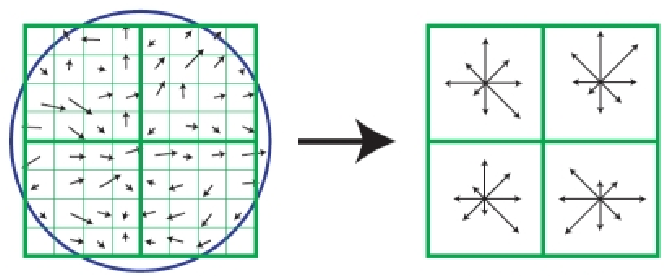
SIFT Descriptor

Image Pixels

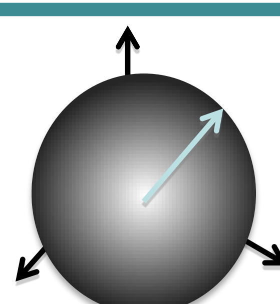
Apply Gabor filters



Spatial pool (Sum)



Normalize to unit length



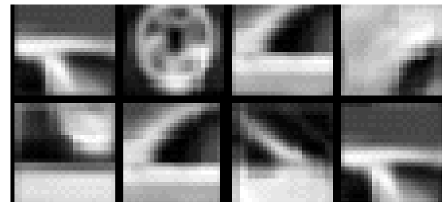
Feature Vector

Slide: R. Fergus

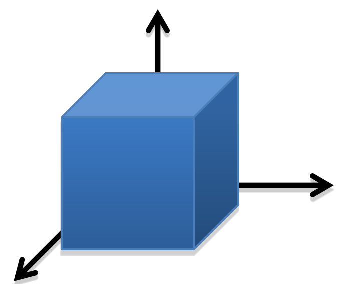
Spatial Pyramid Matching

SIFT Features

Filter with Visual Words



Max



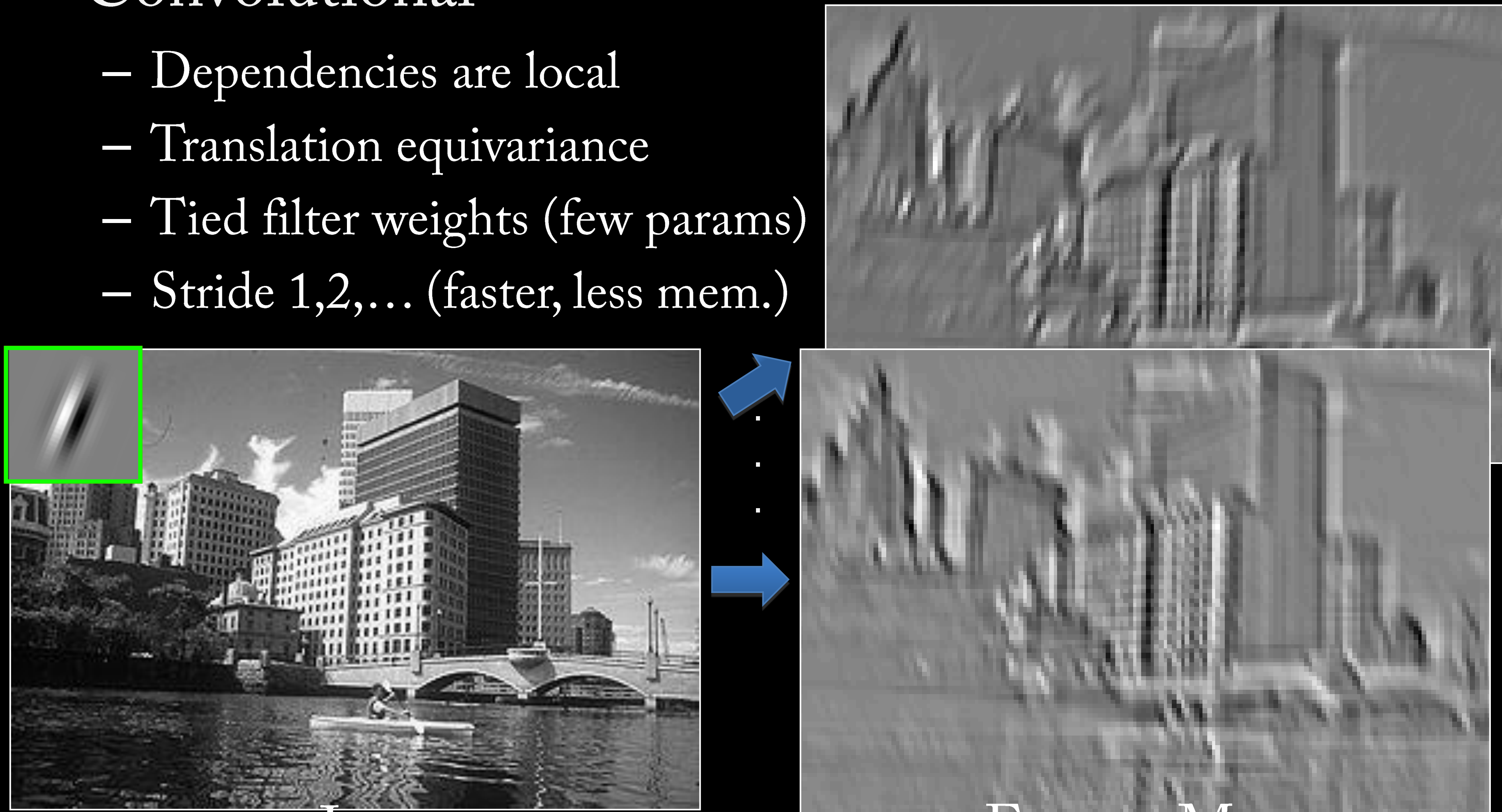
Multi-scale spatial pool (Sum)

Classifier

Slide: R. Fergus

Filtering

- Convolutional
  - Dependencies are local
  - Translation equivariance
  - Tied filter weights (few params)
  - Stride 1,2,… (faster, less mem.)



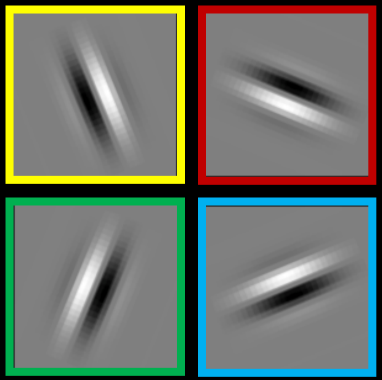
Input

Feature Map

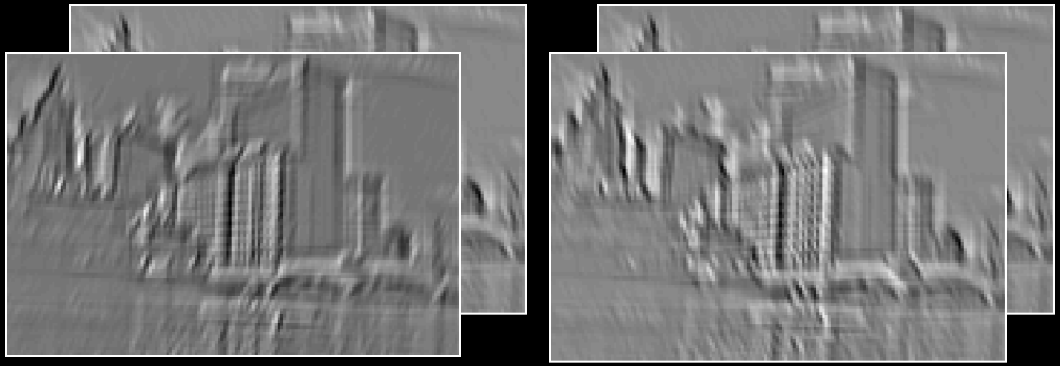
Filtering

- Tiled
  - Filters repeat every n
  - More filters than convolution for given # features

Input



Filters

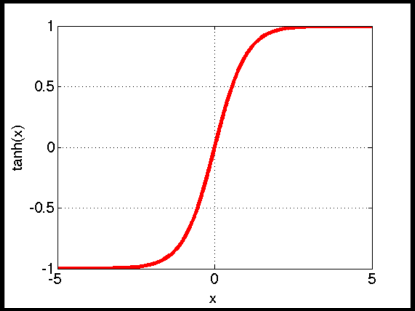


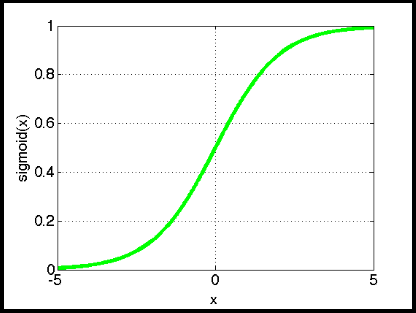
Feature maps

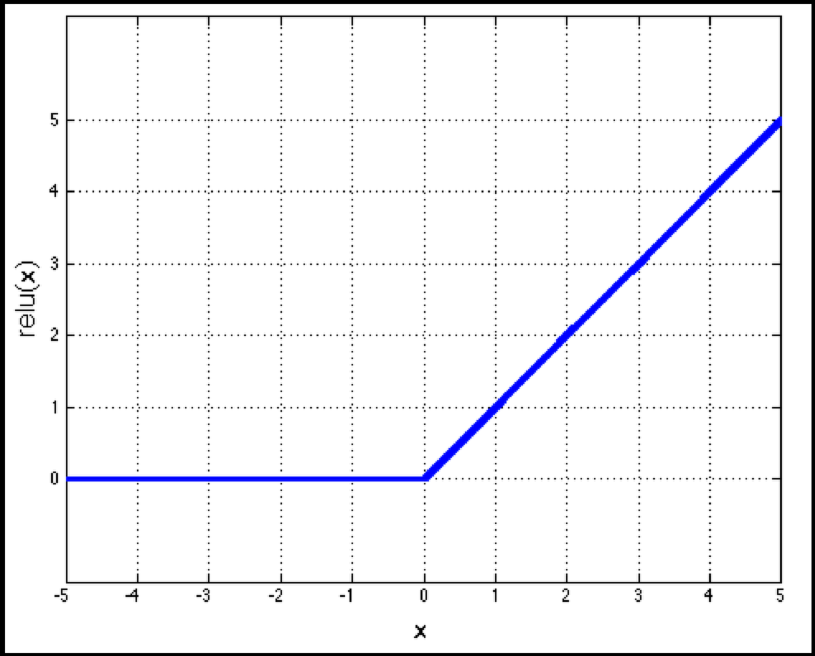
Non-Linearity

- Non-linearity
  - Per-feature independent
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear
    - Simplifies backprop
    - Makes learning faster
    - Avoids saturation issues

    → Preferred option

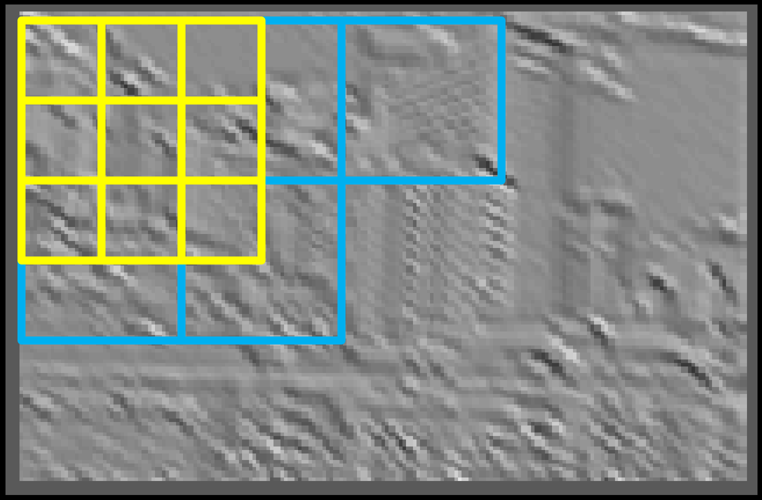




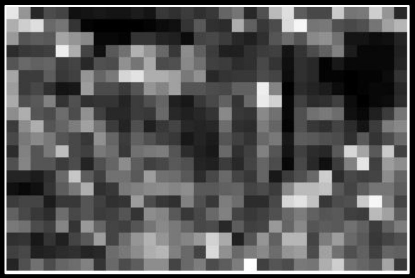


Pooling

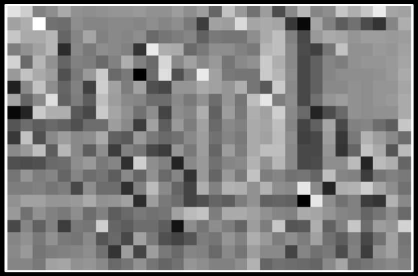
- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
  - Boureau et al. ICML'10 for theoretical analysis



Max



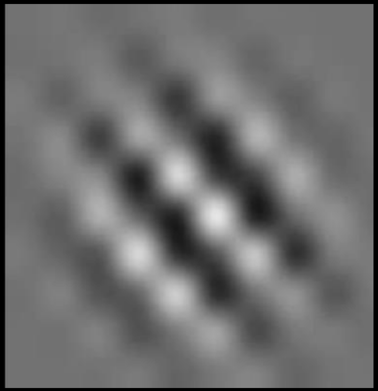
Sum

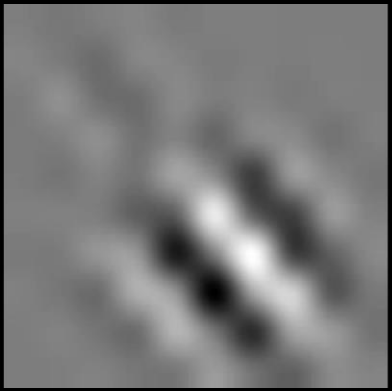


Role of Pooling

- Spatial pooling
  - Invariance to small transformations
  - Larger receptive fields (see more of input)

Visualization technique from [Le et al. NIPS'10]:





Zeiler, Fergus [arXiv 2013]

Videos from: http://ai.stanford.edu/~quocle/TCNNweb

Normalization

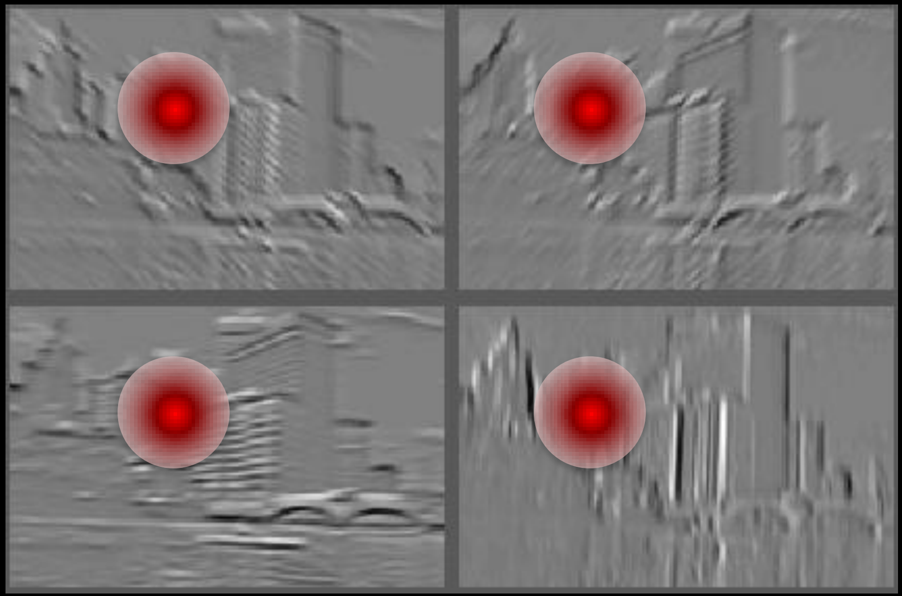
- Contrast normalization
  - See Divisive Normalization in Neuroscience

Input

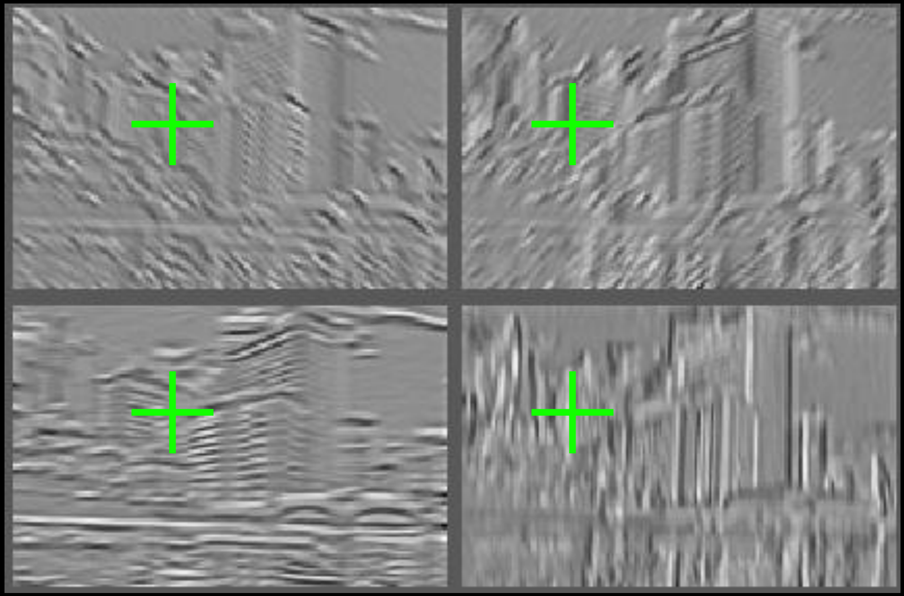
Filters

Normalization

- Contrast normalization (between/across feature maps)
  - Local mean = 0, local std. = 1, "Local" → 7x7 Gaussian
  - Equalizes the features maps



Feature Maps



Feature Maps
After Contrast Normalization

Role of Normalization

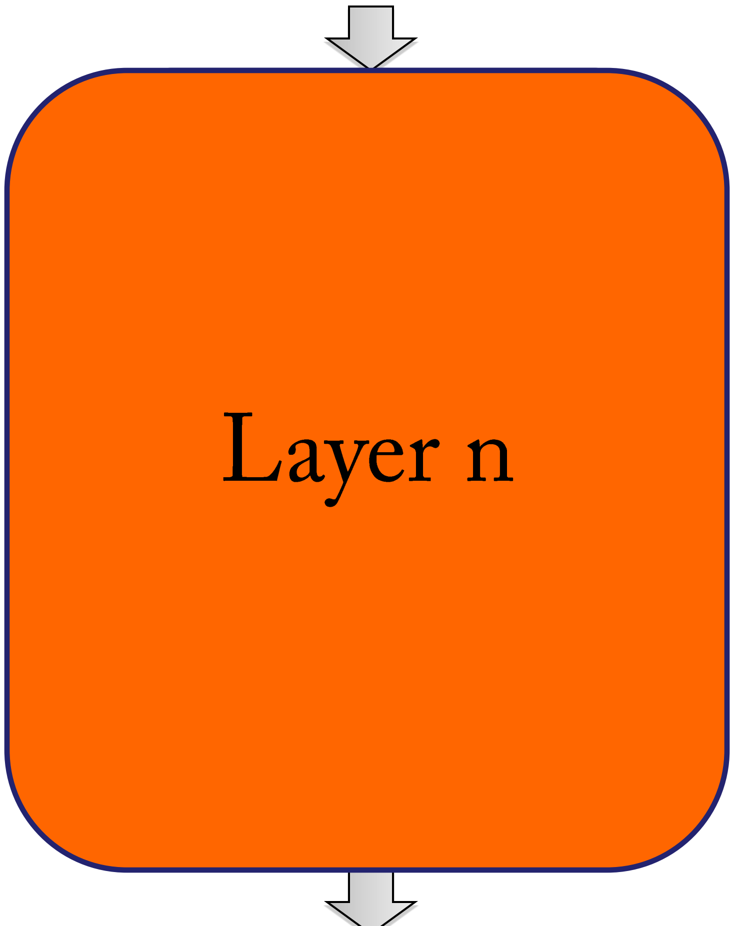
- Introduces local competition between features
  - Poor man's version of "Explaining away" in graphical models
  - Just like top-down models
  - But more local mechanism

- Also helps to scale activations at each layer better for learning
  - Makes energy surface more isotropic
  - So each gradient step makes more progress

- Empirically, seems to help a bit (1-2%) on ImageNet
  - More on other datasets (see [Jarrett et al. ICCV'09] for interesting analysis)

Single Layer Architecture

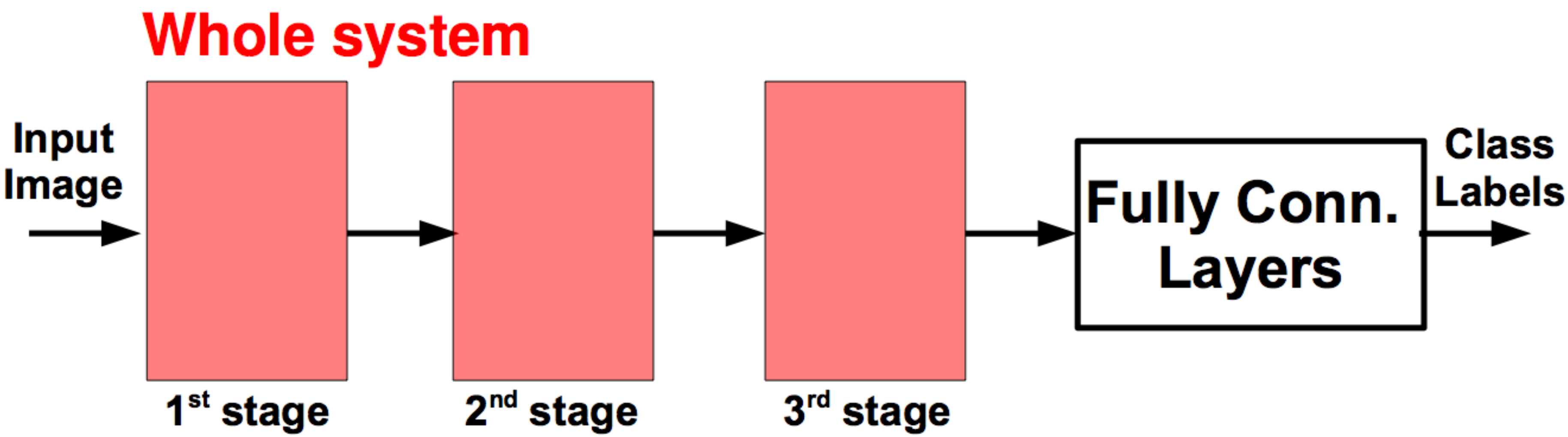
Input: Image Pixels / Features



Output: Features / Classifier

Slide: R. Fergus

ConvNets: Typical Architecture



Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM

Lazebnik et al. "...Spatial Pyramid Matching..." CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM

Sanchez et al. "Image classifcation with F.V.: Theory and practice" IJCV 2012

Slide: M-A Ranzatto

Ranzato

Breakthrough #1

RBM pretraining + backpropagation

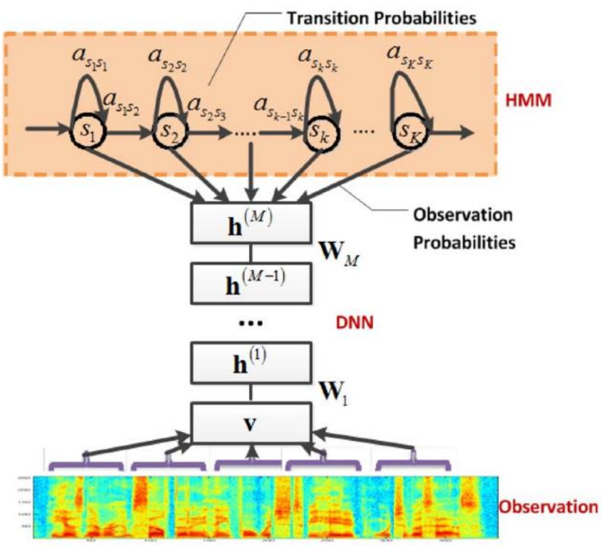


Fig. 1. Diagram of our hybrid architecture employing a deep neural network. The HMM models the sequential property of the speech signal, and the DNN models the scaled observation likelihood of all the senones (tied tri-phone states). The same DNN is replicated over different points in time.

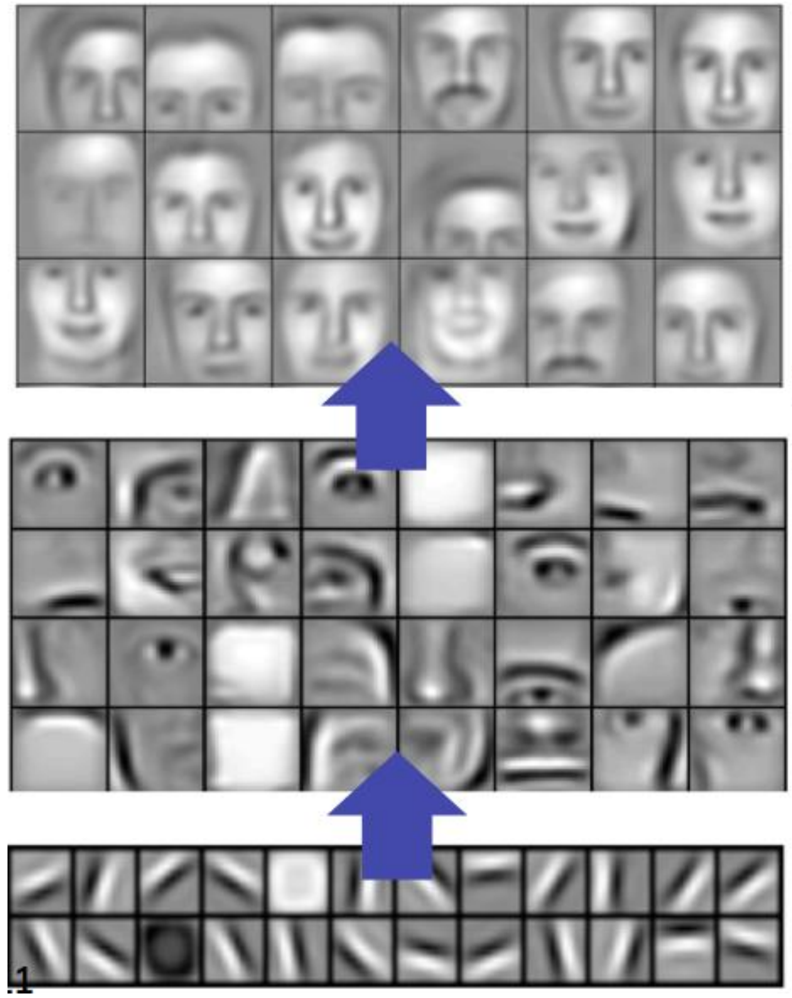
modeling technique	#params [10^6]	WER Hub5'00-SWB	WER RT03S-FSH
GMM, 40 mix DT 309h SI	29.4	23.6	27.4
NN 1 hidden-layer×4634 units	43.6	26.0	29.4
+ 2×5 neighboring frames	45.1	22.4	25.7
DBN-DNN 7 hidden layers×2048 units	45.1	17.1	19.6
+ updated state alignment	45.1	16.4	18.6
+ sparsification	15.2 nz	16.1	18.5
GMM 72 mix DT 2000h SA	102.4	17.1	18.6

"We realized that by modeling senones directly using DNNs, we had managed to outperform state-of-the-art conventional CD-GMM-HMM large-vocabulary speech-recognition systems by a relative error reduction of **more than 16 percent**. This is extremely significant when you consider that speech recognition has been **an active research area for more than five decades**."

http://research.microsoft.com/en-us/news/features/speechrecognition-082911.aspx

Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition, Dahl et. Al. 2011

Convolutional models & deep networks



Honglak Lee & Andrew Ng, ICML 2010

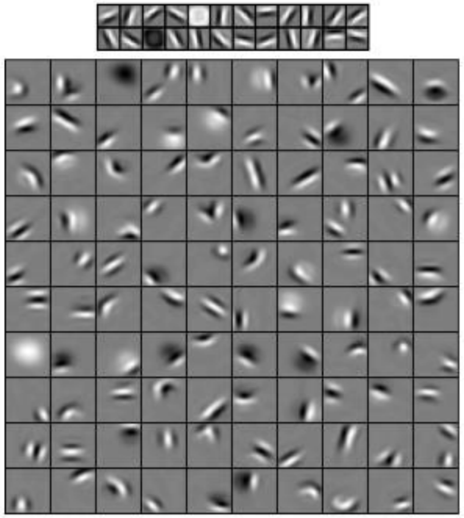


Figure 2. The first layer bases (top) and the second layer bases (bottom) learned from natural images. Each second layer basis (filter) was visualized as a weighted linear combination of the first layer bases.

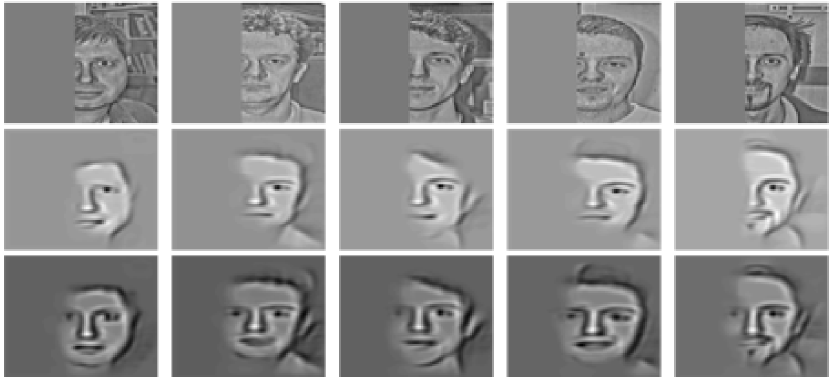


Figure 6. Hierarchical probabilistic inference. For each column: (top) input image. (middle) reconstruction from the second layer units after single bottom-up pass, by projecting the second layer activations into the image space. (bottom) reconstruction from the second layer units after 20 iterations of block Gibbs sampling.

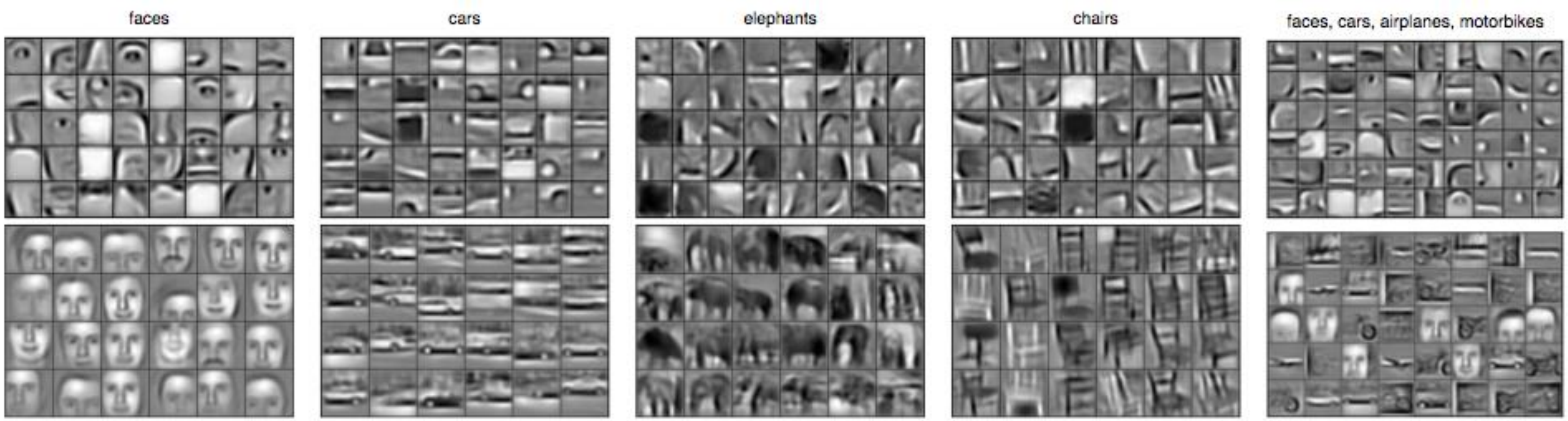
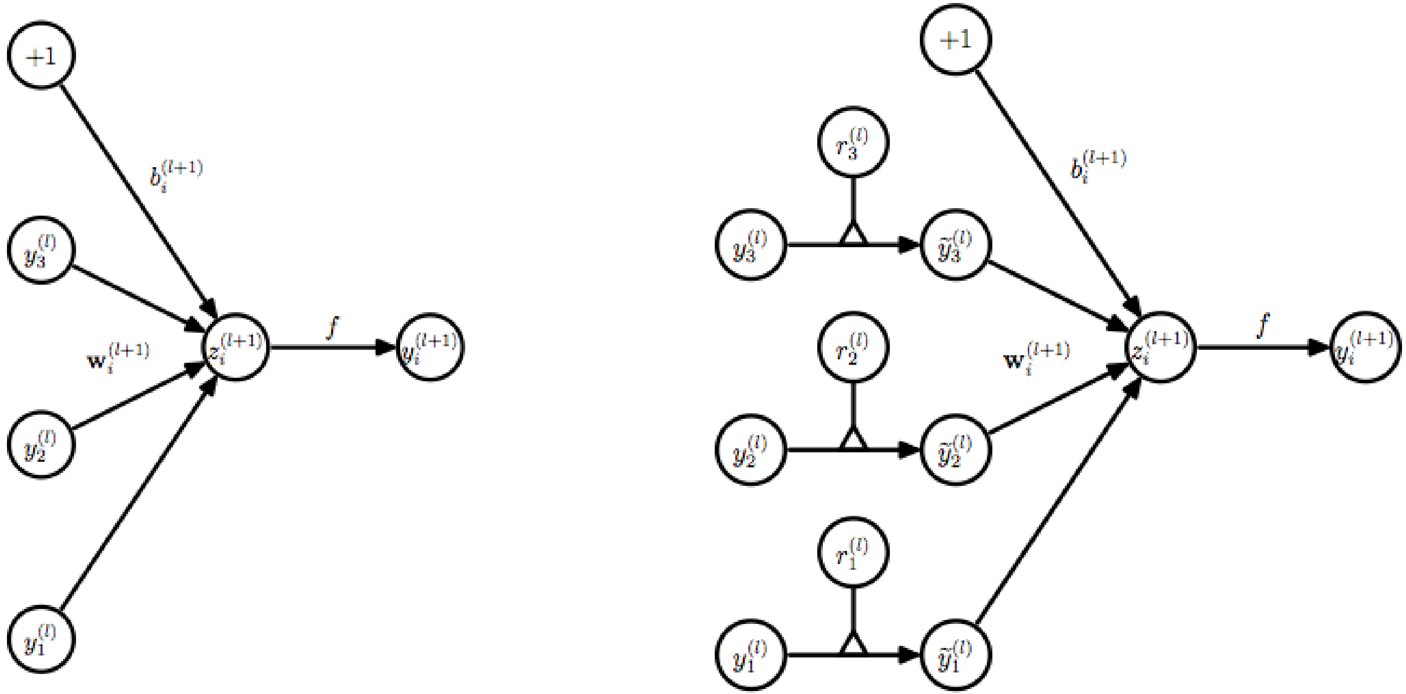


Figure 3. Columns 1-4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).

Dropout



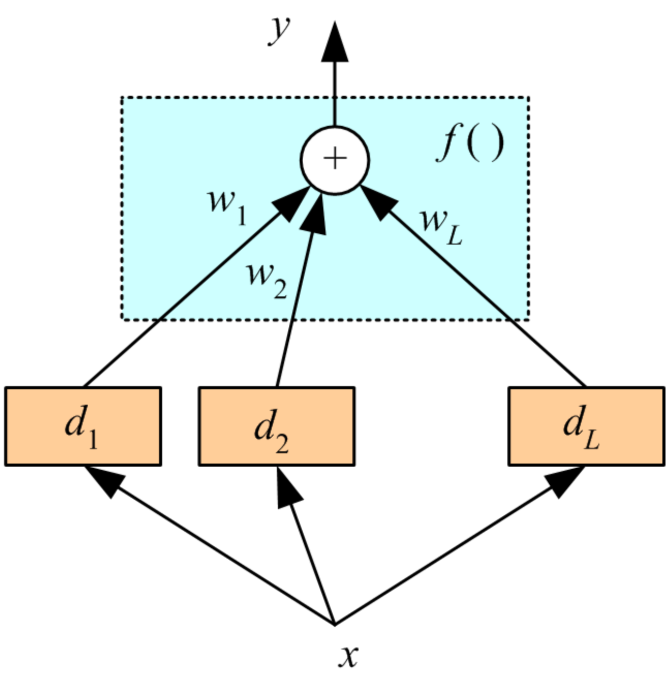
(a) Standard network

(b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

MLCV, Lecture 4: Voting Methods

- Give up idea of building `the' classifier
- Generate a group of base-learners which has higher accuracy when combined
- Main tasks
  - Generating the learners
  - Combining them



MLCV, Lecture 4: Why should this work?

- Committee of M predictors for target output $$y_{COM}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$
- Output: true value + error $$y(\mathbf{x}) = h(\mathbf{x}) + \epsilon(\mathbf{x})$$
- Average sum of squares error for m-th expert:

$$\mathbb{E}_{\mathbf{x}} = \left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

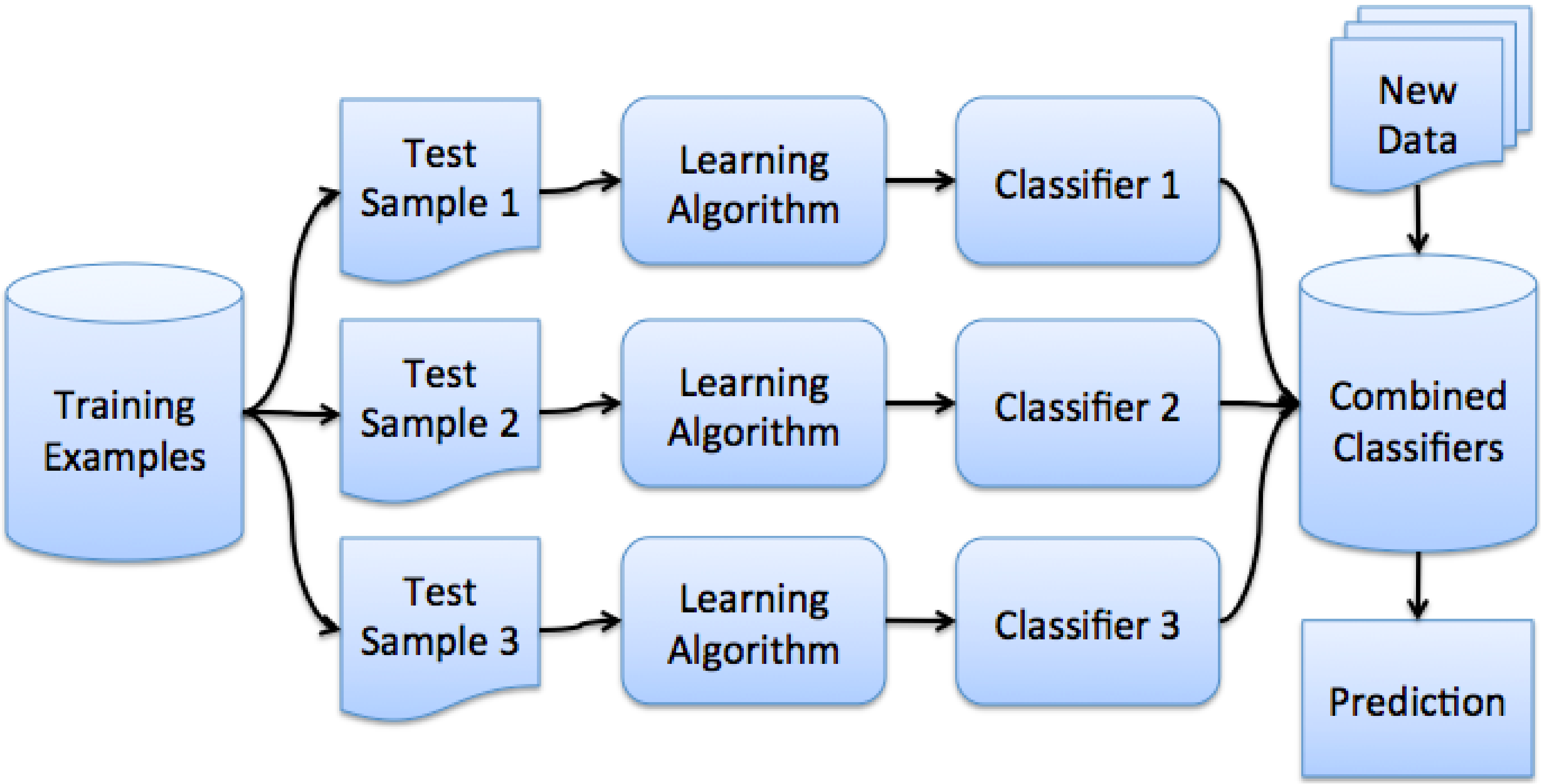
- Average error of individual members: $\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$
- Average error of committee:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right]$$

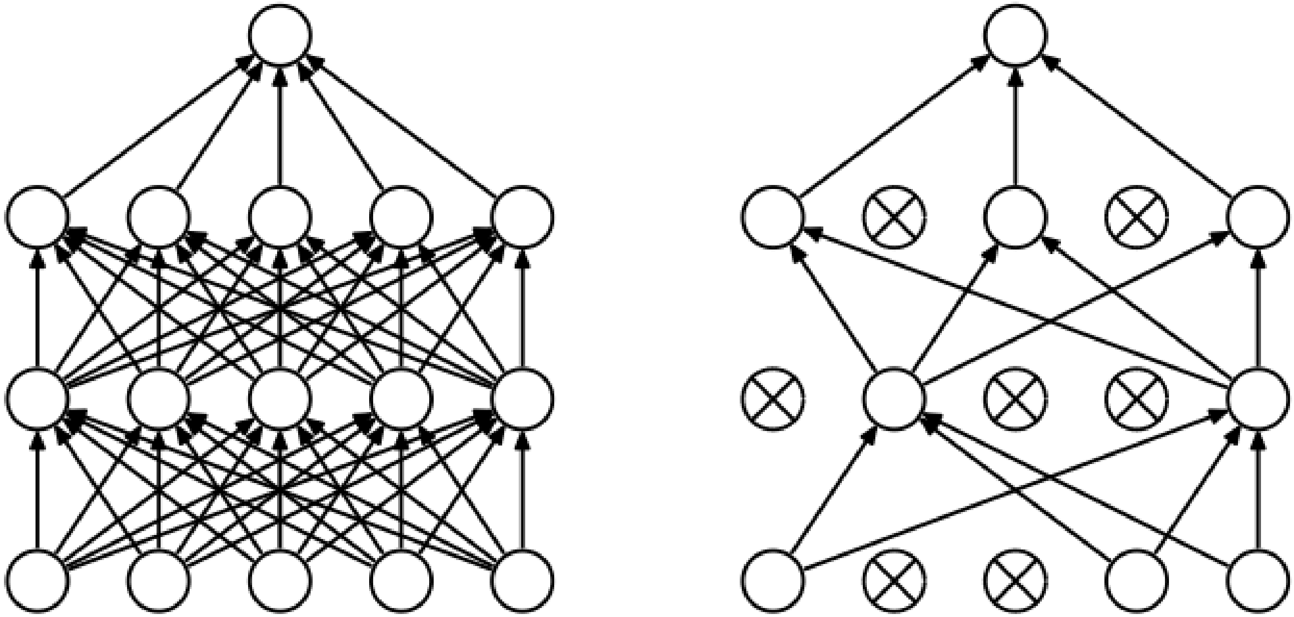
- If errors have zero mean and are uncorrelated: $$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] = 0$$ $$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_j(\mathbf{x})\right] = 0$$

then $$\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$$

Bootstrapped AGGregatING (BAGGING)



Dropout



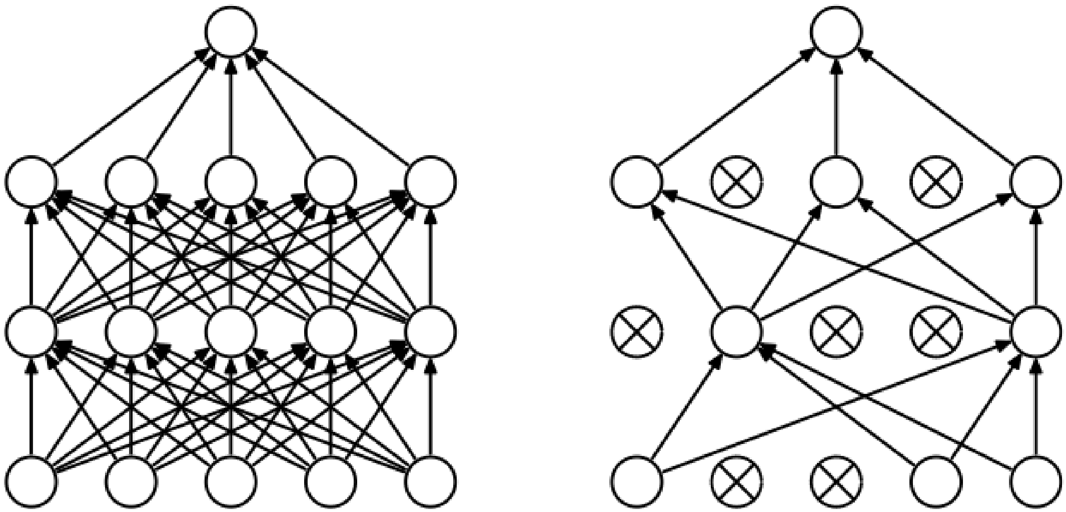
(a) Standard Neural Net (b) After applying dropout.

Each sample is processed by a 'decimated' neural net

- **Decimated nets: distinct classifiers**
- **They should all do the same job**

Improving neural networks by preventing co-adaptation of feature detectors
GE Hinton, N Srivastava, A Krizhevsky, I Sutskever, RR Salakhutdinov, arXiv, 2012, JMLR 2014
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Dropout

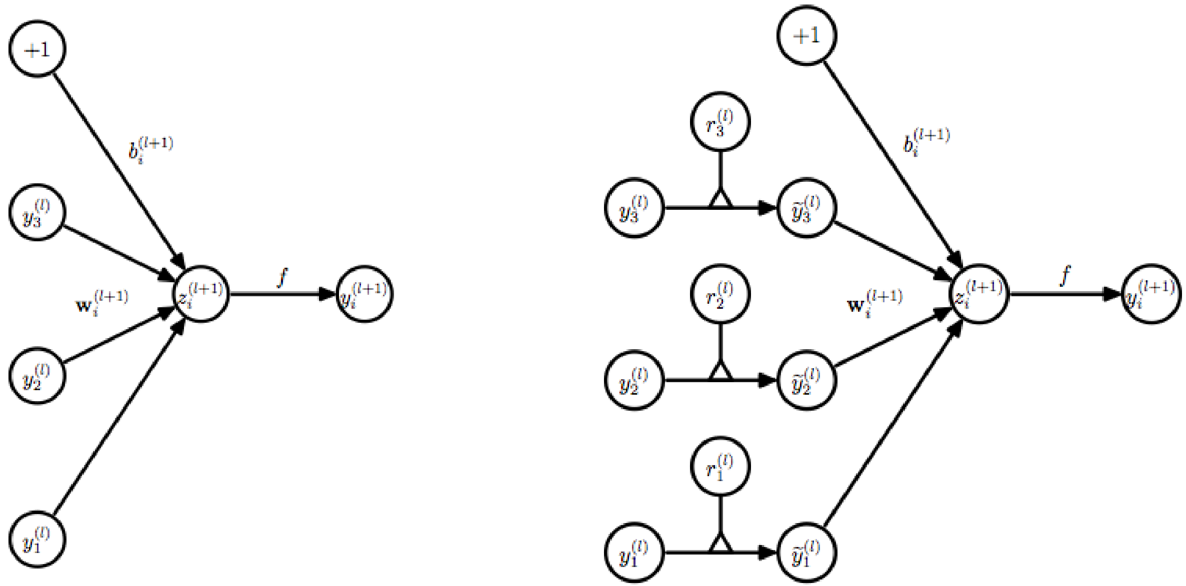


(a) Standard Neural Net (b) After applying dropout.

Applying dropout to a neural network amounts to sampling a "thinned" network from it. The thinned network consists of all the units that survived dropout (Figure 1b). A neural net with n units, can be seen as a collection of 2^n possible thinned neural networks. These networks all share weights so that the total number of parameters is still $O(n^2)$, or less. For each presentation of each training case, a new thinned network is sampled and trained. So training a neural network with dropout can be seen as training a collection of 2^n thinned networks with extensive weight sharing, where each thinned network gets trained very rarely, if at all.

Improving neural networks by preventing co-adaptation of feature detectors
GE Hinton, N Srivastava, A Krizhevsky, I Sutskever, RR Salakhutdinov, arXiv, 2012, JMLR 2014
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Dropout block



(a) Standard network

(b) Dropout network

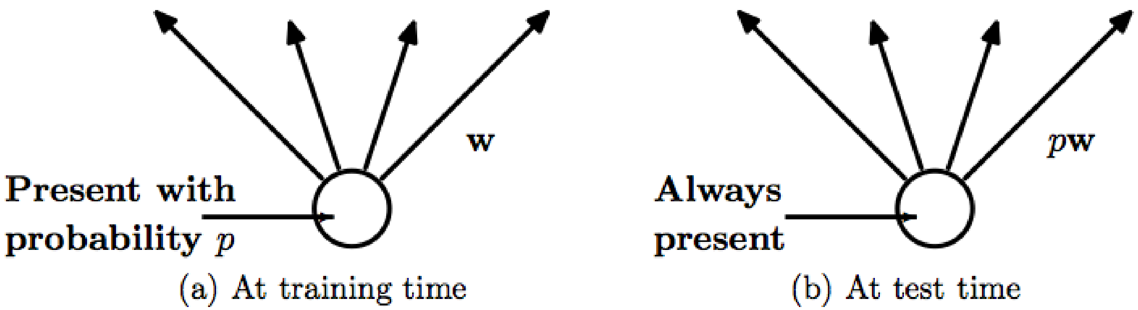
Figure 3: Comparison of the basic operations of a standard and dropout network.

$$
\begin{aligned}
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\mathbf{y}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}),
\end{aligned}
$$

$$
\begin{aligned}
r_j^{(l)} &\sim \mathrm{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

‘Feature noising’

Test time: Deterministic approximation



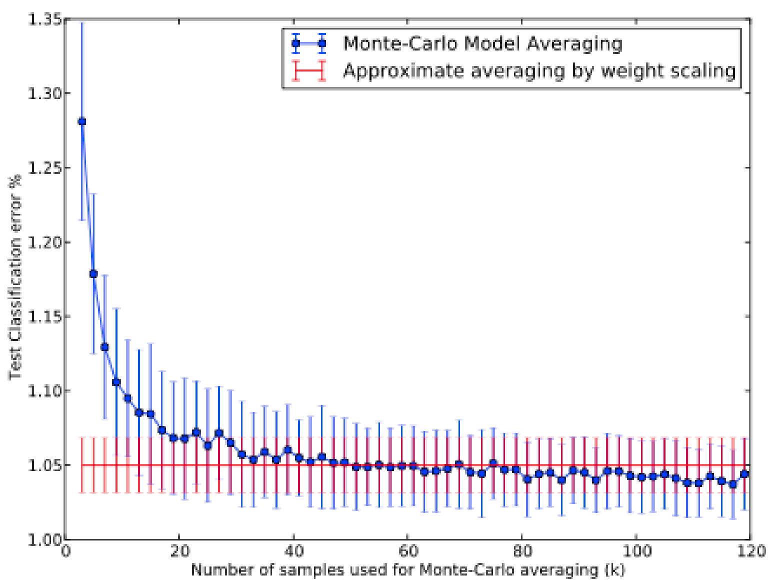
(a) At training time (b) At test time

Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

At test time, the weights are scaled as W(l) = pW(l) as shown in Figure 2. The resulting neural network is used without dropout.

An expensive but more correct way of averaging the models is to sample k neural nets using dropout for each test case and average their predictions. As k → ∞, this Monte-Carlo model average gets close to the true model average.

By computing the error for different values of k we can see how quickly the error rate of the finite-sample average approaches the error rate of the approximate model average.



Dropout performance

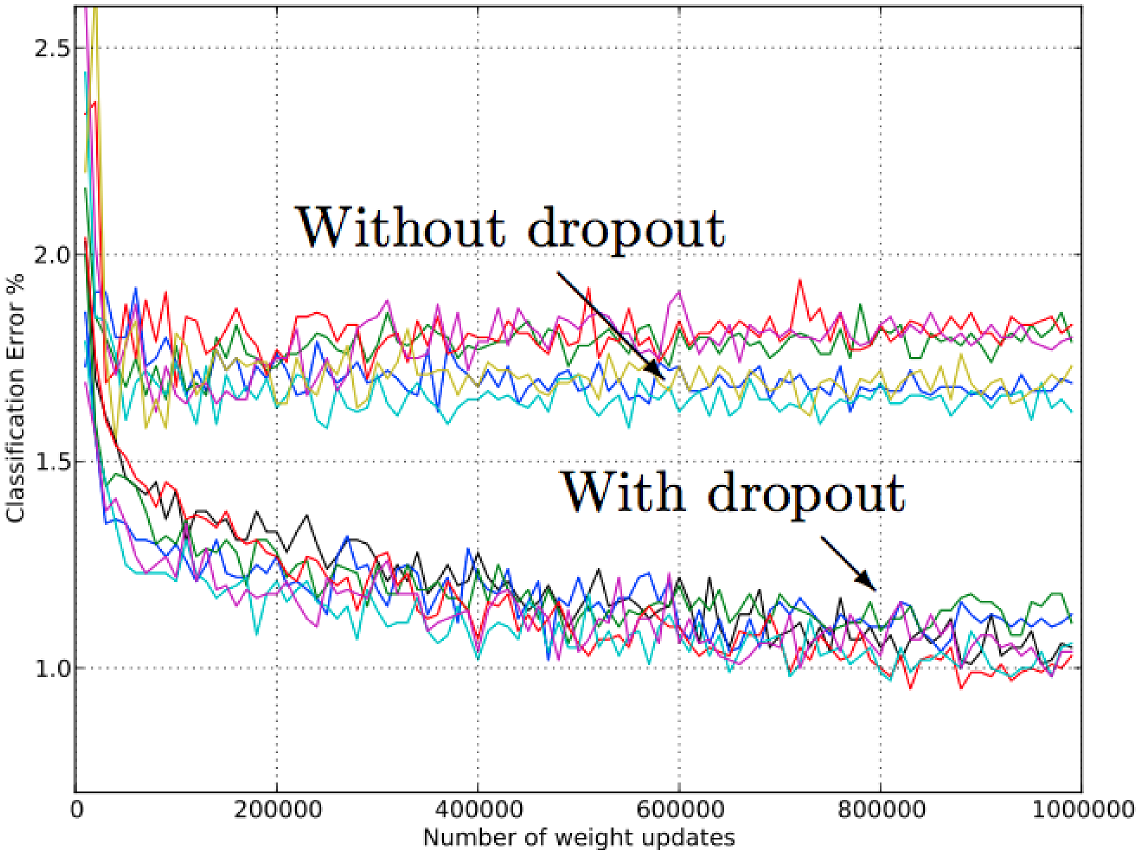


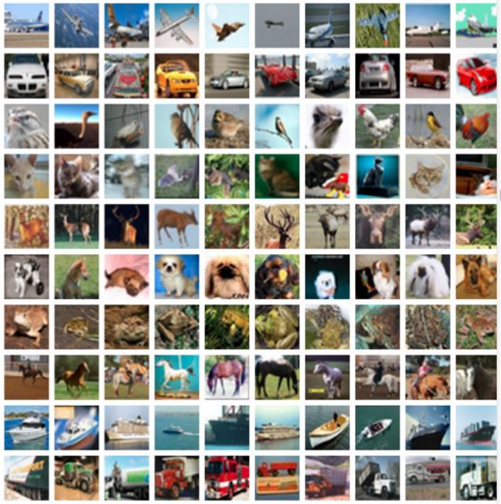
Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Dropout performance

(a) Street View House Numbers (SVHN)

Method	Error %
Binary Features (WDCH) (Netzer et al., 2011)	36.7
HOG (Netzer et al., 2011)	15.0
Stacked Sparse Autoencoders (Netzer et al., 2011)	10.3
KMeans (Netzer et al., 2011)	9.4
Multi-stage Conv Net with average pooling (Sermanet et al., 2012)	9.06
Multi-stage Conv Net + L2 pooling (Sermanet et al., 2012)	5.36
Multi-stage Conv Net + L4 pooling + padding (Sermanet et al., 2012)	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	2.80
Conv Net + max pooling + dropout in all layers	2.55
Conv Net + maxout (Goodfellow et al., 2013)	**2.47**
Human Performance	2.0

Table 3: Results on the Street View House Numbers data set.



(b) CIFAR-10

Method	CIFAR-10	CIFAR-100
Conv Net + max pooling (hand tuned)	15.60	43.48
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	15.13	42.51
Conv Net + max pooling (Snoek et al., 2012)	14.98	-
Conv Net + max pooling + dropout fully connected layers	14.32	41.26
Conv Net + max pooling + dropout in all layers	12.61	**37.20**
Conv Net + maxout (Goodfellow et al., 2013)	**11.68**	38.57

Table 4: Error rates on CIFAR-10 and CIFAR-100.

Breakthrough #2

Dropout performance

Figure 6: Some ImageNet test cases with the 4 most probable labels as predicted by our model. The length of the horizontal bars is proportional to the probability assigned to the labels by the model. Pink indicates ground truth.

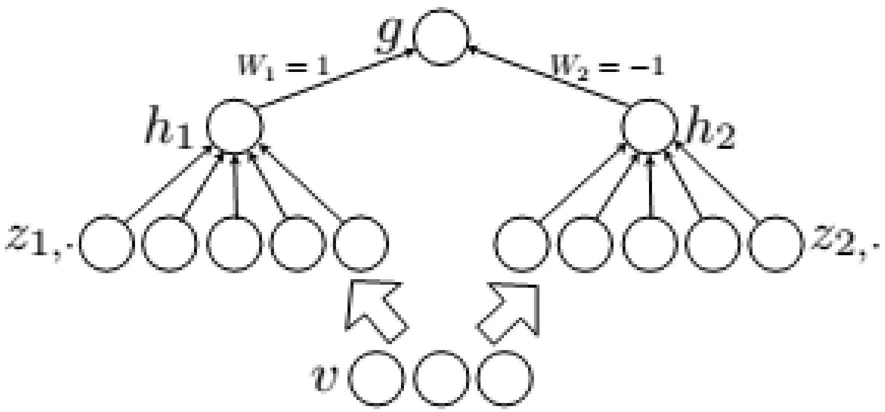
Model	Top-1	Top-5
Sparse Coding (Lin et al., 2010)	47.1	28.2
SIFT + Fisher Vectors (Sanchez and Perronnin, 2011)	45.7	25.7
Conv Net + dropout (Krizhevsky et al., 2012)	37.5	17.0

Table 5: Results on the ILSVRC-2010 test set.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

Maxout (Goodfellow et al, 2013)



In a convolutional network, a maxout feature map can be constructed by taking the maximum across k affine feature maps (i.e., pool across channels, in addition spatial locations)

http://www-etud.iro.umontreal.ca/~goodfeli/maxout.html

Maxout units vs. ReLU

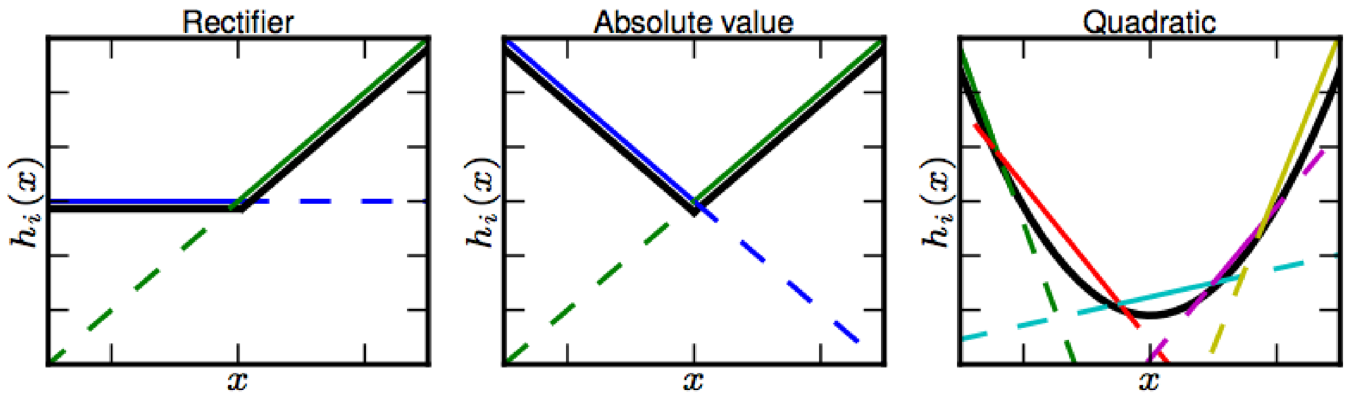


Figure 1. Graphical depiction of how the maxout activation function can implement the rectified linear, absolute value rectifier, and approximate the quadratic activation function. This diagram is 2D and only shows how maxout behaves with a 1D input, but in multiple dimensions a maxout unit can approximate arbitrary convex functions.

Filters learned with maxout

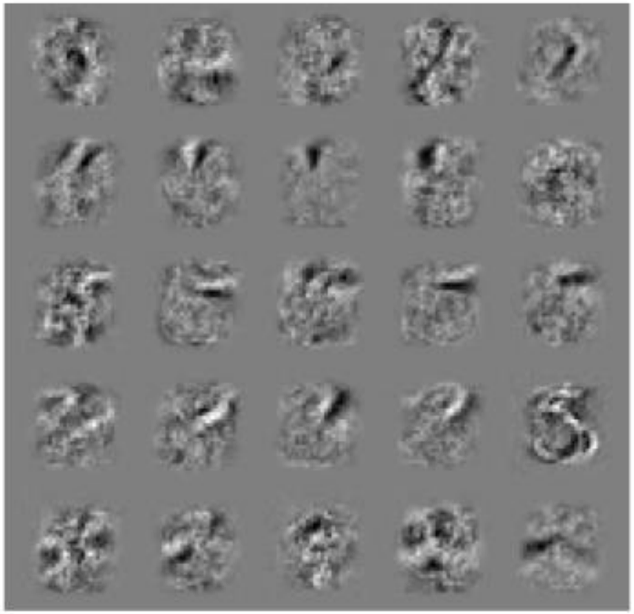
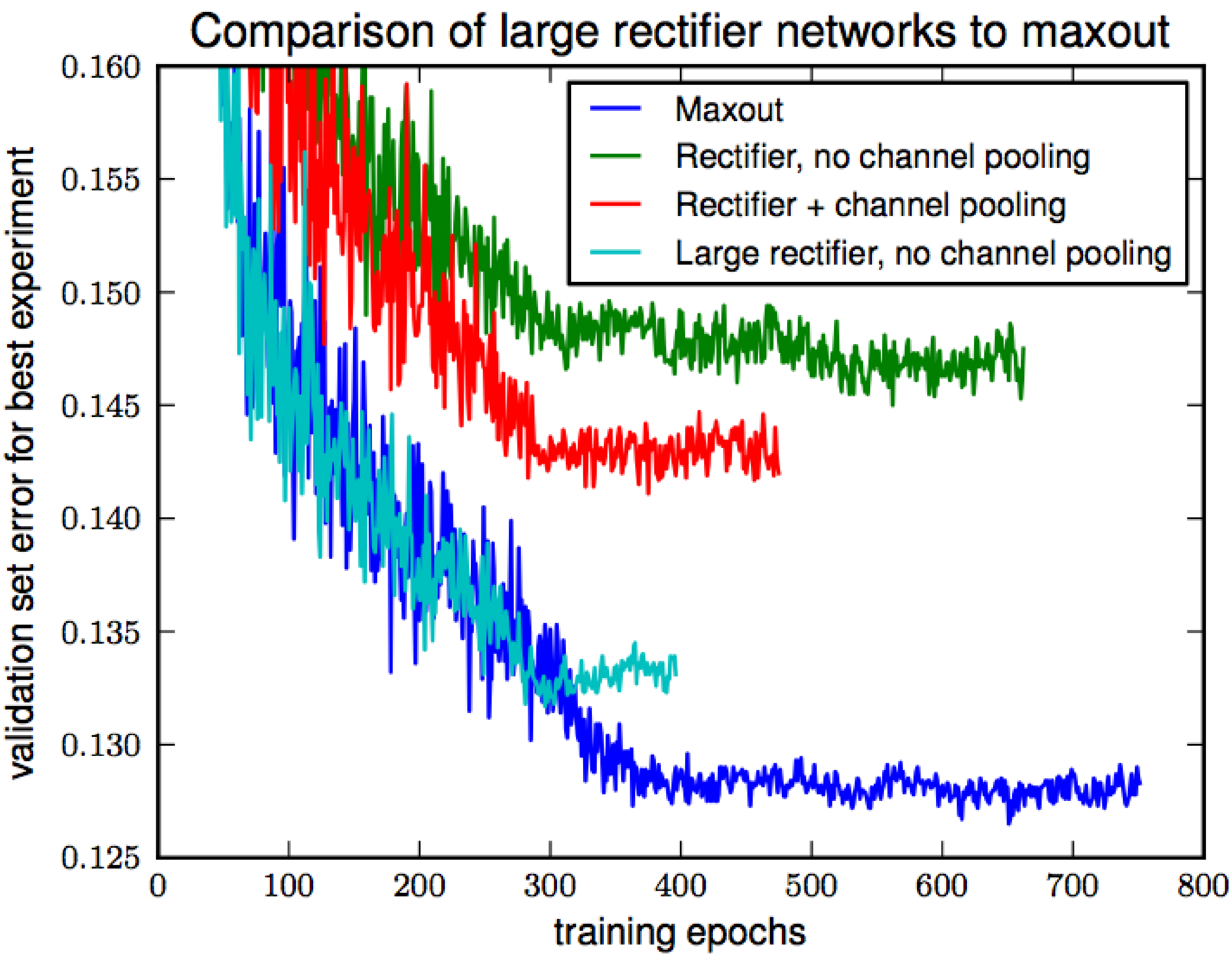
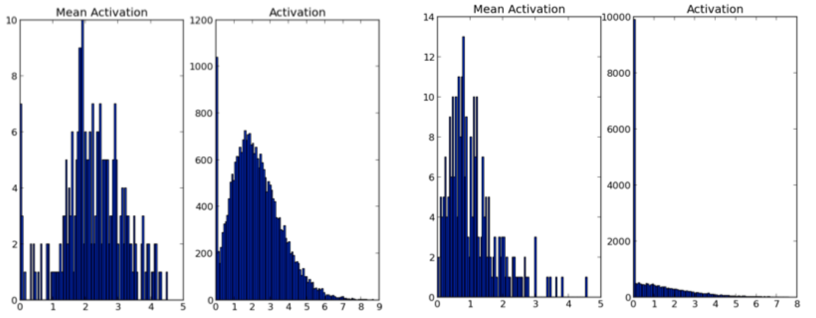


Figure 4. Example filters learned by a maxout MLP trained with dropout on MNIST. Each row contains the filters whose responses are pooled to form a maxout unit.

Maxout vs. ReLUs



Maxout: responses are not sparse



(a) Without dropout

(b) Dropout with $p = 0.5$.

Figure 8: Effect of dropout on sparsity. ReLUs were used for both models. **Left**: The histogram of mean activations shows that most units have a mean activation of about 2.0. The histogram of activations shows a huge mode away from zero. Clearly, a large fraction of units have high activation. **Right**: The histogram of mean activations shows that most units have a smaller mean mean activation of about 0.7. The histogram of activations shows a sharp peak at zero. Very few units have high activation.

Dropout + ReLUs:
sparse responses

Why? Regularizer cost decreases

How: units get stuck at zero ('dead' units)

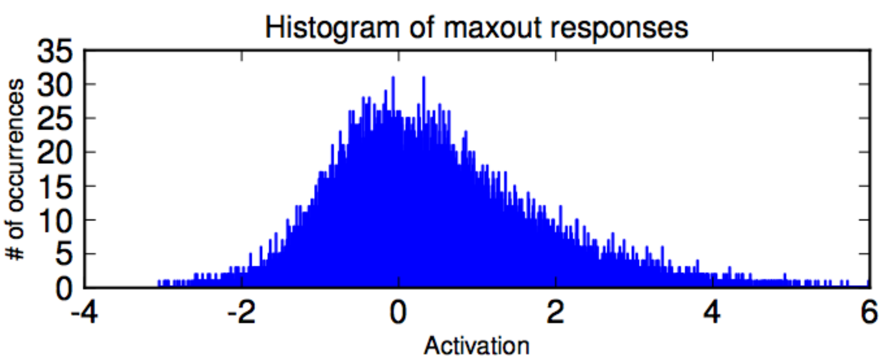


Figure 2. The activations of maxout units are not sparse.

Dropout + MaxOut:
non-sparse responses

Why? No '0' term among max-ed filters

Maxout does not have ‘dead units’

ReLUs: units get stuck below 0, never updated

Maxout: always differentiable, with nonzero derivative

A GLM defines a conditional distribution over a response $y \in \mathcal{Y}$ given an input feature vector $x \in \mathbb{R}^d$:

$$p_\beta(y \mid x) \stackrel{\text{def}}{=} h(y) \exp\{y\, x \cdot \beta - A(x \cdot \beta)\}, \quad \ell_{x,y}(\beta) \stackrel{\text{def}}{=} -\log p_\beta(y \mid x). \tag{1}$$

Here, $h(y)$ is a quantity independent of x and β, $A(\cdot)$ is the log-partition function, and $\ell_{x,y}(\beta)$ is the loss function (i.e., the negative log likelihood); Table 1 contains a summary of notation. Common examples of GLMs include linear ($\mathcal{Y} = \mathbb{R}$), logistic ($\mathcal{Y} = \{0, 1\}$), and Poisson ($\mathcal{Y} = \{0, 1, 2, \dots\}$) regression.

Given n training examples (x_i, y_i), the standard maximum likelihood estimate $\hat{\beta} \in \mathbb{R}^d$ minimizes the empirical loss over the training examples:

$$\hat{\beta} \stackrel{\text{def}}{=} \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \ell_{x_i,\, y_i}(\beta). \tag{2}$$

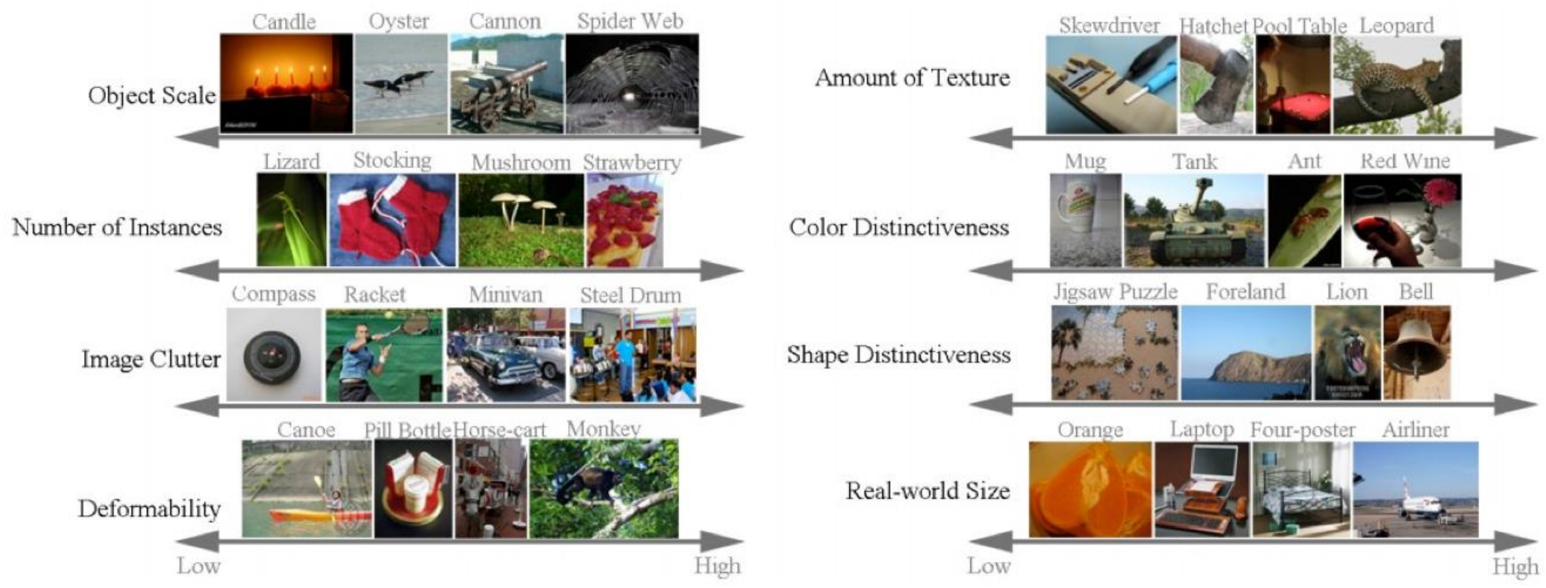
With artificial feature noising, we replace the observed feature vectors x_i with noisy versions $\tilde{x}_i = \nu(x_i, \xi_i)$, where ν is our noising function and ξ_i is an independent random variable. We first create many noisy copies of the dataset, and then average out the auxiliary noise. In this paper, we will consider two types of noise:

Question: When does CNN work well and when does it not?

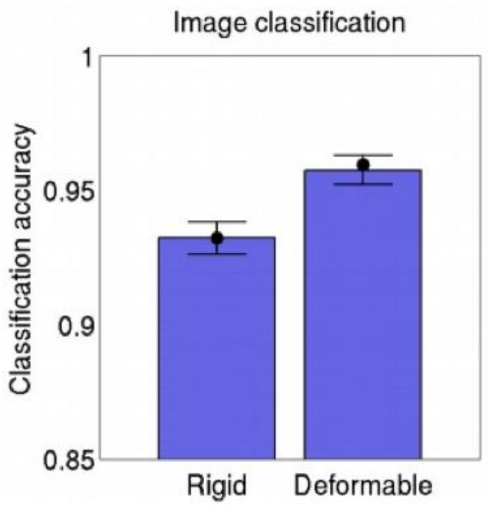
ImageNet (ILSVRC competition) analysis

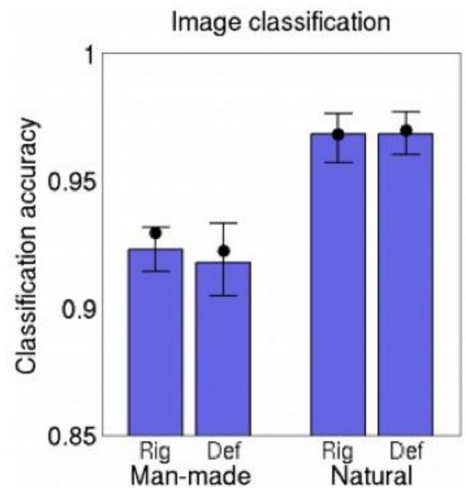
1. Detecting avocados to zucchinis: what have we done, and where are we going?
2. ImageNet Large Scale Visual Recognition Challenge

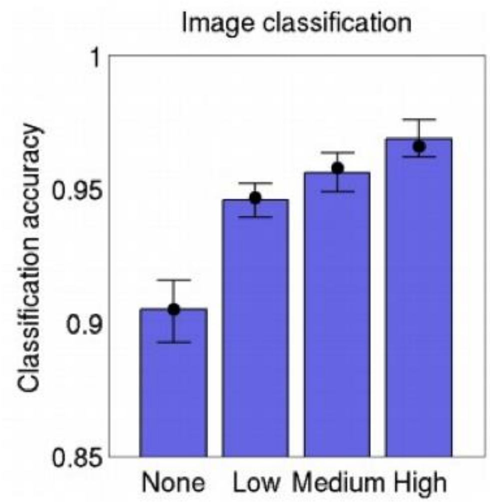
[Olga Russakovsky et al.]



Slide credit:Fei-Fei Li

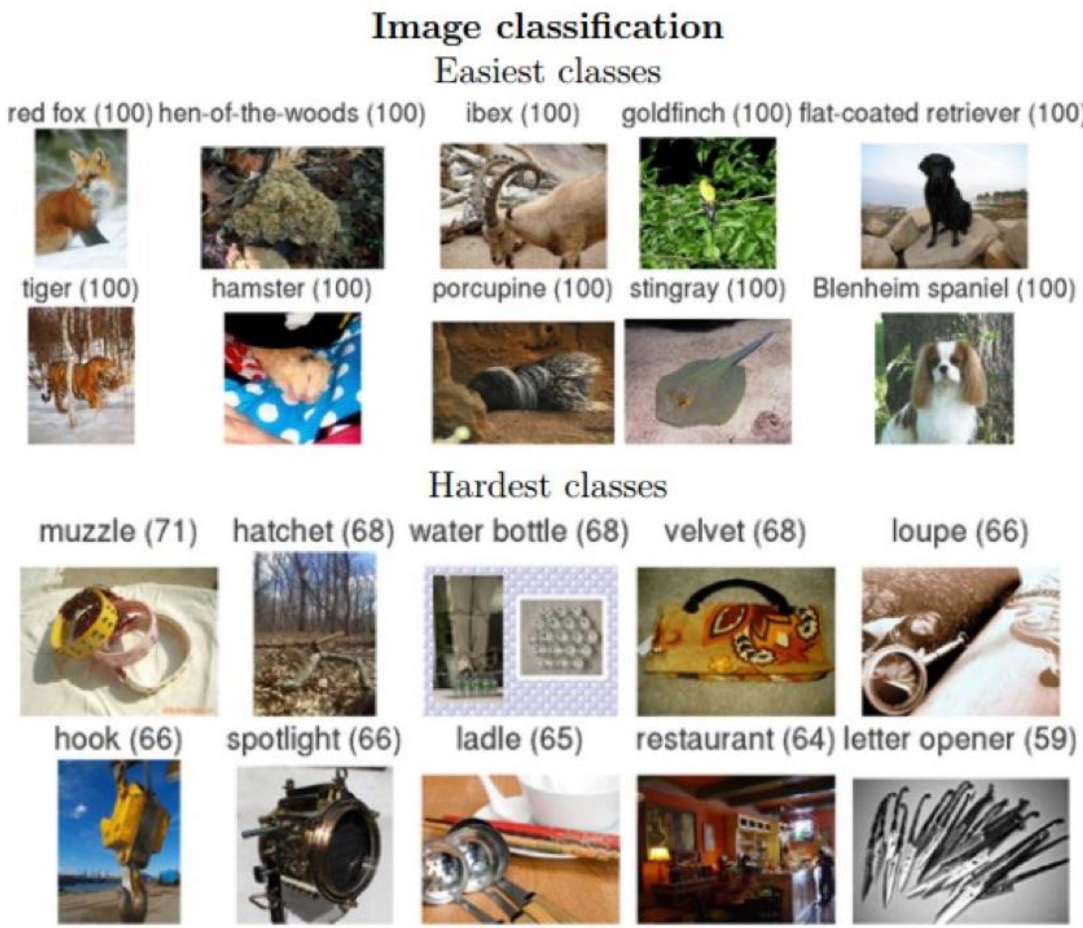






(Amount of texture)

Slide credit:Fei-Fei Li



Slide credit:Fei-Fei Li

CNN vs. Human

[What I learned from competing against a ConvNet on ImageNet]
Karpathy, 2014: http://bit.ly/humanvsconvnet

Try it out yourself: http://cs.stanford.edu/people/karpathy/ilsvrc/

Slide credit:Fei-Fei Li

	GoogLeNet correct	GoogLeNet wrong
Human correct	1352/1500 ☺	72/1500 • Objects very small or thin • Abstract representations • Image filters
Human wrong	46/1500 • Fine-grained recognition • Class unawareness • Insufficient training data	30/1500 • Multiple objects • Incorrect annotations

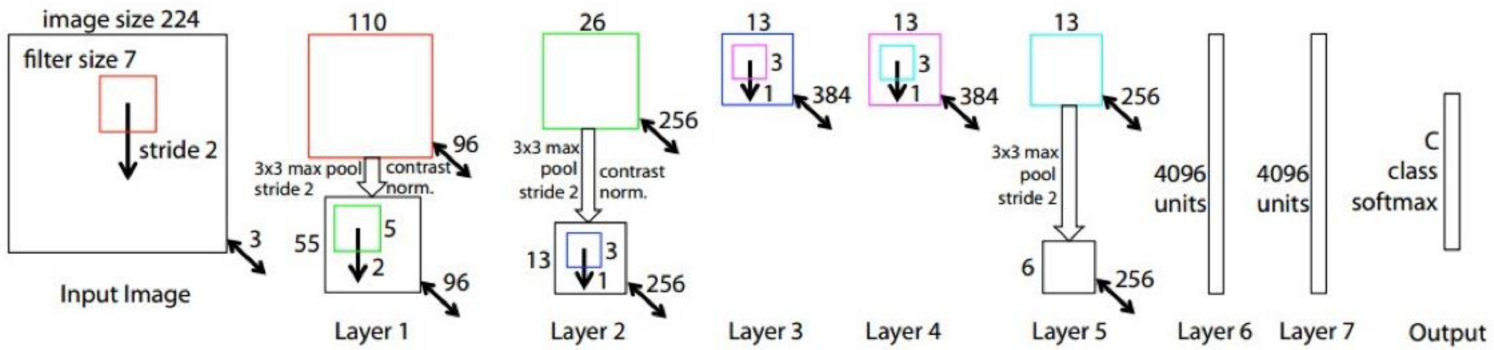
GoogLeNet: 6.7%
Team Human: 5.1% phew...

rule, ruler	king crab, Alaska crab	sidewinder	saltshaker, salt shaker	reel	hatchet	schipperke
pencil box, pencil case	pizza, pizza pie	maze, labyrinth	pill bottle	stethoscope	vase	schipperke
rubber eraser, rubber	strawberry	gar, garfish	water bottle	whistle	pitcher, ewer	groenendael
ballpoint, ballpoint pen	orange	valley, vale	lotion	ice lolly, lolly	coffeepot	doormat, welcome mat
pencil sharpener	fig	hammerhead	hair spray	hair spray	mask	teddy, teddy bear
carpenter's kit, tool kit	ice cream, icecream	sea snake	beer bottle	maypole	cup	jigsaw puzzle

Slide credit:Fei-Fei Li

Understanding the source of ConvNet performance

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]

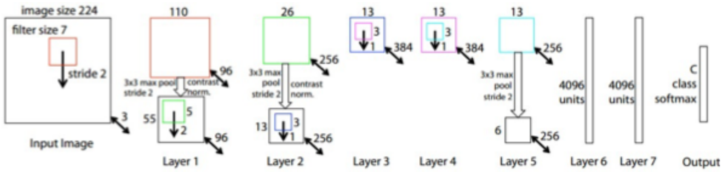


Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

Slide credit:Fei-Fei Li

Understanding the source of ConvNet performance

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]



- Remove 2 FC layers (6,7): lose some small performance
- Remove 2 Conv layers (3,4): lose about equal performance
- Remove 2FC 2Conv (3,4,6,7): Very bad (71% error)

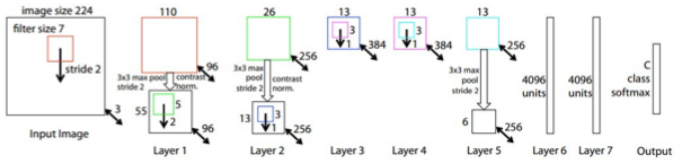
=> Depth is important

Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

Slide credit:Fei-Fei Li

Understanding the source of ConvNet performance

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]



- Remove 2 FC layers (6,7): lose some small performance
- Remove 2 Conv layers (3,4): lose about equal performance
- Remove 2FC 2Conv (3,4,6,7): Very bad (71% error)

=> Depth is important

- Changing size of FC layers: little to no improvement
- Changing size of Conv layers: reasonable improvement!

Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

Our Model (as per Fig. 3)	33.1	38.4	16.5
Adjust layers 6,7: 2048 units	38.2	40.2	17.6
Adjust layers 6,7: 8192 units	22.0	38.8	17.0
Adjust layers 3,4,5: 512,1024,512 maps	18.8	**37.5**	**16.0**
Adjust layers 6,7: 8192 units and Layers 3,4,5: 512,1024,512 maps	**10.0**	38.3	16.9

Slide credit:Fei-Fei Li

GoogleNet

[Going deeper with convolutions, Szegedy et al., 2014]

GoogLeNet
12x less params than Krizhevsky et al.
=> ~5M params

Q: How to reduce the number of parameters?
A: Throw away the FC layers (only part of their answer (Inception module

After last pooling layer, volume is of size **[7x7x1024]**
Normally you would place the first 4096-D FC layer here (Many M param

Instead: use Average pooling in each depth slice:
=> **[1x1x1024]**
performance actually improves 0.6% (less overfitting?)

Slide credit:Fei-Fei Li

Summary: What makes Convnets Tick

- - depth
  - small filter sizes
  - Conv layers > FC layers

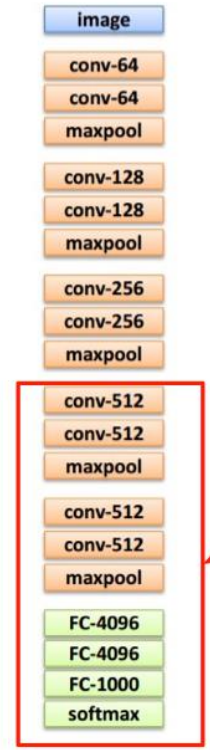
Slide credit:Fei-Fei Li

Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

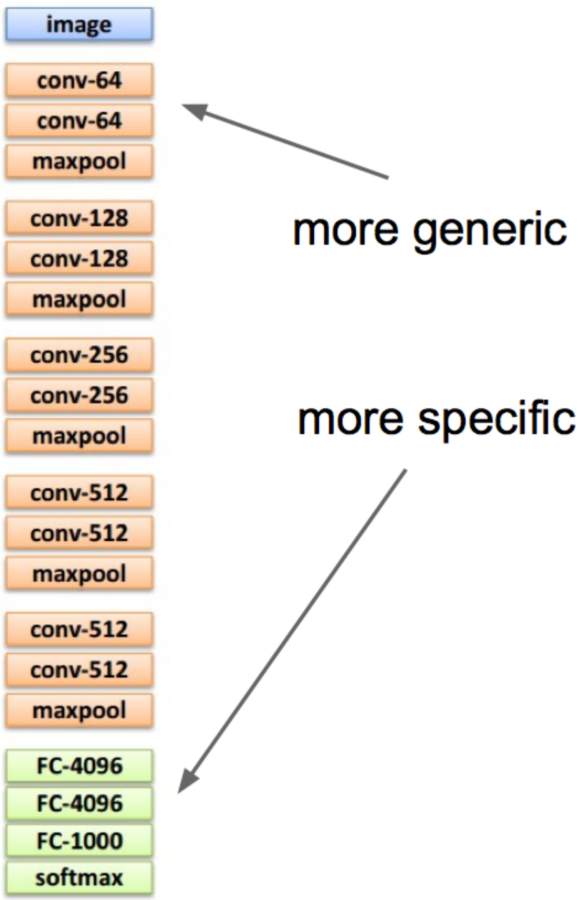


3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

retrain bigger portion of the network, or even all of it.

tip: use only ~1/10th of the original learning rate in finetuning to player, and ~1/100th on intermediate layers

Slide credit:Fei-Fei Li



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Slide credit:Fei-Fei Li

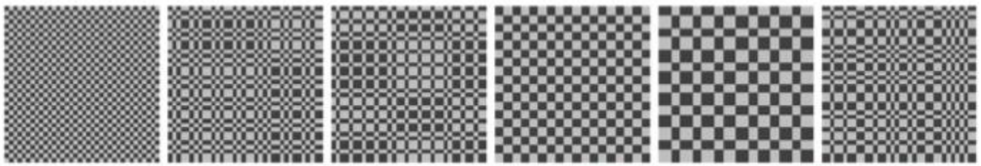
Data Augmentation

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.)

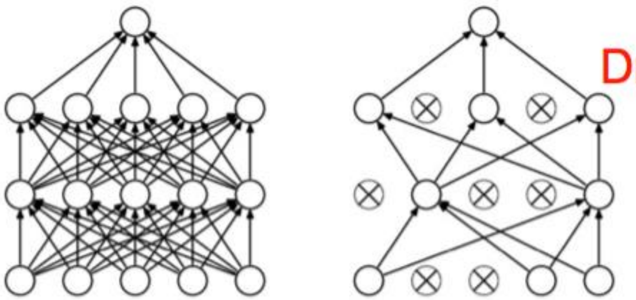
- Simple: Change contrast small amounts, jitter the color distributions, etc.
- Vignette,... (go crazy)

Notice the more general theme:

1. Introduce a form of randomness in forward pass
2. Marginalize over the noise distribution during prediction



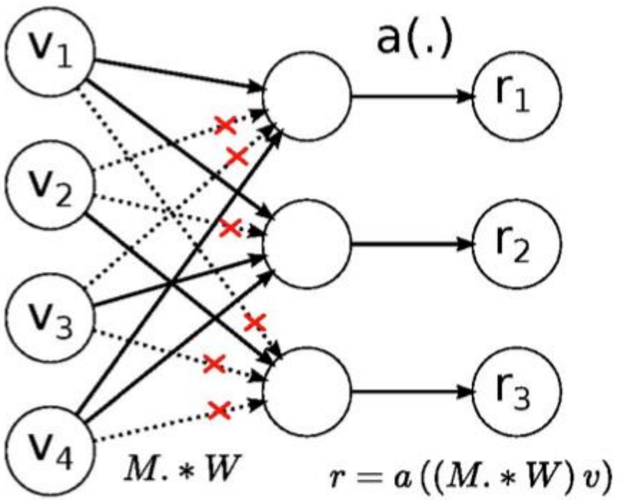
Fractional Pooling



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

Dropout

Data Augmentation,
Model Ensembles



DropConnect