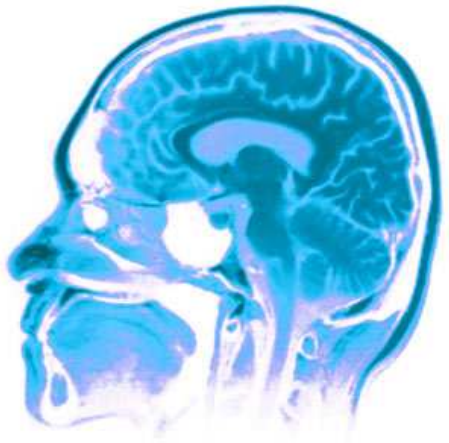
CPSC540



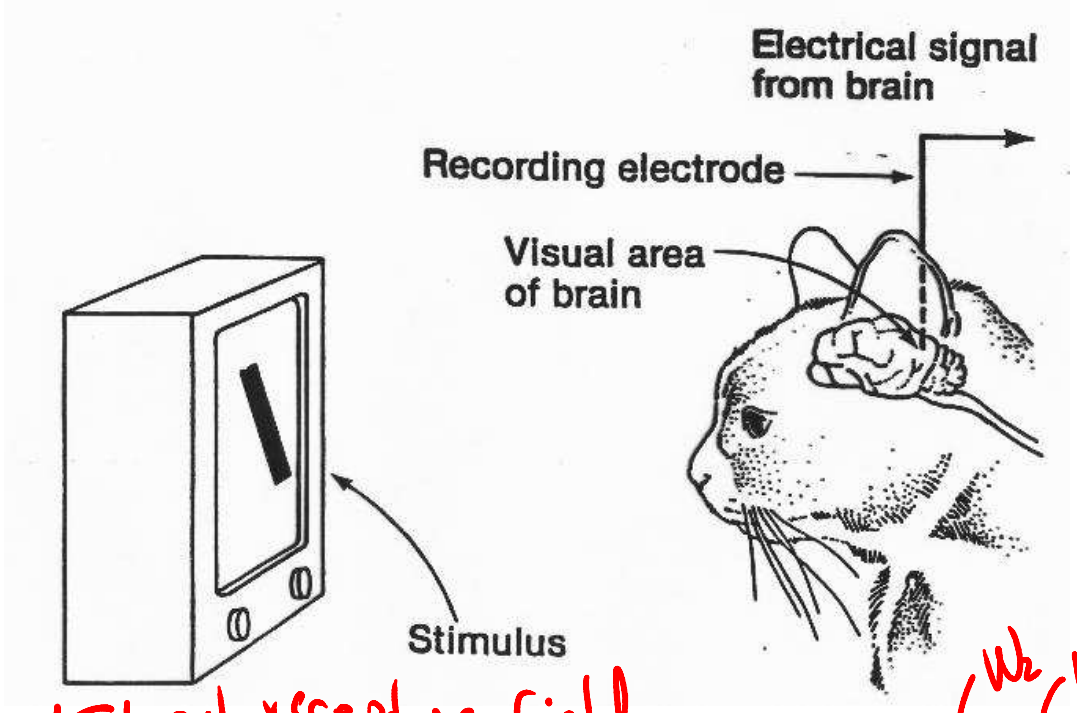
Deep learning

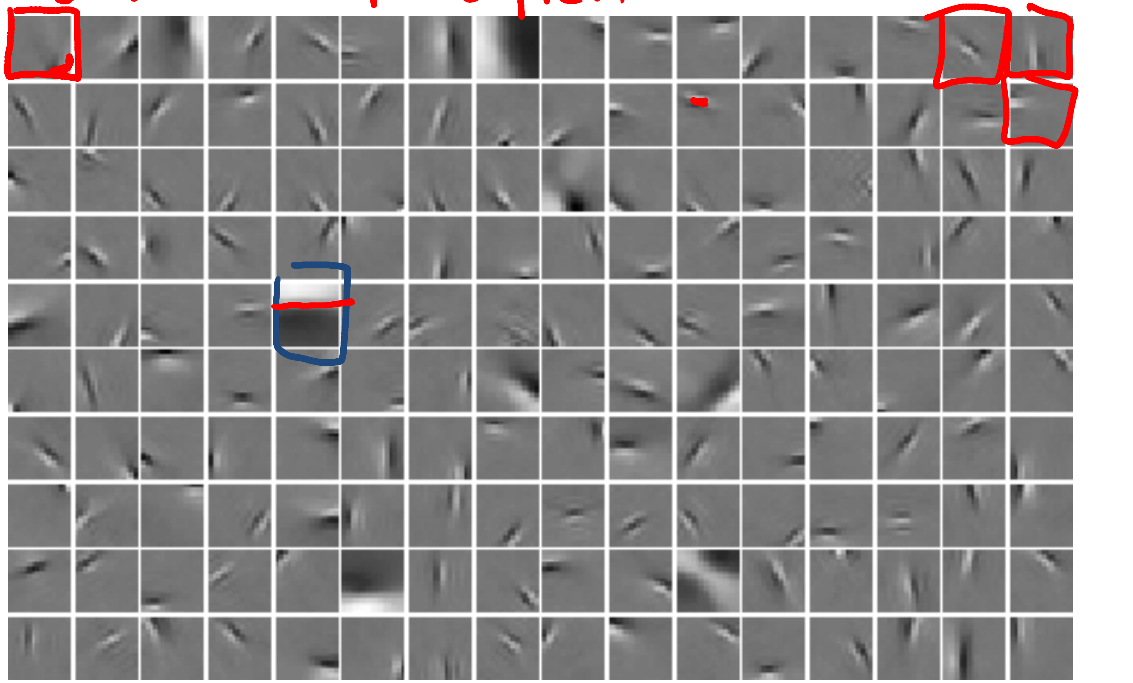
Nando de Freitas
March, 2013
University of British Columbia

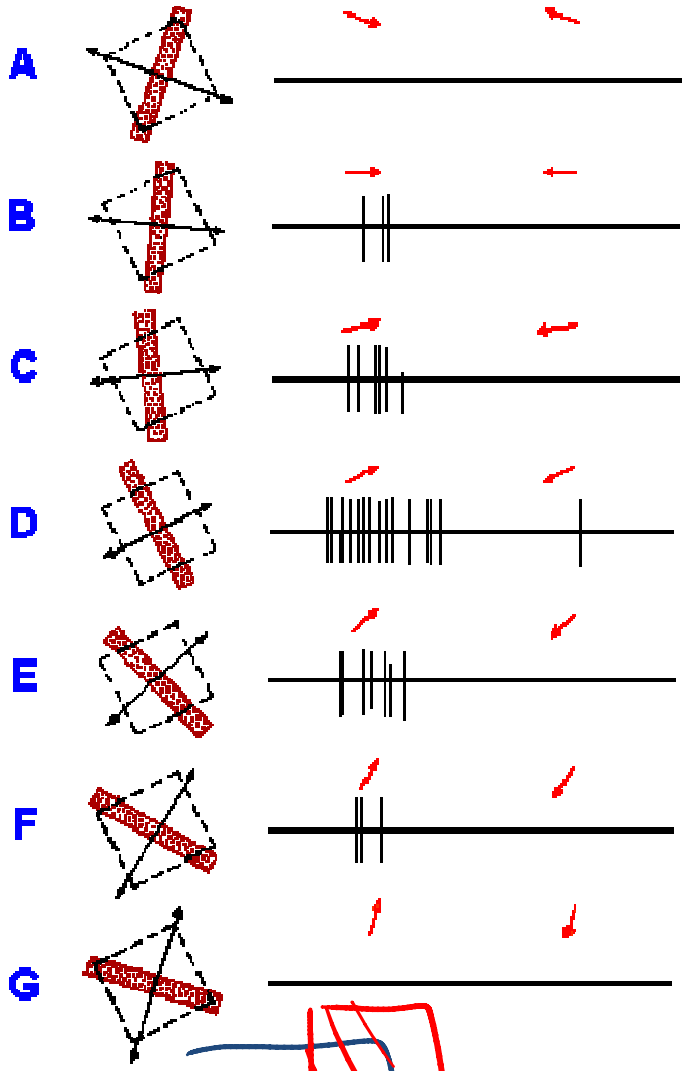
Outline of the lecture

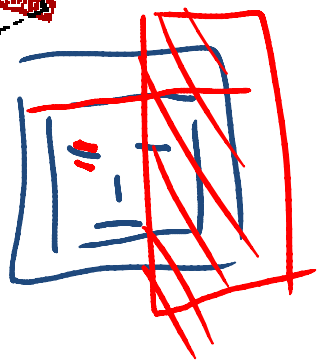
This lecture provides an overview of two state-of-the-art neural networks: The google net and dropout nets. It discusses:

- ❑ Unsupervised learning with NNs (autoencoders)
- ❑ Object recognition (classification and detection)
- ❑ Pooling
- ❑ Local contrast normalization
- ❑ Group regularization
- ❑ Regularization via dropout

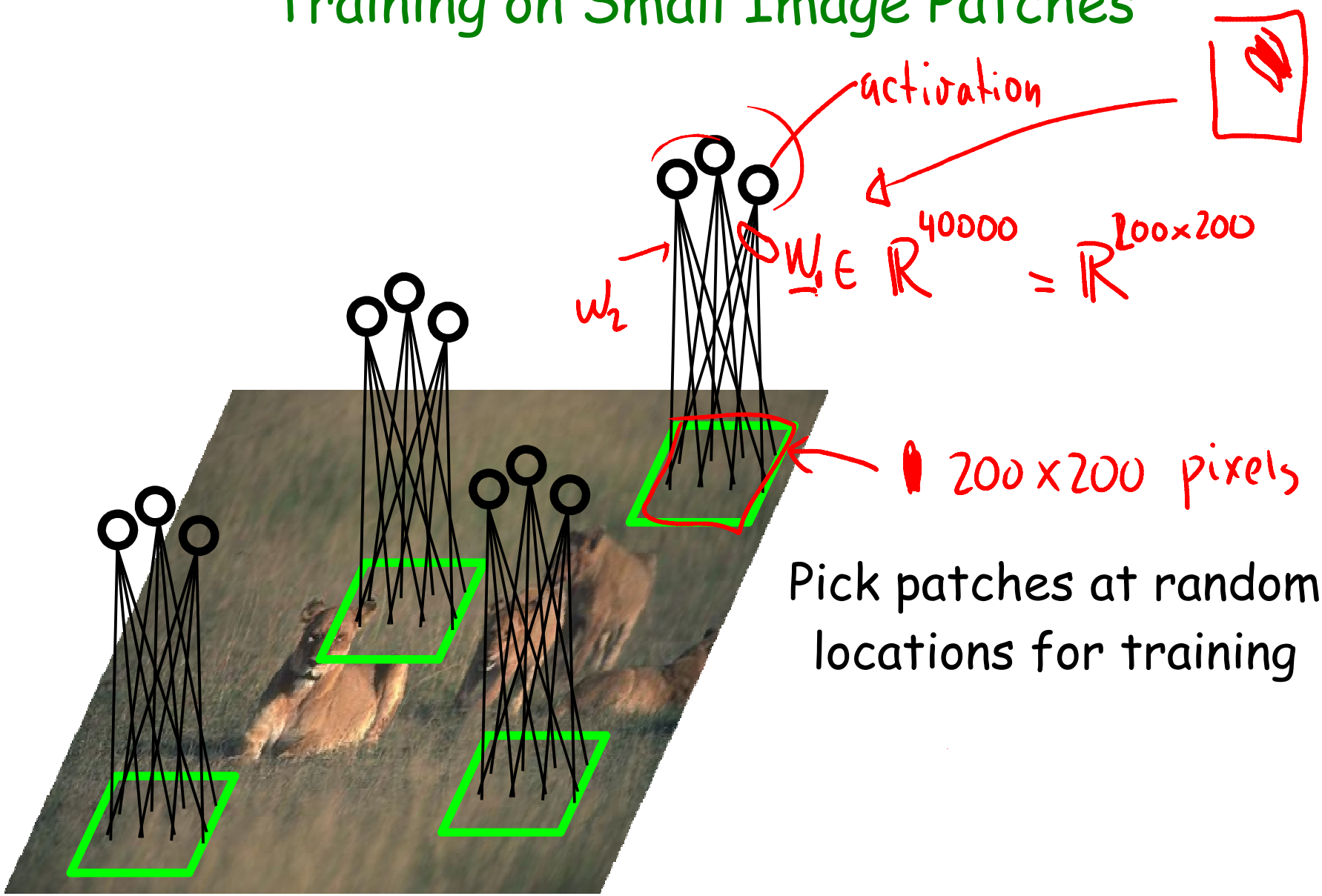

Electrical signal
from brain
Recording electrode
Visual area
of brain
Stimulus


local receptive field
W2
W1


A
B
C
D
E
F
G



Training on Small Image Patches

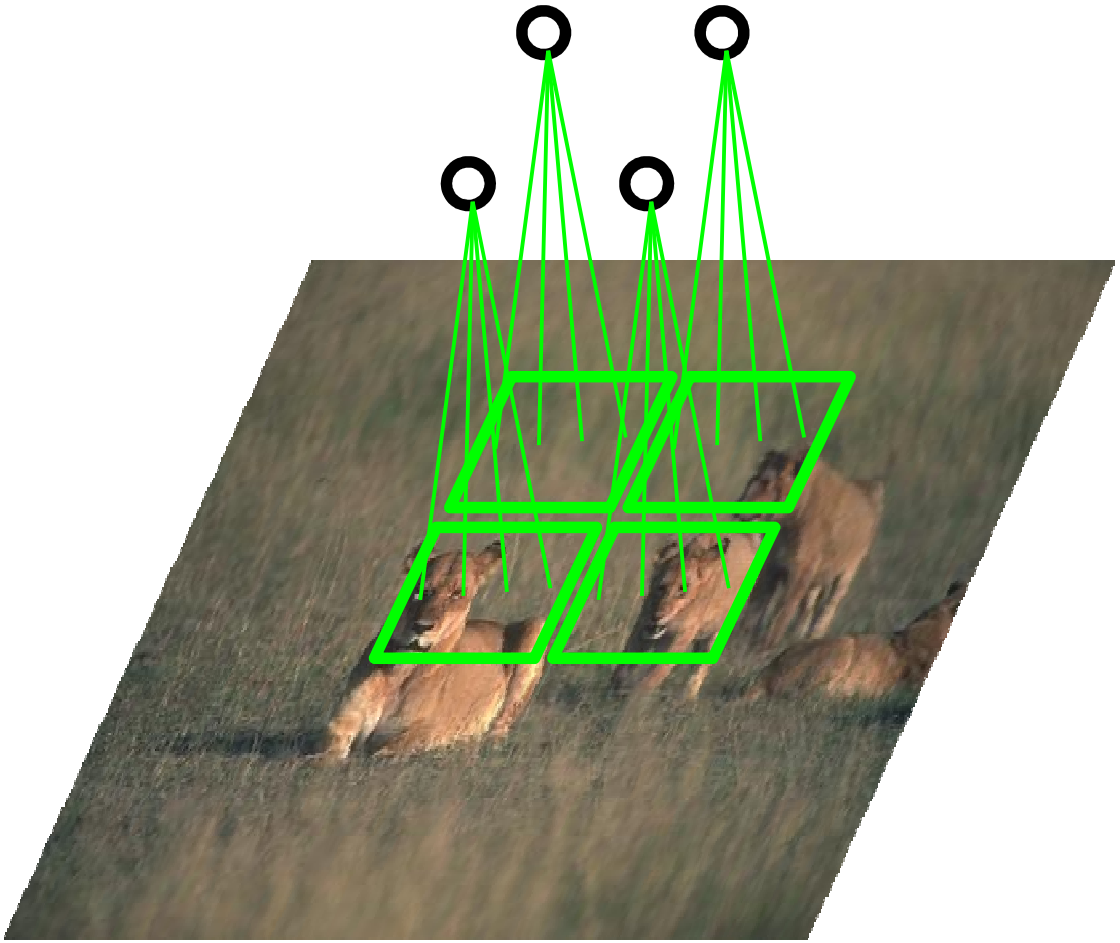


Pick patches at random locations for training

Ranzato

From Patches to High-Resolution Images

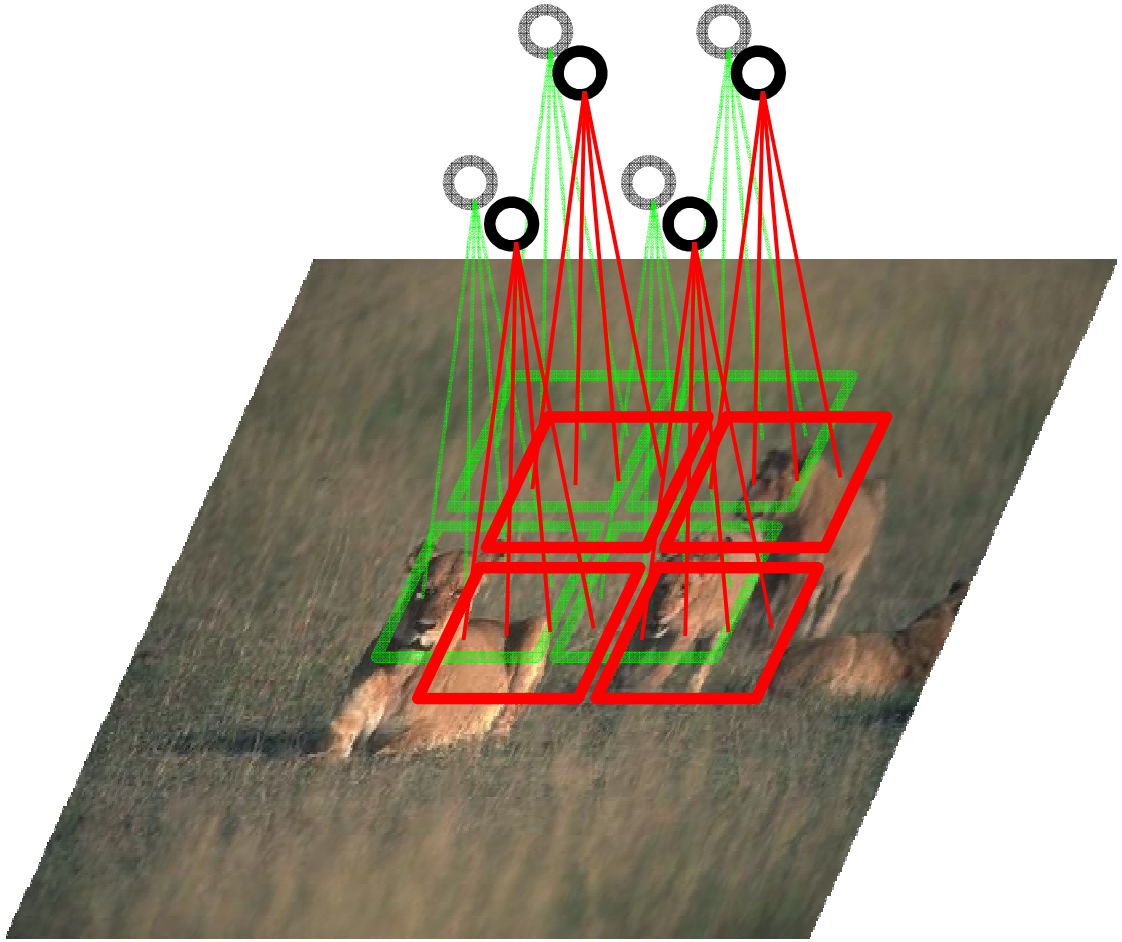
IDEA: have one subset of filters applied to these locations,



Ranzato

From Patches to High-Resolution Images

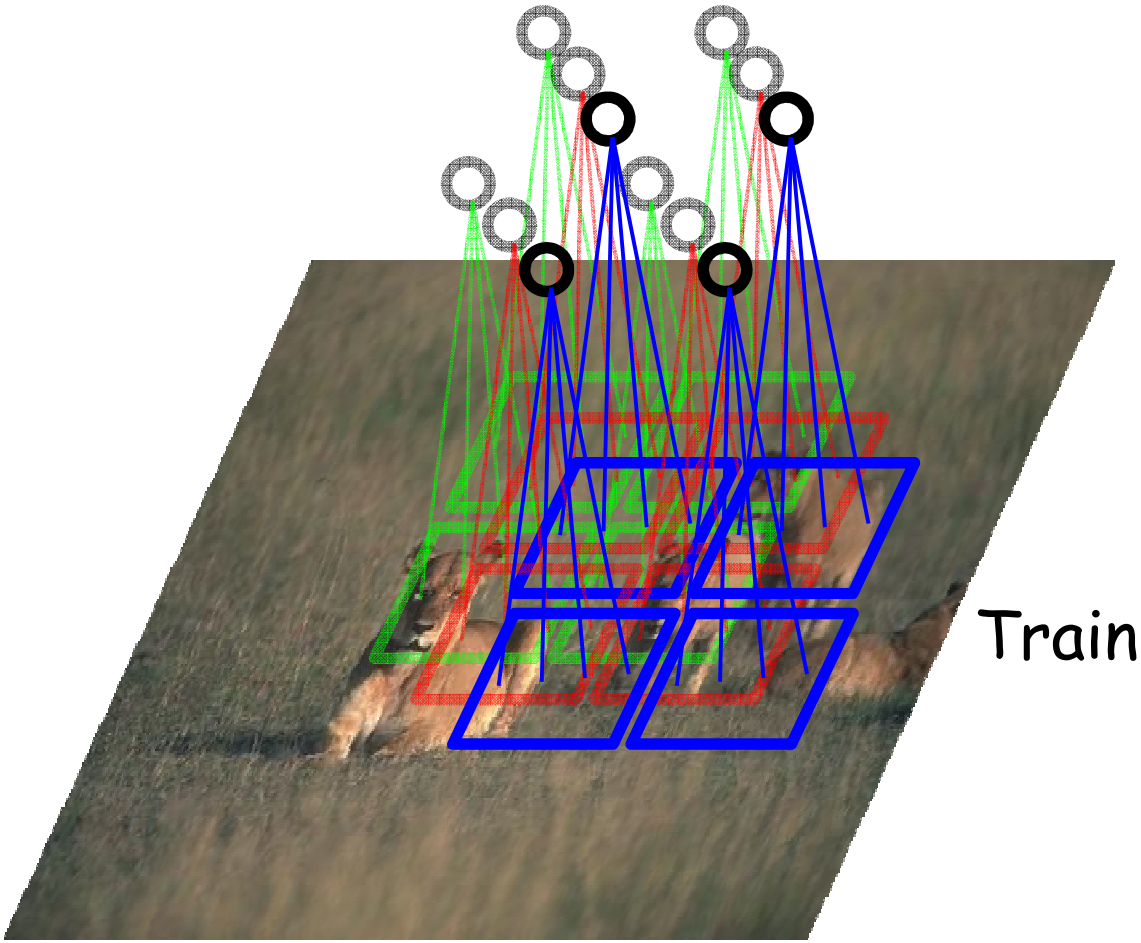
IDEA: have one subset of filters applied to these locations, another subset to these locations



Ranzato

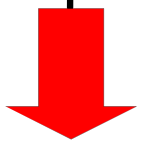
From Patches to High-Resolution Images

IDEA: have one subset of filters applied to these locations, another subset to these locations, etc.



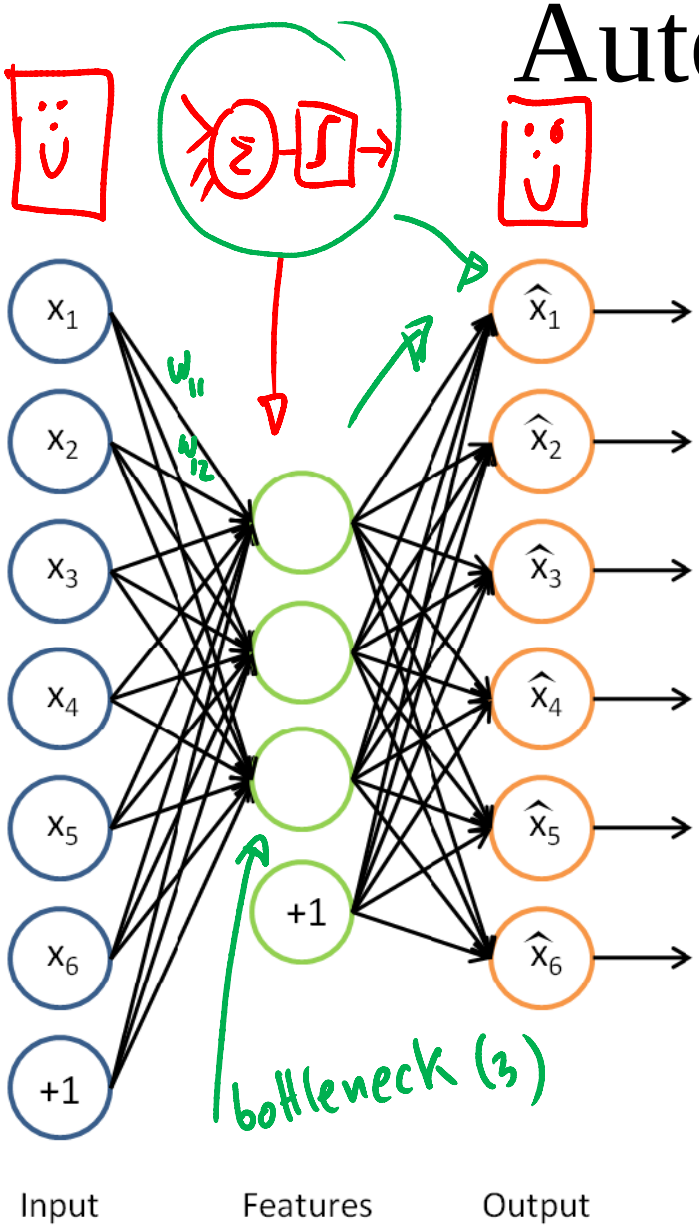
Gregor LeCun arXiv 2010
Ranzato, Mnih, Hinton NIPS 2010

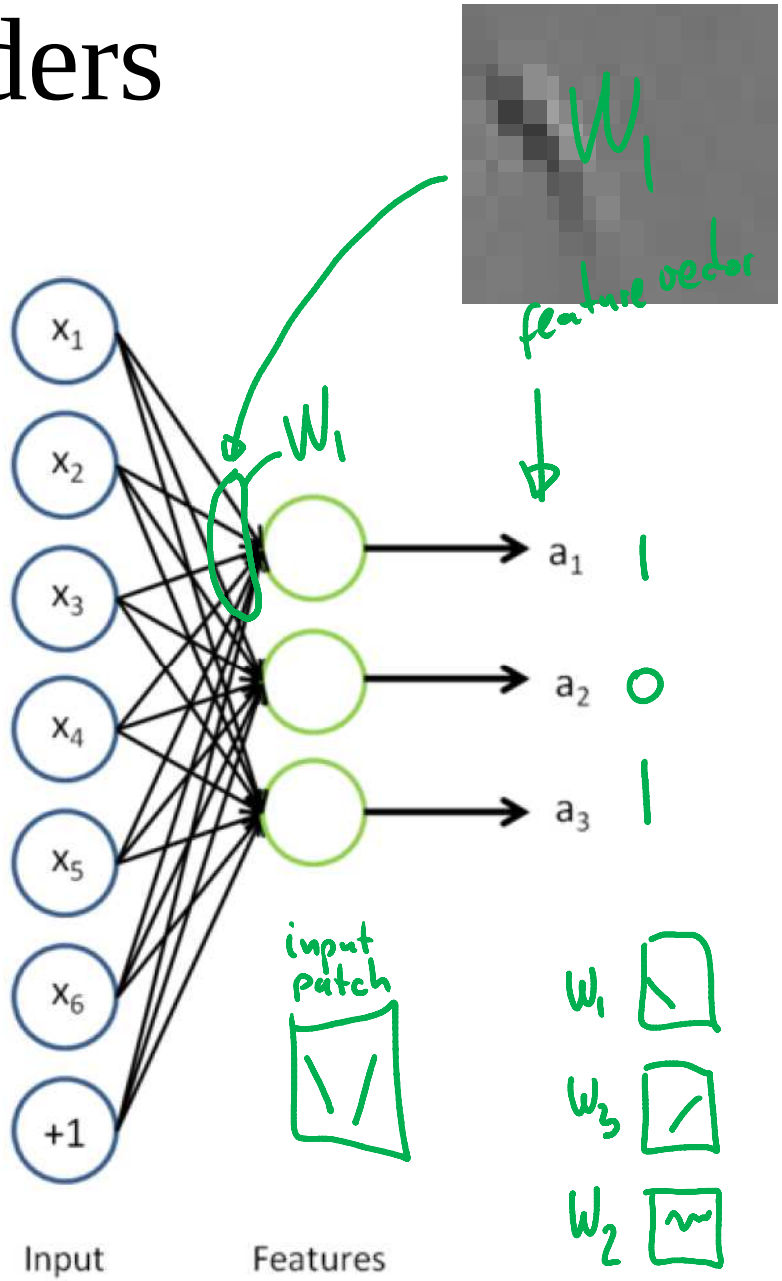
Train jointly all parameters.



No block artifacts
Reduced redundancy

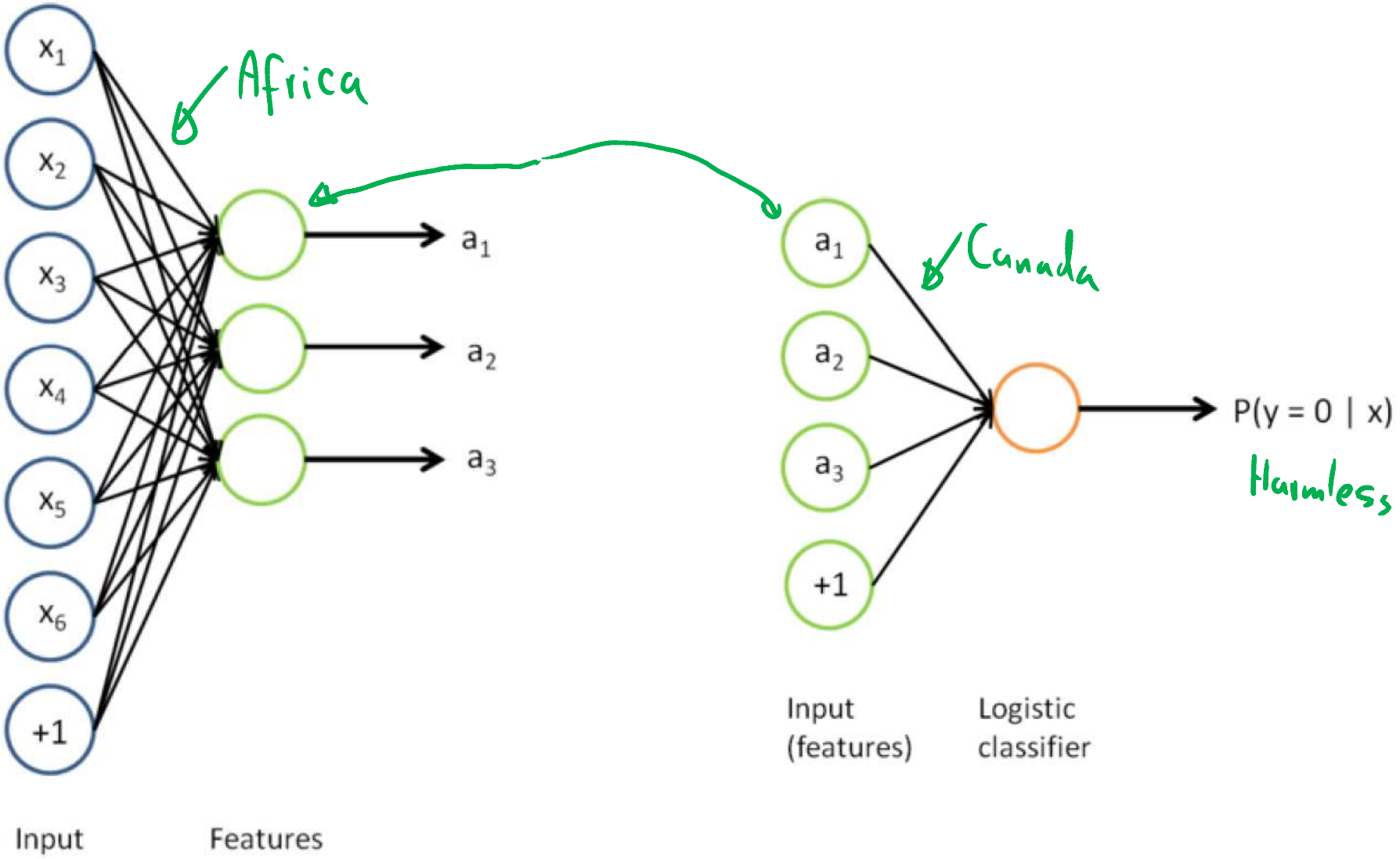
Autoencoders

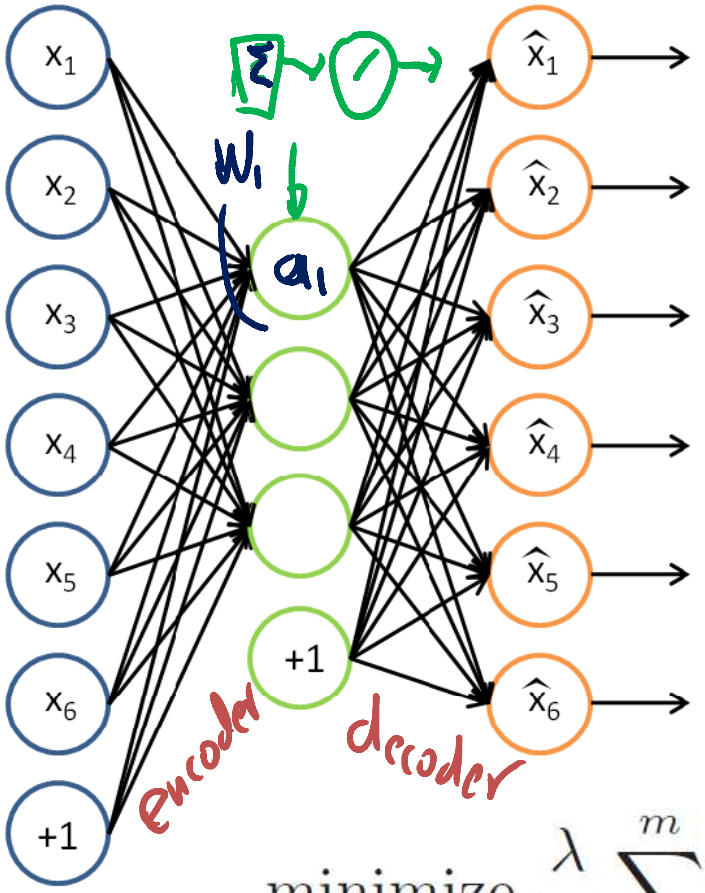


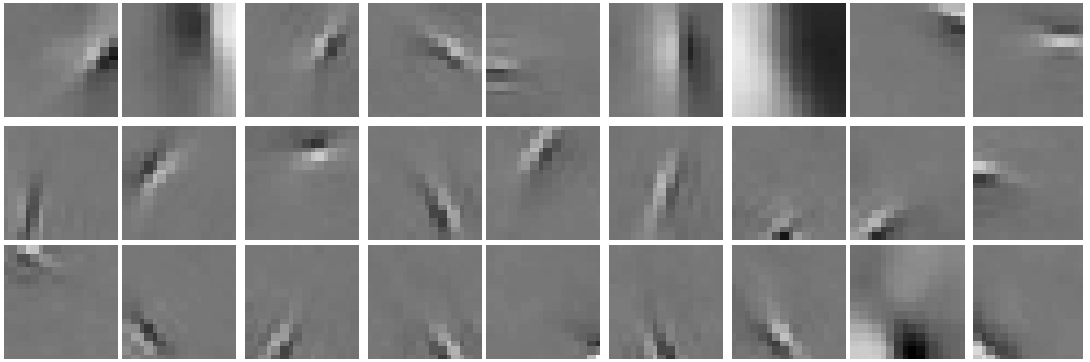


[Andrew Ng, MLSS 2012]

Autoencoders
self-taught learning / transfer learning







$$\underset{1\times 1}{a_1} = \underset{1\times 7}{W_1}\,\underset{7\times 1}{x^{(i)}} \qquad a W_1^T = \hat{x}$$

$$W_1 \in \mathbb{R}^{1\times 7} \qquad x^{(i)} \in \mathbb{R}^{7\times 1}$$

$$\underset{W}{\text{minimize}}\ \frac{\lambda}{m}\sum_{i=1}^{m}\|W^T\underbrace{[Wx^{(i)}]}_{a^{(i)}} - x^{(i)}\|_2^2 + \sum_{i=1}^{m}\sum_{j=1}^{k} g(W_j x^{(i)})$$

$\hat{x}^{(i)}$

$$\underset{W,b,c}{\text{minimize}}\ \frac{\lambda}{m}\sum_{i=1}^{m}\|\underbrace{\sigma(W^T\sigma(Wx^{(i)}+b)+c)}_{\hat{x}^{(i)}} - x^{(i)}\|_2^2 + \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{k} g(W_j x^{(i)})}_{\text{regularizer}}$$

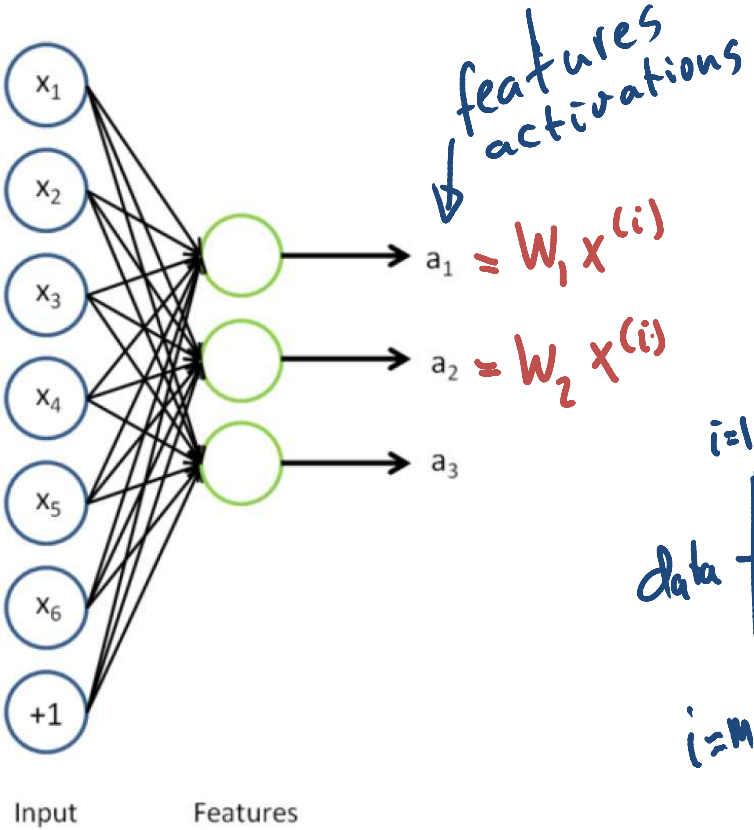
e.g., smooth L_1 penalty: $g(.) := \log(\cosh(.))$

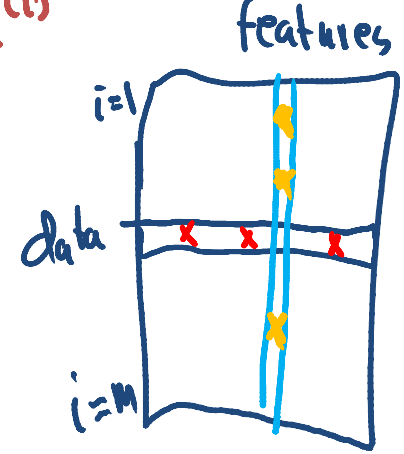
Activation sparseness regularizer

$$\underset{W}{\text{minimize}} \frac{\lambda}{m} \sum_{i=1}^{m} \|W^T W x^{(i)} - x^{(i)}\|_2^2 + \sum_{i=1}^{m} \sum_{j=1}^{k} g(W_j x^{(i)})$$

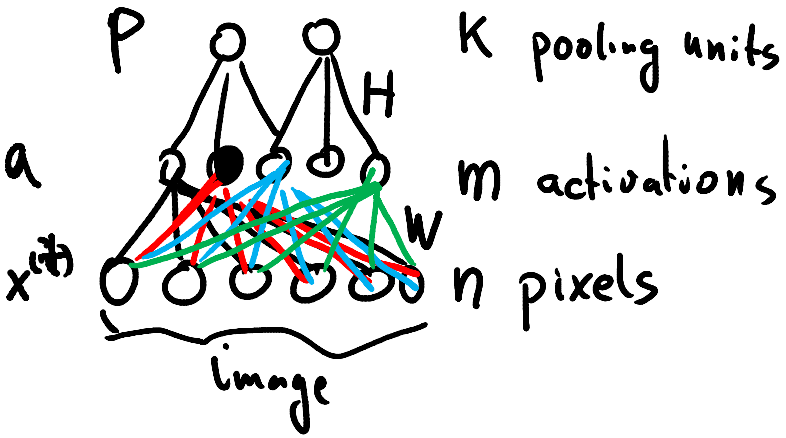
$$+ \sum_{i=1}^{m} \sum_{j=1}^{k} |W_j x^{(i)}|$$

$$+ \sum_{j=1}^{k} \sum_{i=1}^{m} |W_j x^{(i)}|$$





Autoencoders with pooling (Simple and complex cells)



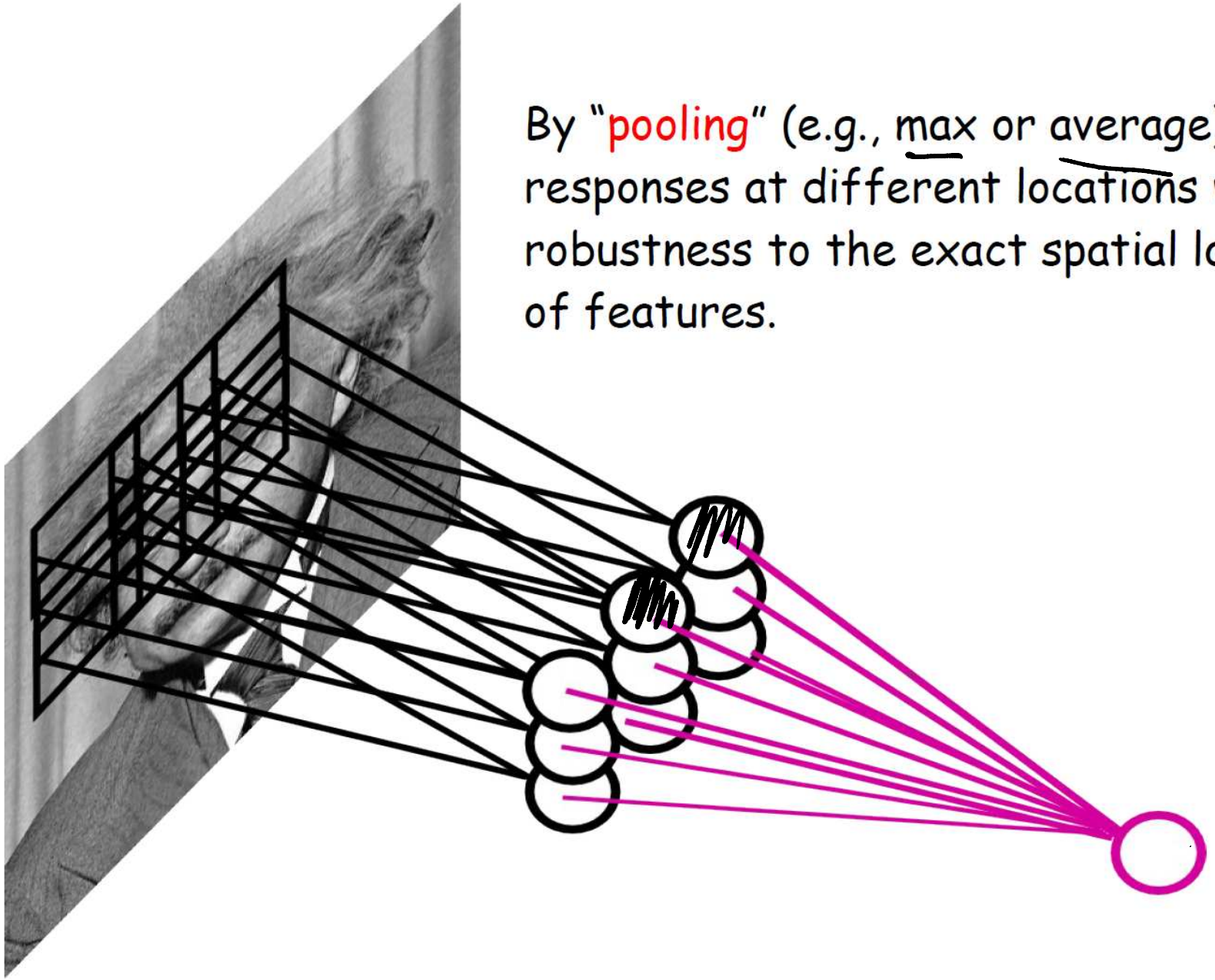
① $$a_k = \sum_{j=1}^{n} W_{kj} x_j^{(t)}$$

$$\underline{a} = W x^{(t)}$$

② $$P_i = \sqrt{\sum_{k=1}^{m} H_{ik} a_k^2}$$

$$P_i = \sqrt{\underline{H}_i a^2}$$

POOLING

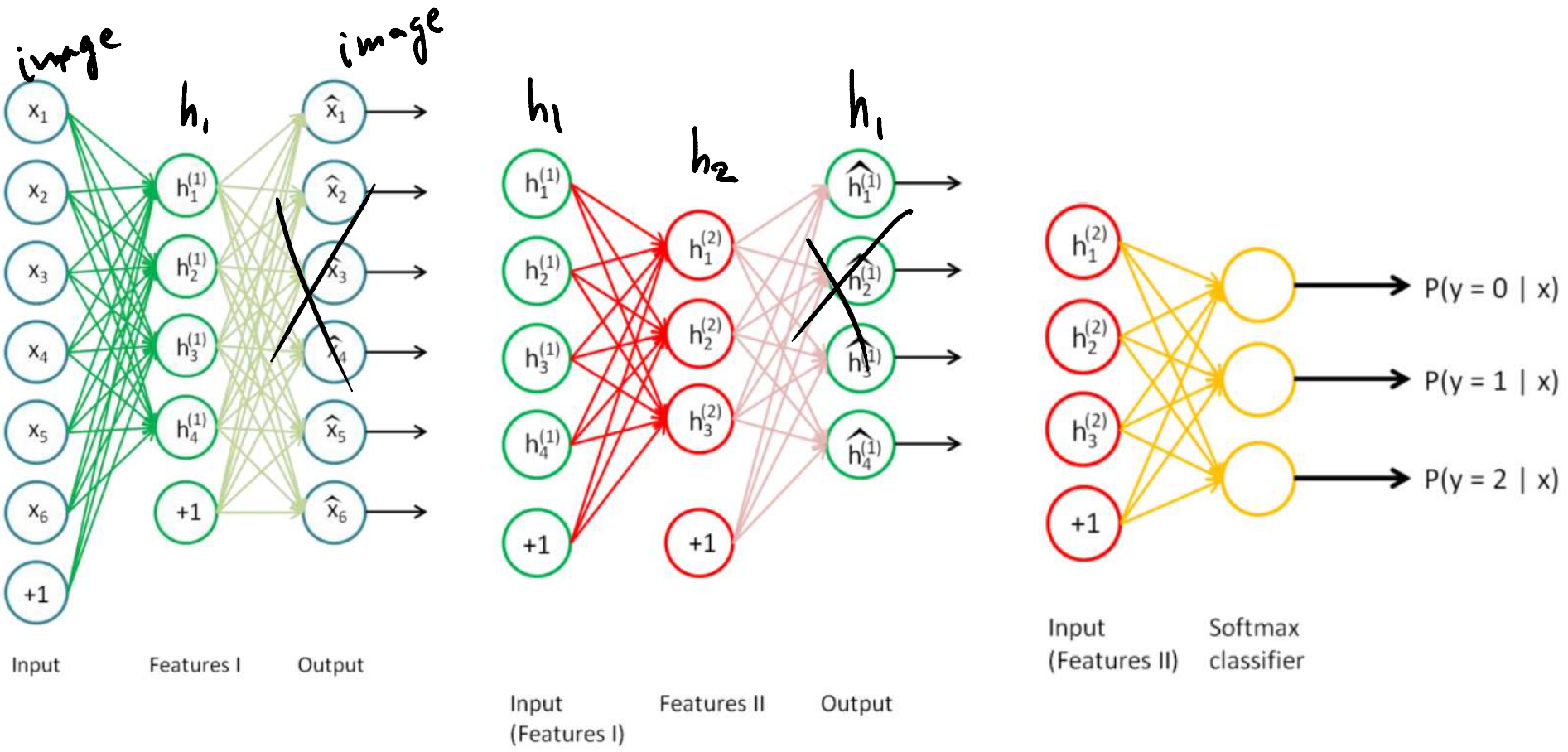


By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

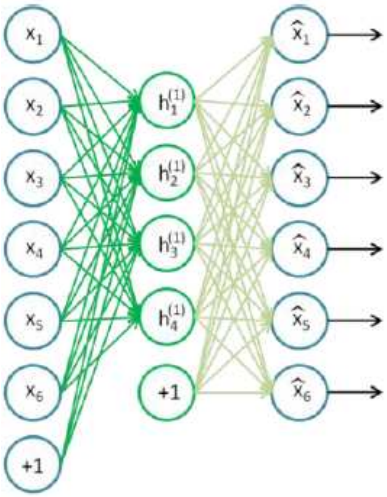
More layers:

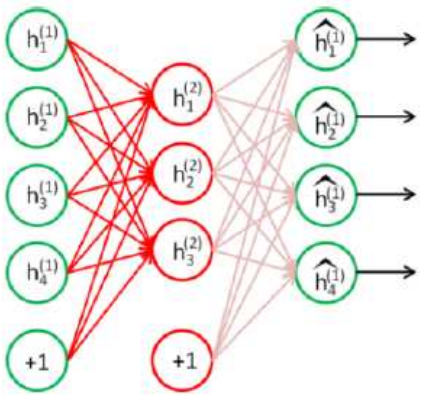
deep learning

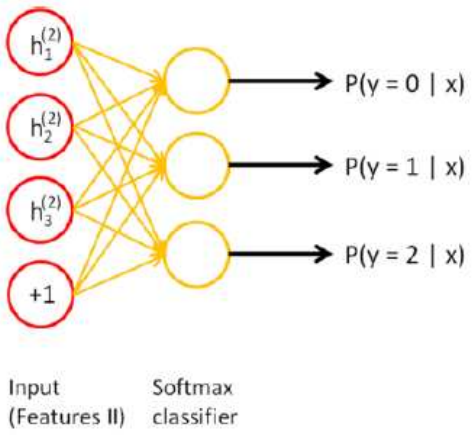
Greedy layer-wise training

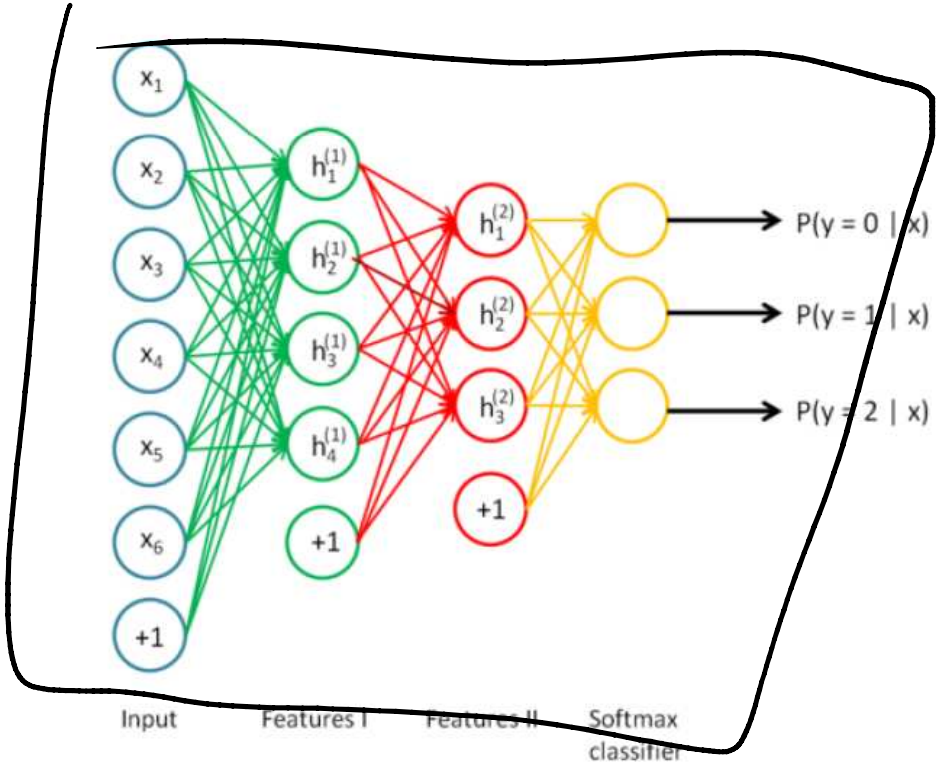


[Andrew Ng, MLSS 2012]


Input
Features I
Output

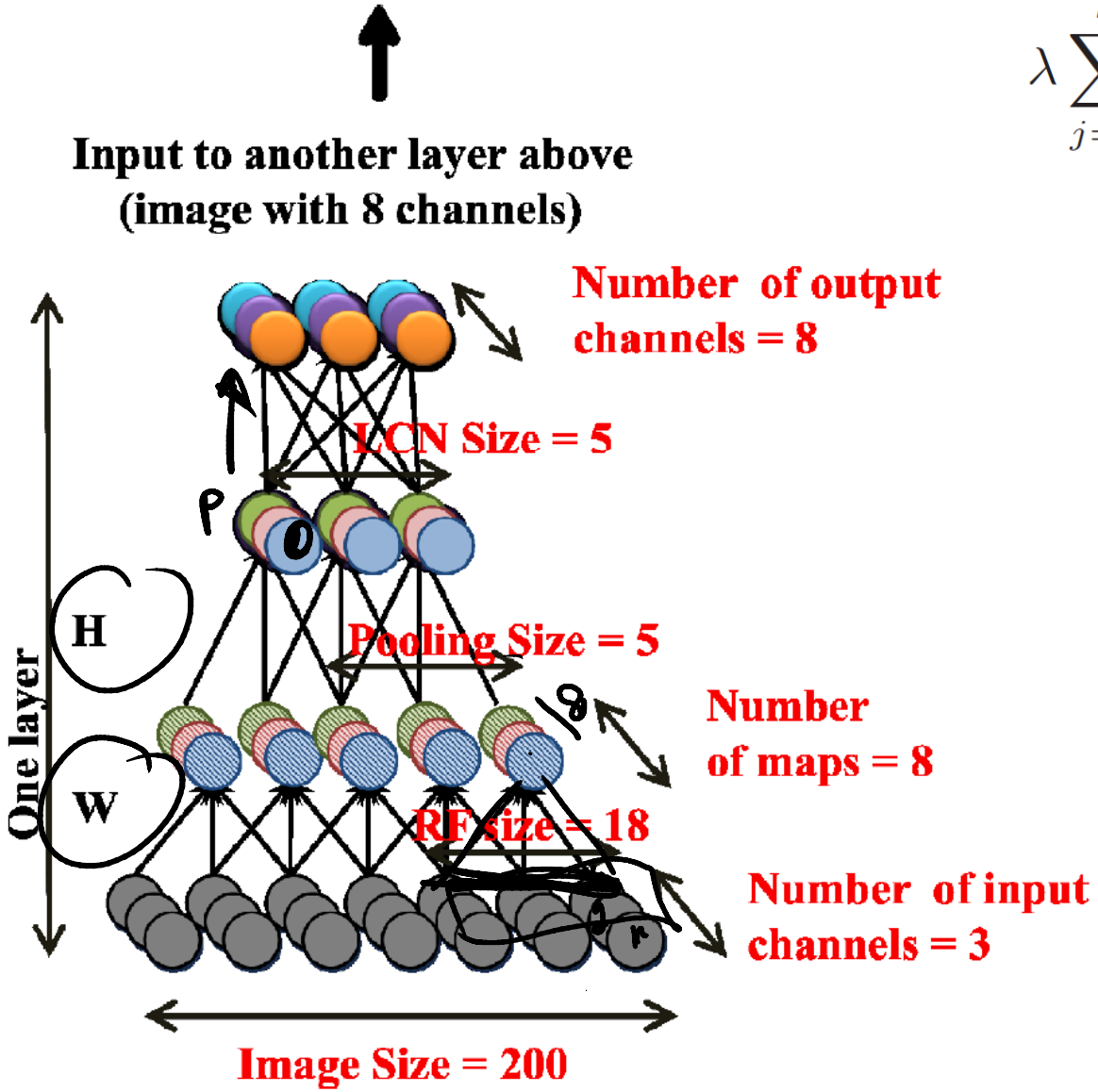

Input
(Features I)
Features II
Output


P(y = 0 | x)
P(y = 1 | x)
P(y = 2 | x)
Input
(Features II)
Softmax
classifier


P(y = 0 | x)
P(y = 1 | x)
P(y = 2 | x)
Input
Features I
Features II
Softmax
classifier

Google autoencoder

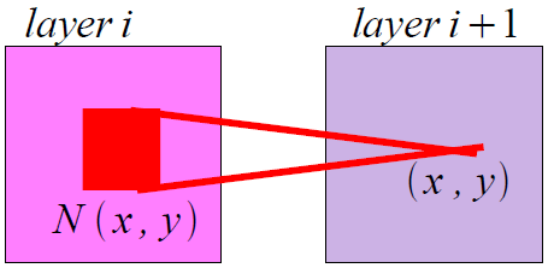
$$\underset{W_1,W_2}{\text{minimize}} \sum_{i=1}^{m} \left(\left\| W_2 W_1^T x^{(i)} - x^{(i)} \right\|_2^2 + \lambda \sum_{j=1}^{k} \sqrt{\epsilon + H_j (W_1^T x^{(i)})^2} \right).$$



POOLING & LCN

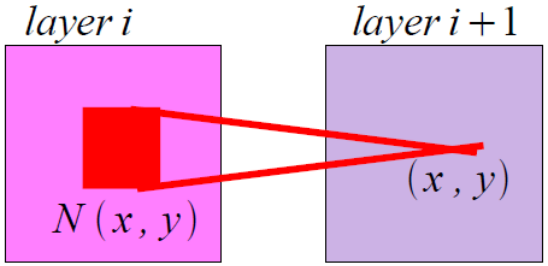
Over the years, some new modules have proven to be very effective when plugged into conv-nets:

- L2 Pooling



$$h_{i+1,x,y} = \sqrt{\sum_{(j,k) \in N(x,y)} h^2_{i,j,k}}$$

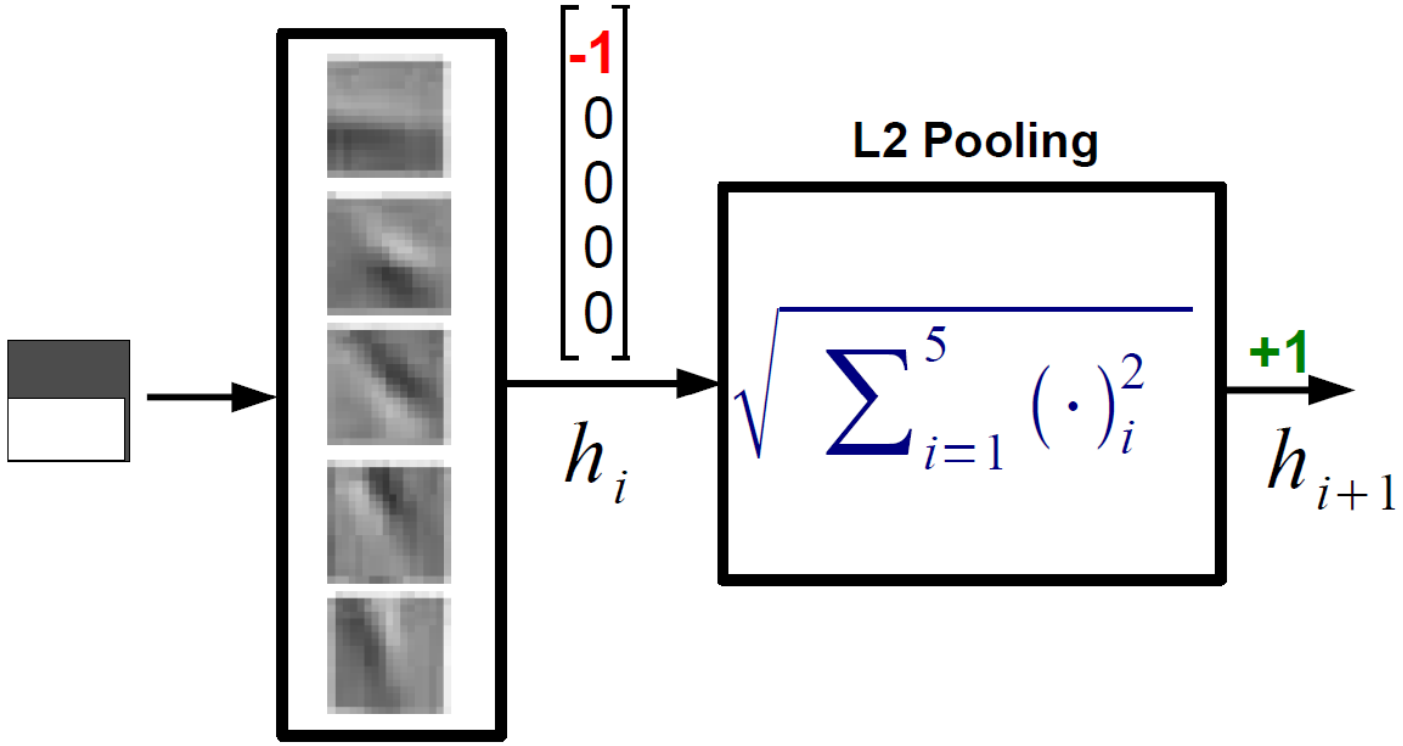
- Local Contrast Normalization



$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$

Jarrett et al. "What is the best multi-stage architecture for object recognition?" ICCV 2009

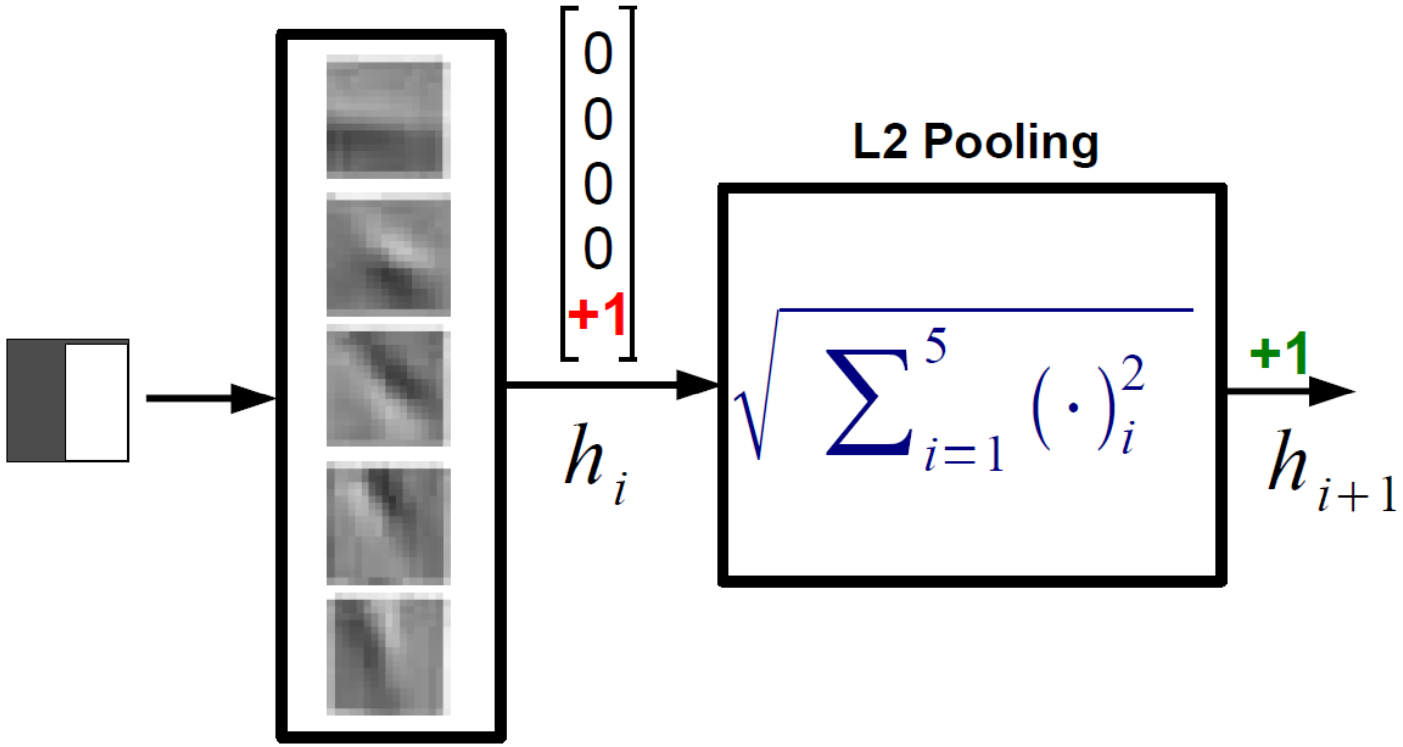
L2 POOLING



Kavukguoglu et al. "Learning invariant features ..." CVPR 2009

Ranzato

L2 POOLING

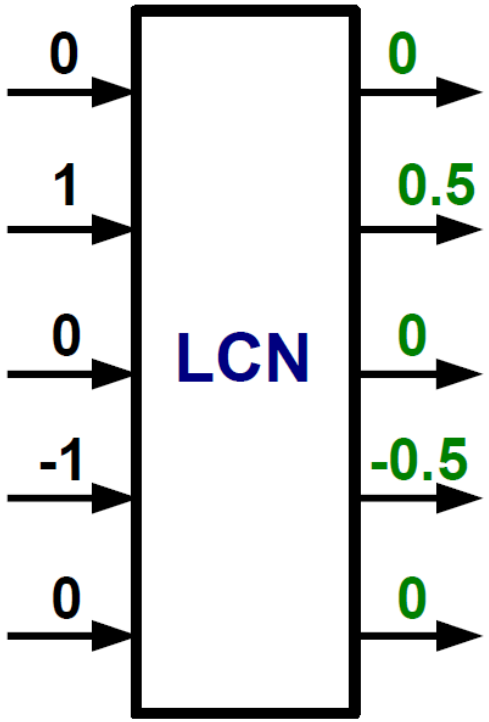


Kavukguoglu et al. "Learning invariant features ..." CVPR 2009

L2 Pooling helps learning
representations more robust to local distortions!

LOCAL CONTRAST NORMALIZATION

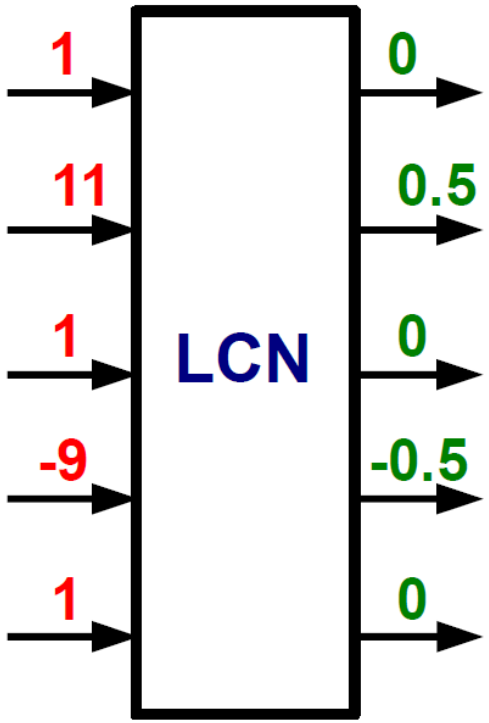
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



Ranzato

LOCAL CONTRAST NORMALIZATION

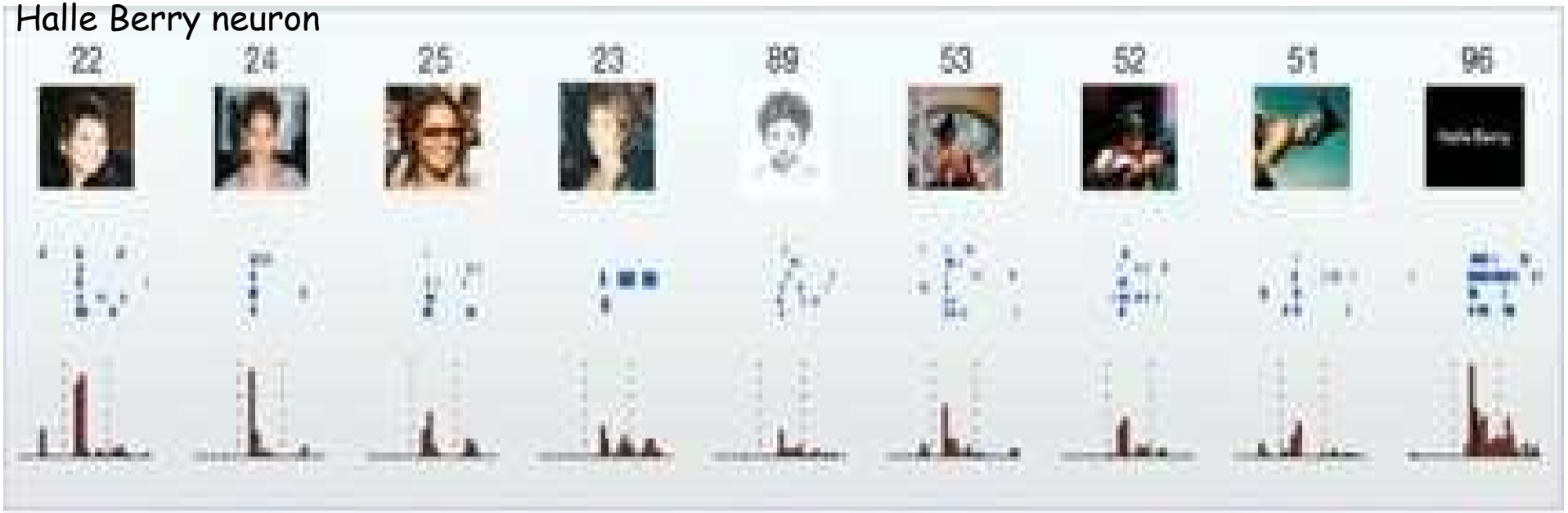
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



L2 Pooling & Local Contrast Normalization
help learning more invariant representations!

from Quiroga et al. "Invariant visual representation by single neurons in the human brain" Nature 2005

"Here we report on a remarkable subset of MTL neurons that are selectively activated by strikingly different pictures of given individuals, landmarks or objects and in some cases even by letter strings with their names."



Unsupervised Learning With 1B Parameters

DATA: 10M youtube (unlabeled) frames of size 200x200.

Unsupervised Learning With 1B Parameters

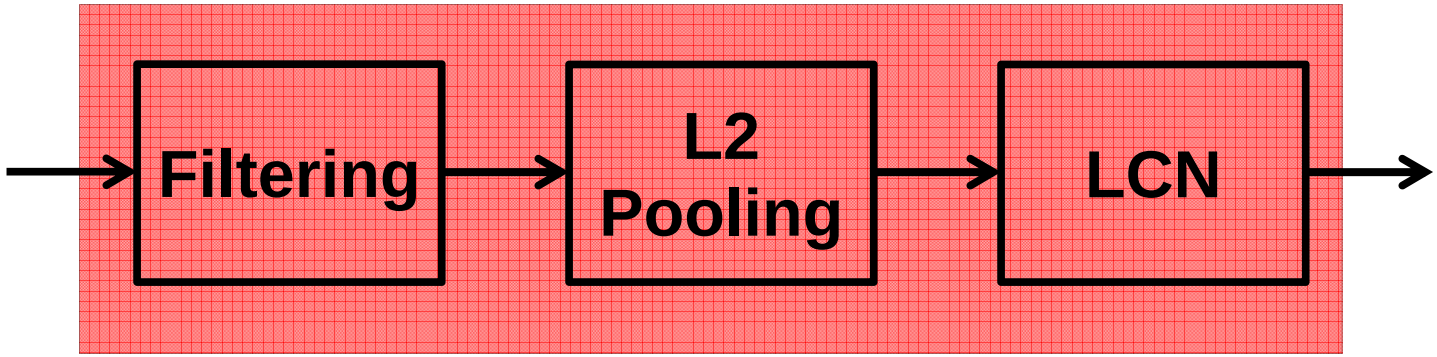
Deep Net:

- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
    - 18x18 filters
    - 8 filters at each location
    - L2 pooling and LCN over 5x5 neighborhoods
- training jointly the three layers by:
    - reconstructing the input of each layer
    - sparsity on the code

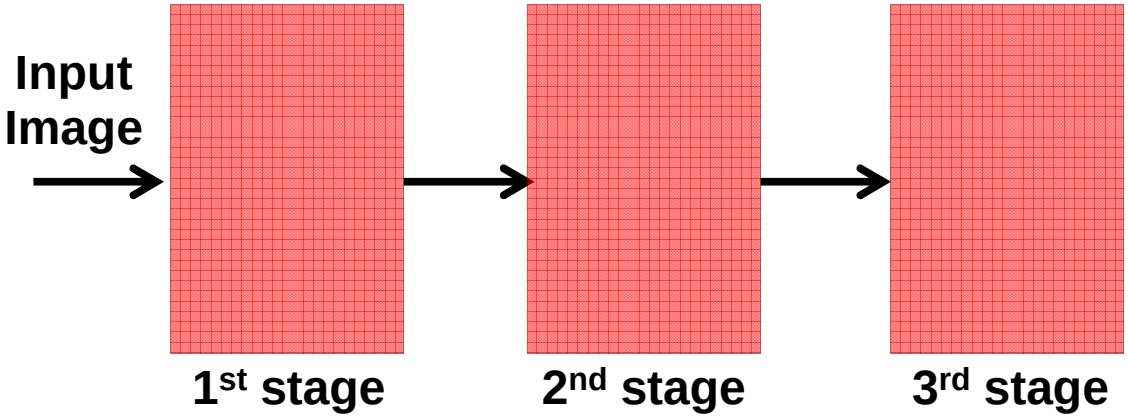
Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012

Unsupervised Learning With 1B Parameters

One stage (zoom)



Whole system



Ranzato

Validating Unsupervised Learning

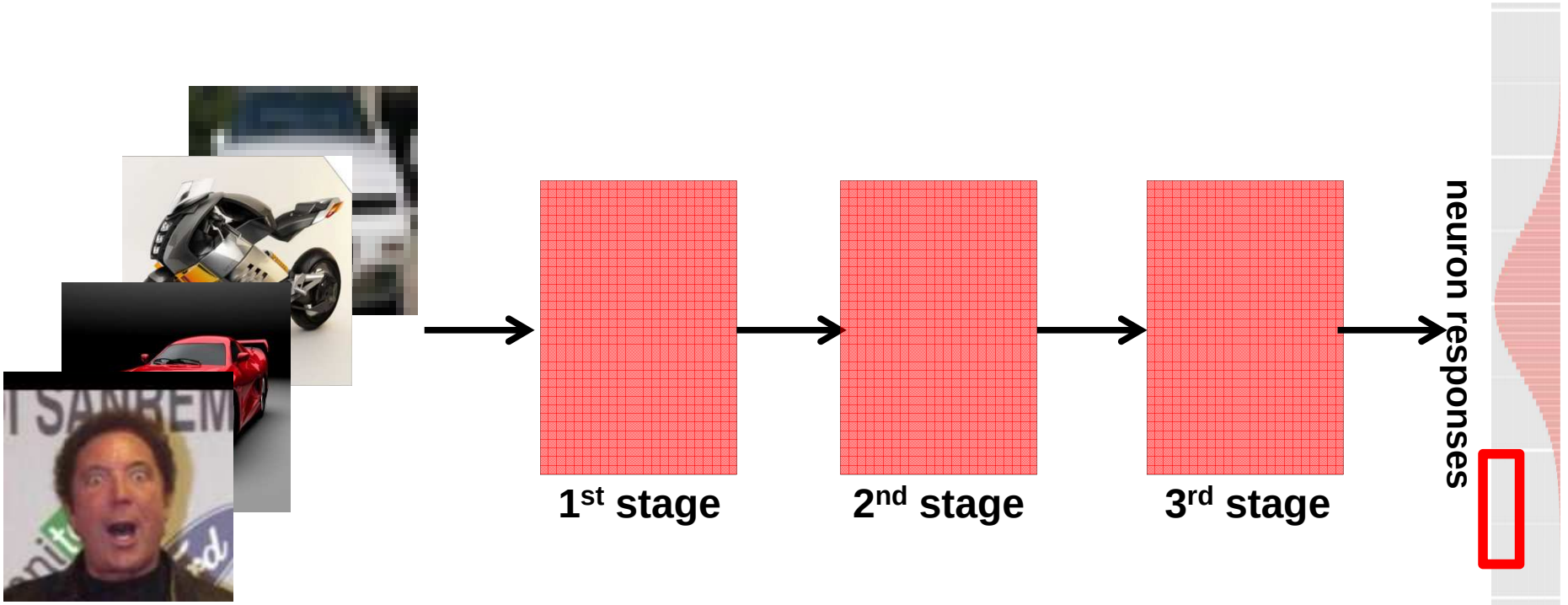
The network has seen lots of objects during training, but without any label.

Q.: how can we validate unsupervised learning?

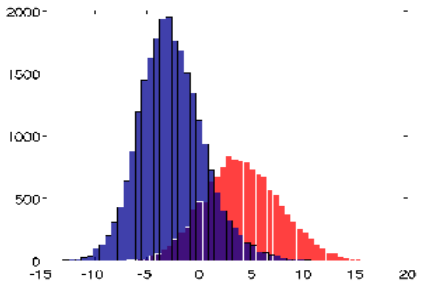
Q.: Did the network form any high-level representation?
E.g., does it have any neuron responding for faces?

- build validation set with 50% faces, 50% random images
- study properties of neurons

Validating Unsupervised Learning



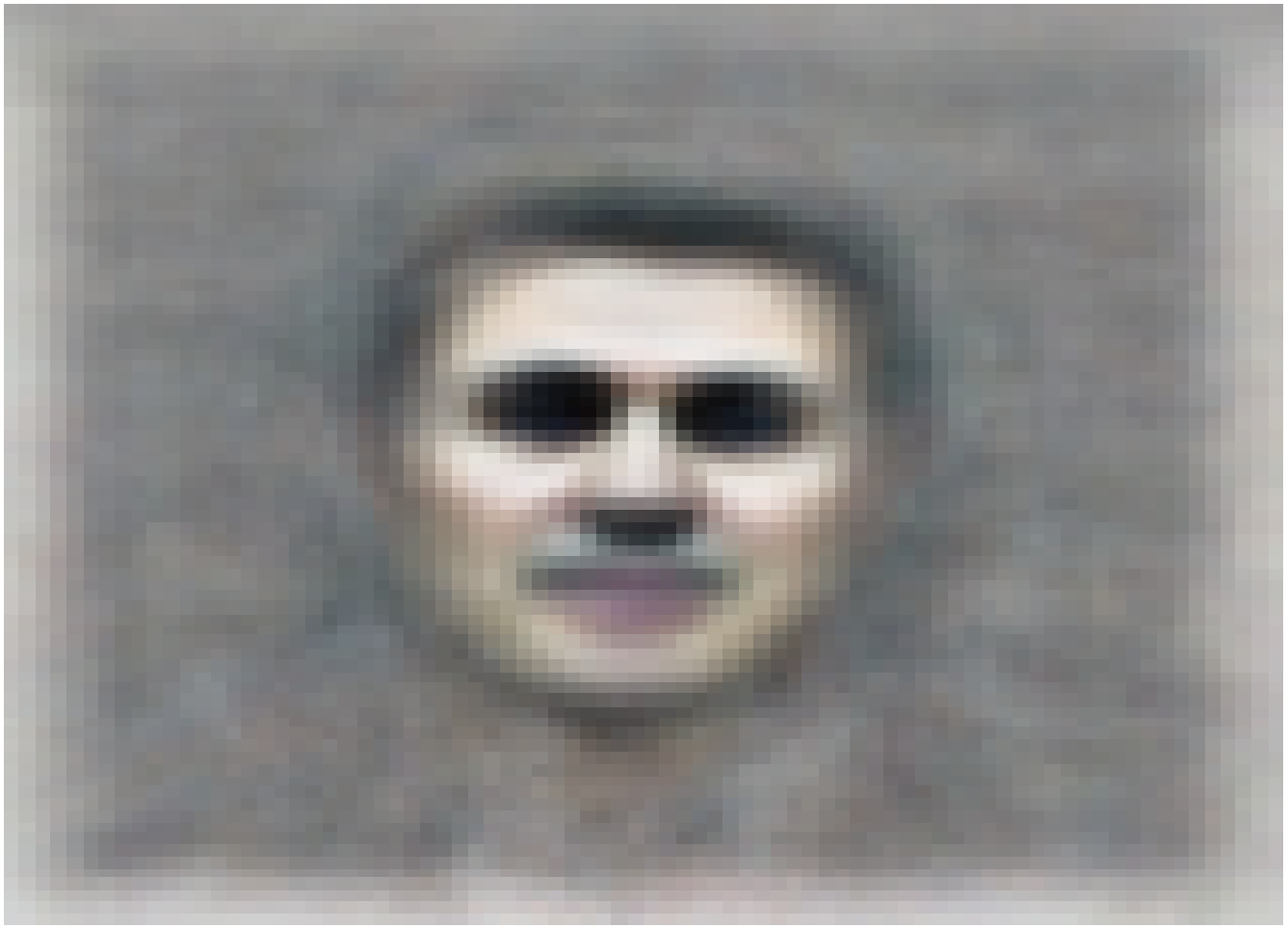
Face / No face



Ranzato

Top Images For Best Face Neuron

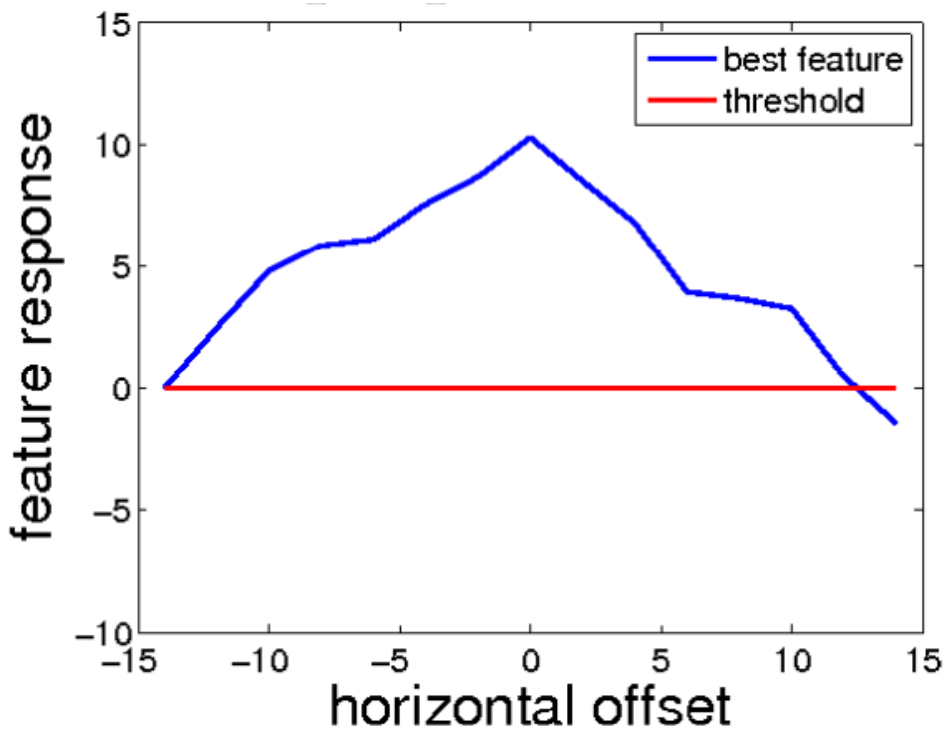
Best Input For Face Neuron

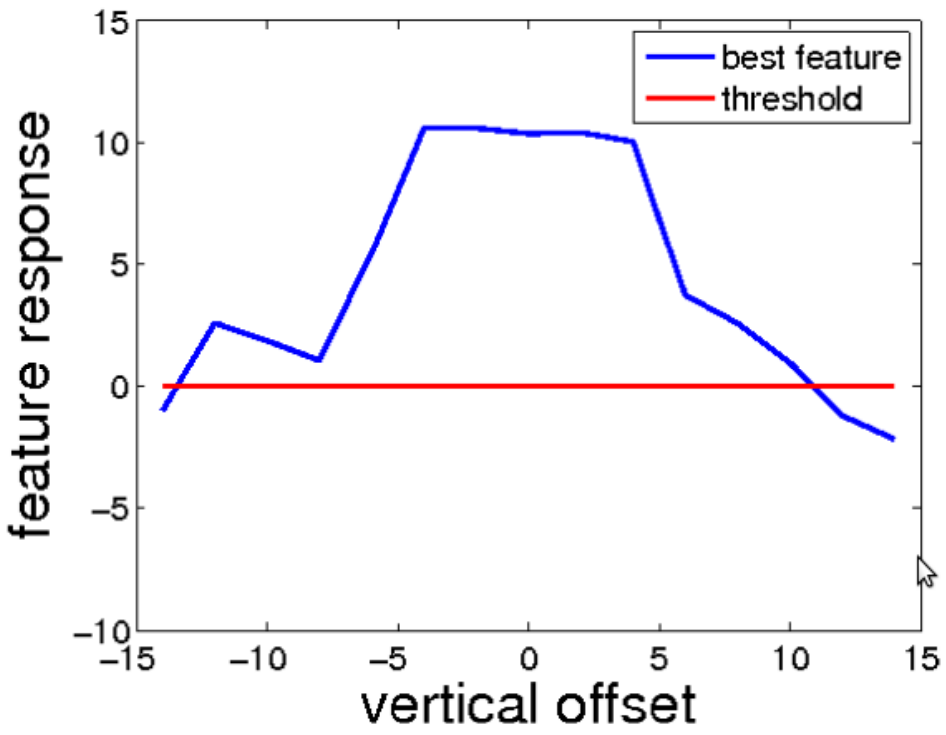


$$x^* = \arg\min_x f(x; W, H), \quad \text{subject to } ||x||_2 = 1$$

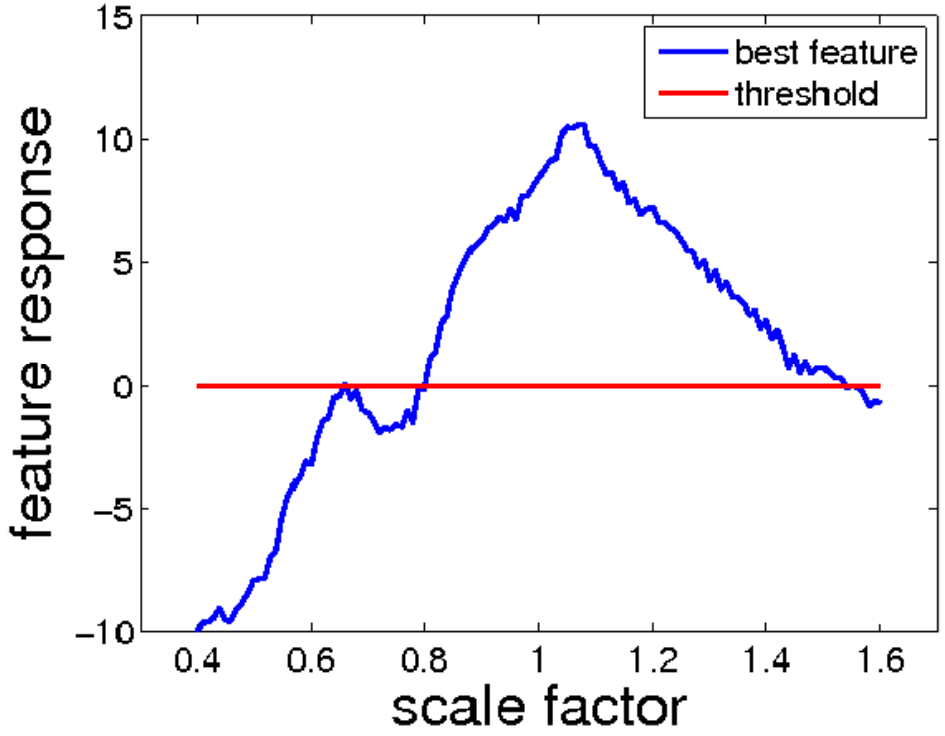
Ranzato

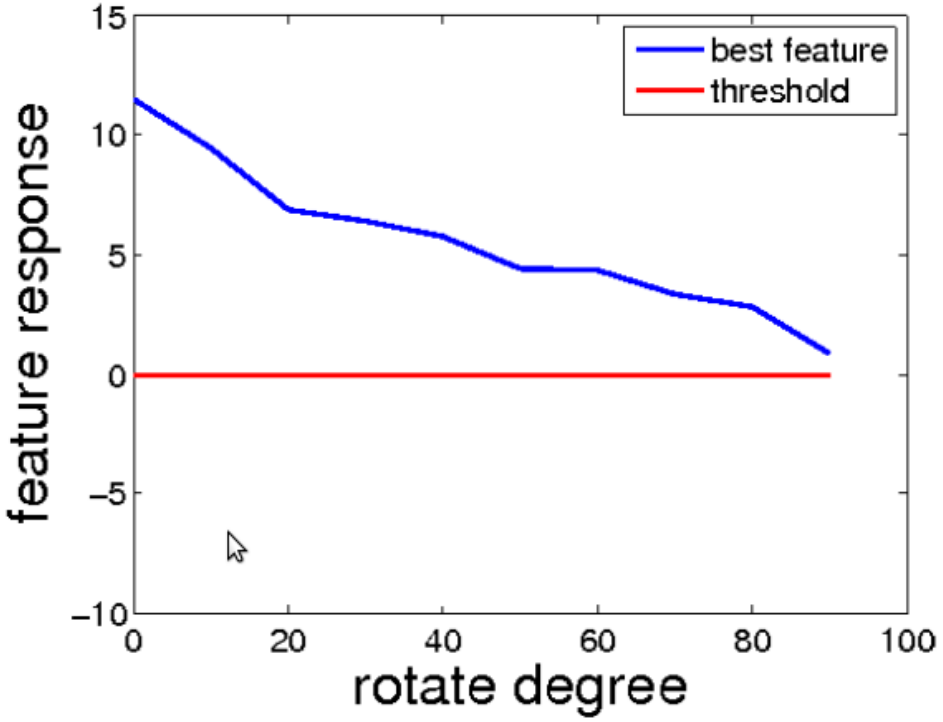
Invariance Properties





Invariance Properties





Cat Neuron

Top Images for Cat Neuron

Object Recognition on ImageNet

IMAGENET v.2011 (16M images, 20K categories)

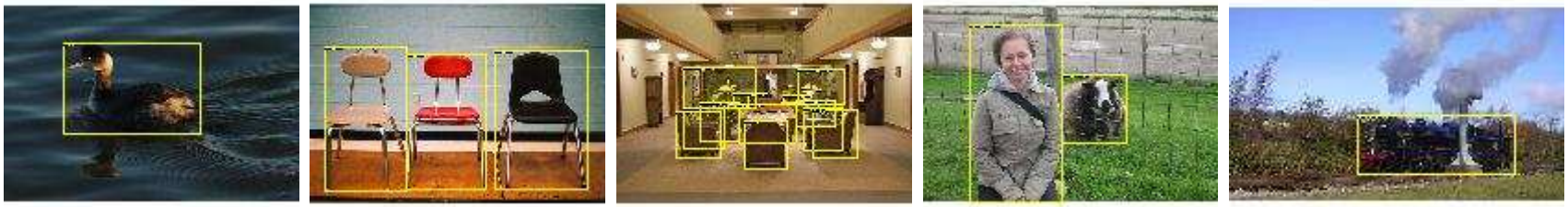
METHOD	ACCURACY %
Weston & Bengio 2011	9.3
Linear Classifier on deep features	13.1
Deep Net (from random)	13.6
Deep Net (from unsup.)	**15.8**

Dropout

Visual Object Classes Challenge 2012 (VOC2012)

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

• **Classification**: For 20 classes, predict presence/absence of an example of that class in the test image.
• **Detection**: Predict the bounding box and label of each object from the 20 target classes in the test image.



Person: person
Animal: bird, cat, cow, dog, horse, sheep
Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

Task 1 (classification)

Team name	Error (5 guesses)	Description
SuperVision	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	0.16422	Using only supplied training data
ISI	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	0.26646	Naive sum of scores from classifiers using each FV.
ISI	0.26952	Naive sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
OXFORD_VGG	0.26979	Mixed selection from High-Level SVM scores and Baseline Scores, decision is performed by looking at the validation performance
XRCE/INRIA	0.27058	
OXFORD_VGG	0.27079	High-Level SVM over Fine Level Classification score, DPM score and Baseline Classification scores (Fisher Vectors over Dense SIFT and Color Statistics)
OXFORD_VGG	0.27302	Baseline: SVM trained on Fisher Vectors over Dense SIFT and Color Statistics
University of Amsterdam	0.29576	See text above
XRCE/INRIA	0.33419	
LEAR-XRCE	0.34464	Trained on ILSVRC'12 - using a mixture of NCM classifiers
LEAR-XRCE	0.36184	Trained on ILSVRC'12 - using NCM
LEAR-XRCE	0.38006	Trained on ILSVRC'10 - using a mixture of NCM classifiers
LEAR-XRCE	0.41048	Trained on ILSVRC'10 - using NCM

Task 2 (localization)

Team name	Error (5 guesses)	Description
SuperVision	0.335463	Using extra training data for classification from ImageNet Fall 2011 release
SuperVision	0.341905	Using only supplied training data
OXFORD_VGG	0.500342	Re-ranked DPM detection over Mixed selection from High-Level SVM scores and Baseline Scores, decision is performed by looking at the validation performance
OXFORD_VGG	0.50139	Re-ranked DPM detection over High-Level SVM Scores
OXFORD_VGG	0.522189	Re-ranked DPM detection over High-Level SVM Scores - First bbox selection heuristic
OXFORD_VGG	0.529482	DPM detection over baseline classification scores
ISI	0.536474	We use the cascade object detection with deformable part models, restricting the sizes of bounding boxes.
ISI	0.536546	We use the cascade object detection with deformable part models, restricting the sizes of bounding boxes.

Averaging many models

- To win a machine learning competition (e.g. Netflix) you need to use many different types of model and then combine them to make predictions at test time.

- Decision trees are not very powerful models, but they are easy to fit to data and very fast at test time.
  - Averaging many decision trees works really well. Its called random forests. Kinect uses this.
  - We make the individual trees different by giving them different training sets. That's called bagging

Hinton

Two ways to average models

- Mixture: We can combine models by taking the arithmetic means of their output probabilities:

Model A:	.3	.2	.5
Model B:	.1	.8	.1
Combined	.2	.5	.3

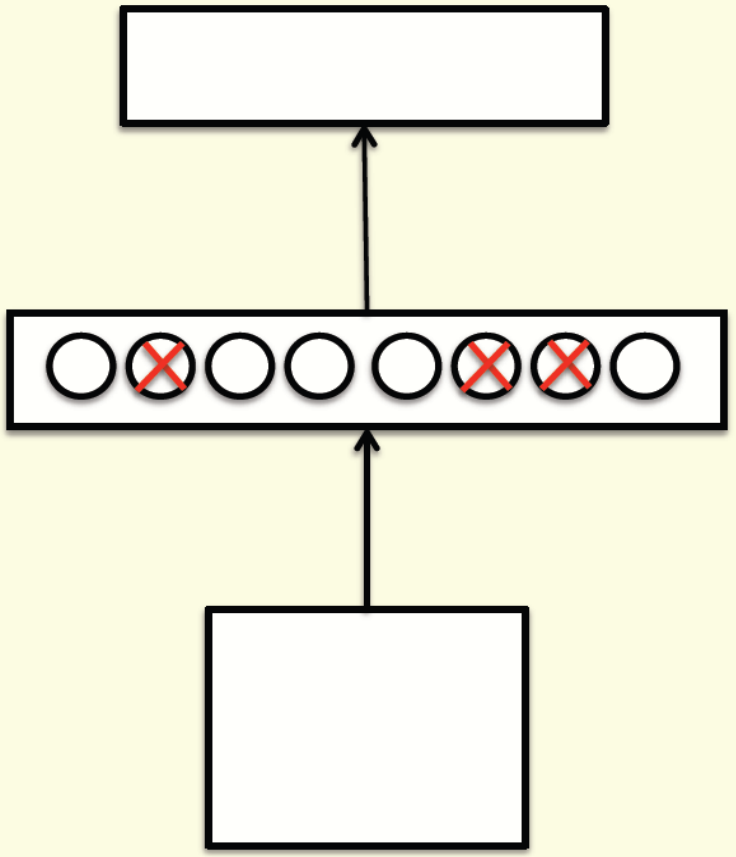
- Product: We can combine models by taking the geometric means of their output probabilities:

Model A:	.3	.2	.5	
Model B:	.1	.8	.1	
Combined	$\sqrt{.03}$	$\sqrt{.16}$	$\sqrt{.05}$	/sum

Hinton

Dropout: An efficient way to average many large neural nets.

- Consider a neural net with one hidden layer.
- Each time we present a training example, we randomly omit each hidden unit with probability 0.5.
- So we are randomly sampling from 2^H different architectures.
  - All architectures share weights.



Hinton

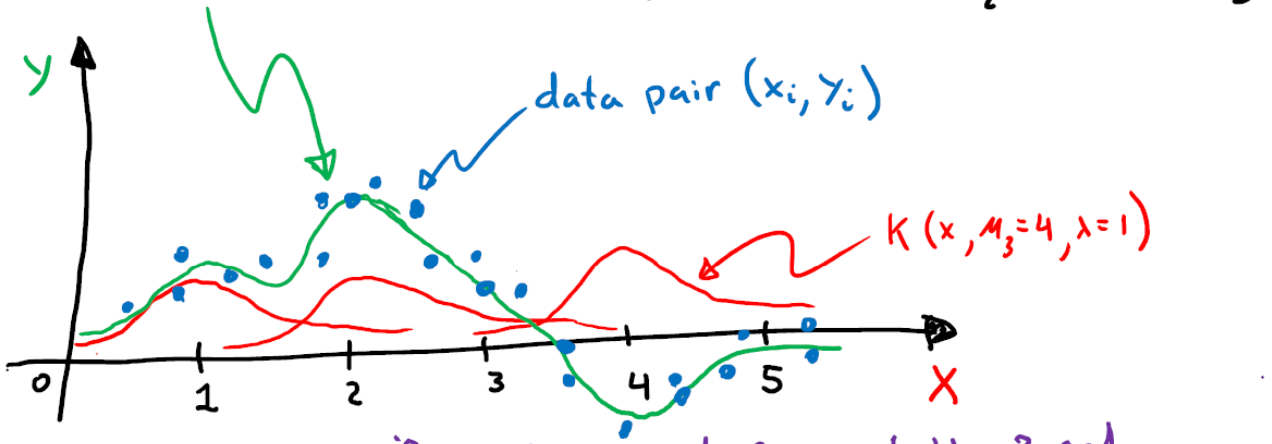
Dropout as a form of model averaging

- We sample from 2^H models. So only a few of the models ever get trained, and they only get one training example.
  - This is as extreme as bagging can get.
- The sharing of the weights means that every model is very strongly regularized.
  - It's a much better regularizer than L2 or L1 penalties that pull the weights towards zero.

Hinton

Why it works: Preventing co-adaptation

Example 1: $\hat{Y}(x) = e^{-\|x-1\|^2}\theta_1 + e^{-\|x-2\|^2}\theta_2 + e^{-\|x-4\|^2}\theta_3$

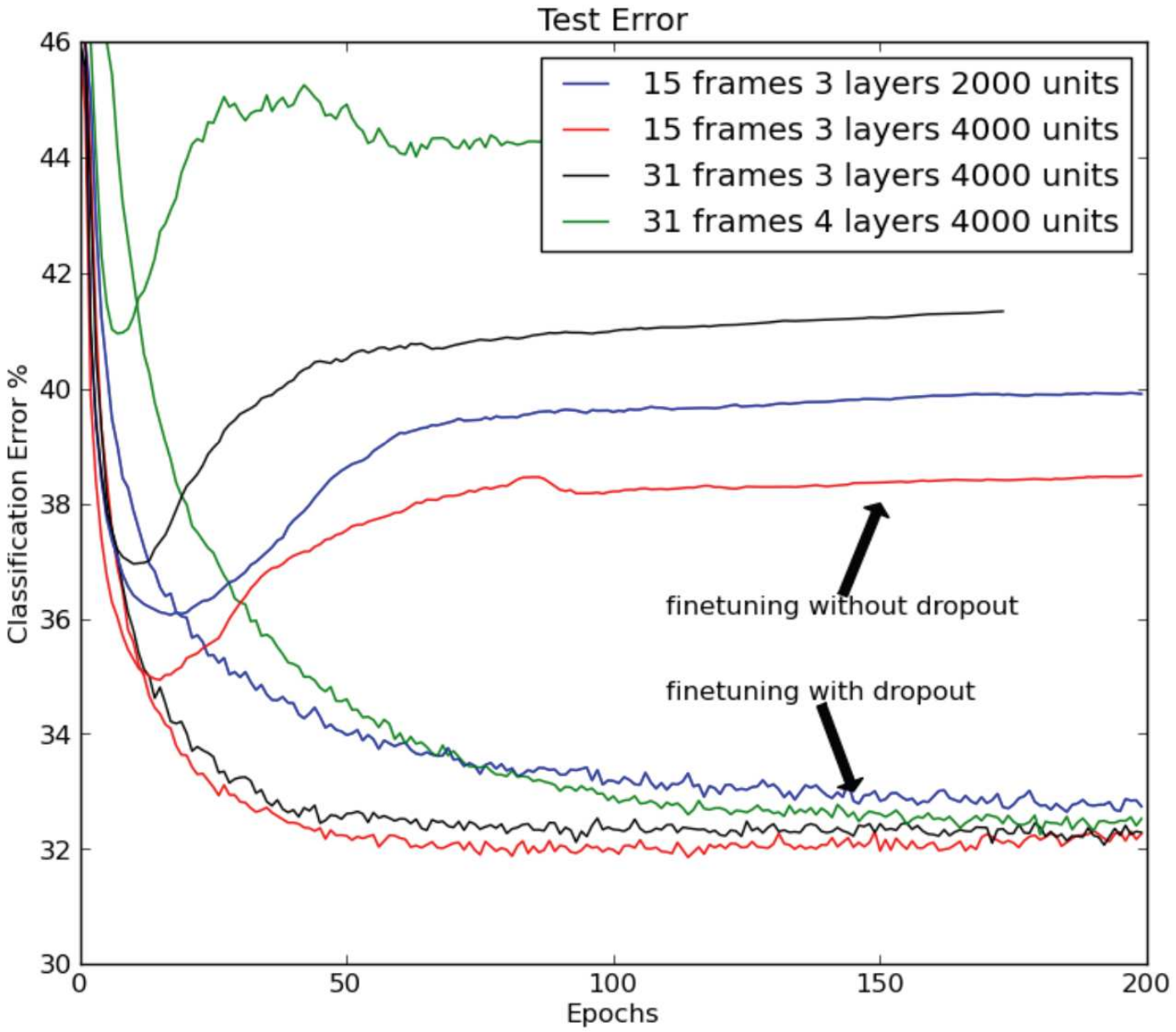


The green curve is a weighted sum of the 3 red curves

But what do we do at test time?

- We could sample many different architectures and take the geometric mean of their output distributions.
- It better to use all of the hidden units, but to halve their outgoing weights.
  - This exactly computes the geometric mean of the predictions of all 2^H models.

Experiments on TIMIT



Next lecture

In the next lecture, we will learn about MCMC and how to apply it to perform Bayesian inference in Neural Networks.