# Convolutional Neural Networks

Machine Perception

Otmar Hilliges

19 March 2020

[Some slides adopted from Karpathy, De Freitas, Serre]

Advanced Interactive Technologies

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Last week

Universal function approximator theorem

Backprop Algorithm

# This Week

Motivation for CNNs (with vague references to Neuro Science)

Convolutional Neural Networks for Perception

# Computer Vision Tasks

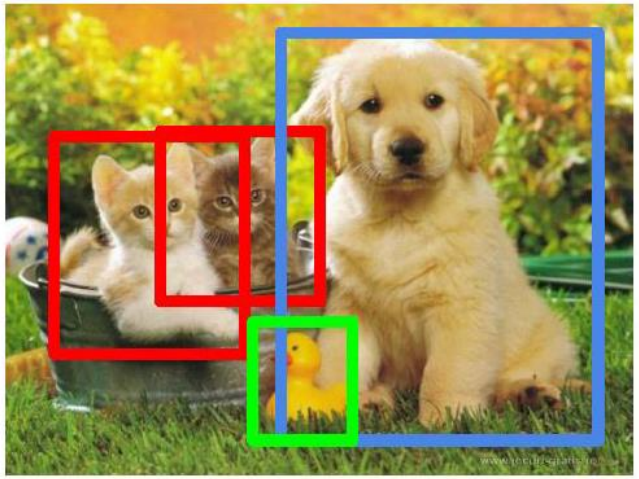
**Classification**

CAT

**Classification + Localization**

CAT

**Object Detection**



CAT + DOG + DUCK

**Instance Segmentation**

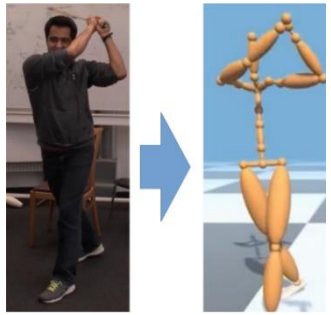
CAT + DOG + DUCK

Single Object

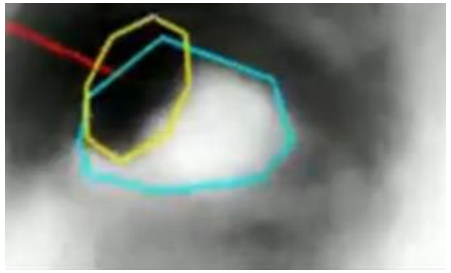
Multiple Objects

# Some Advanced Tasks

## Body Pose Estimation

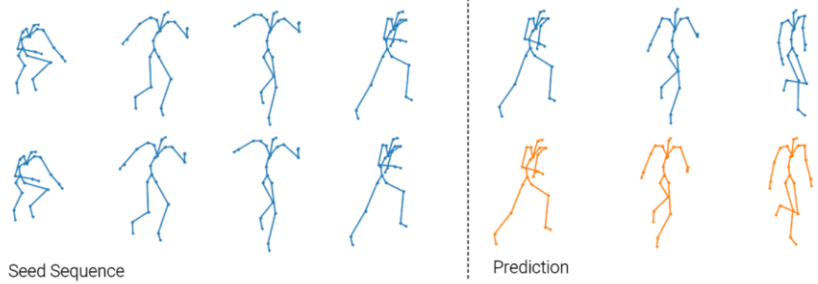


## Eye Gaze Estimation



## Dynamic Gesture Recognition

## Human Motion Prediction



# Feedforward object recognition (RSVP)

内容是一些图片的闪现

特异性

# Specificity versus Invariance

Goal:

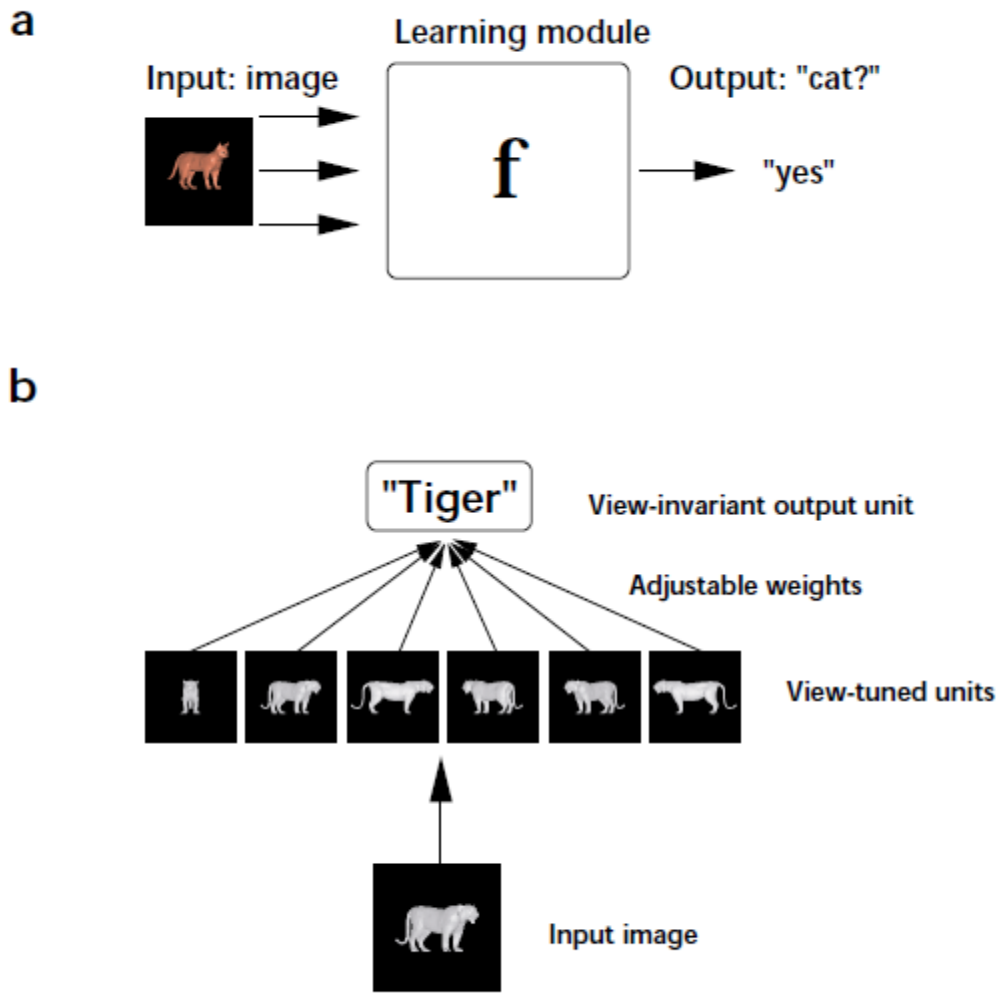
- Good Classification performance

Trade-off between:

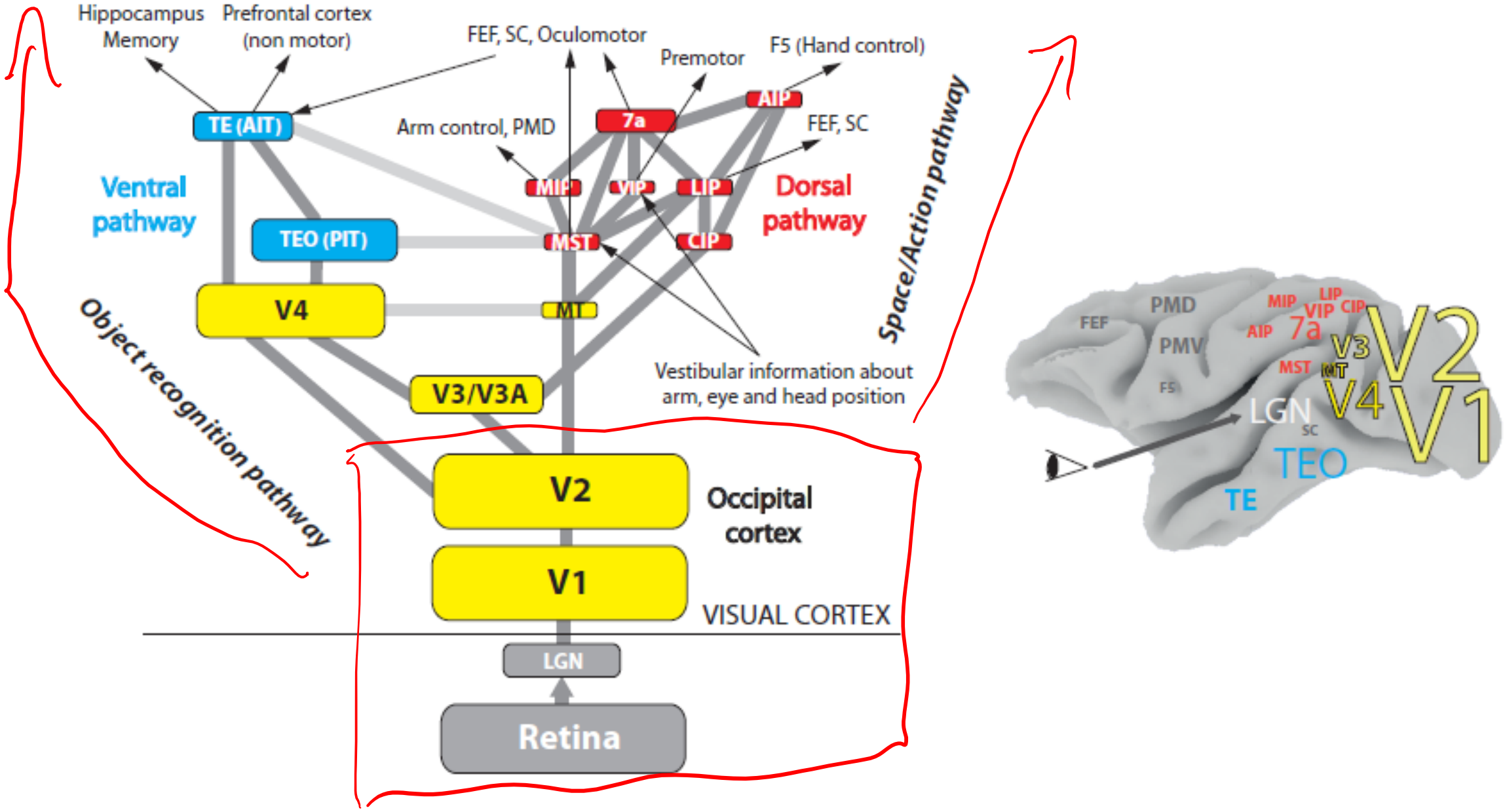
- Invariance
  - Affine transforms
  - Lighting
- Specificity

Impacts:

- Generalization ability




[Riesenhuber, Poggio. 2000. Models of Object Recognition. Nature]




[Kruger, Maersk, Jannssen et al. TPAMI 2012]

# Hubel & Wiesel – Visual Cortex & IT

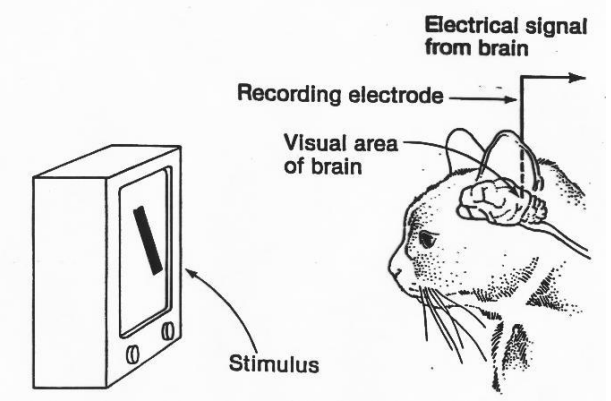
## 1959

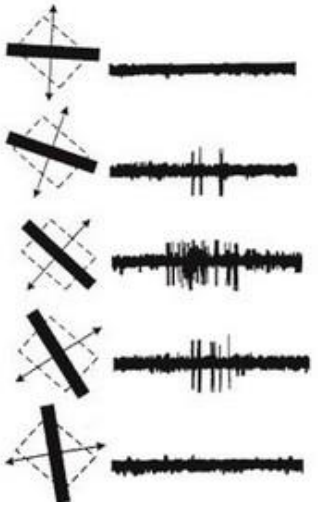
RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

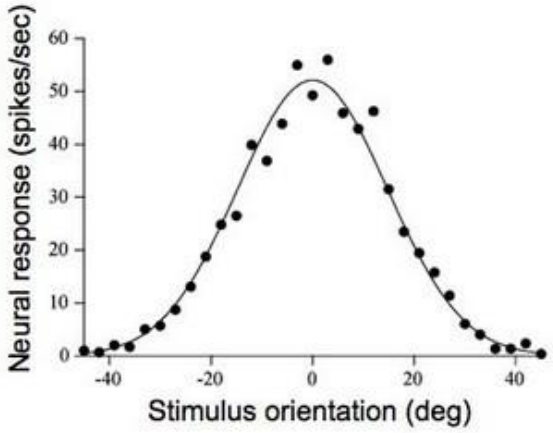
## 1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

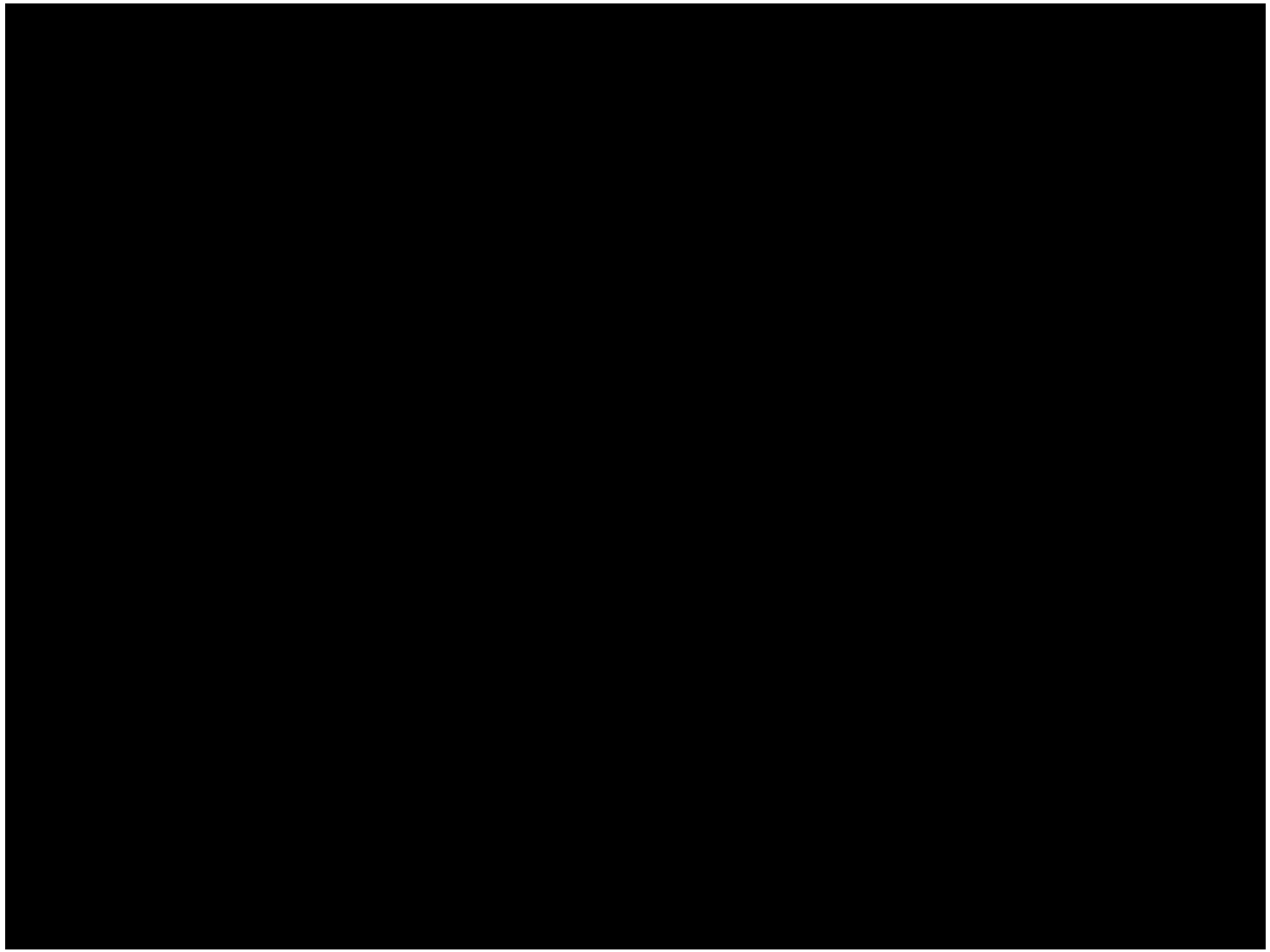
## 1968...





# Huber & Wiesel – Mapping Simple and Complex cells



[Hubel & Wiesel. 1962.
Receptive fields of single neurones in the cat's striate cortex.
J Physiol.1959;148:574–591.]

https://www.youtube.com/watch?v=8VdFf3egwfg

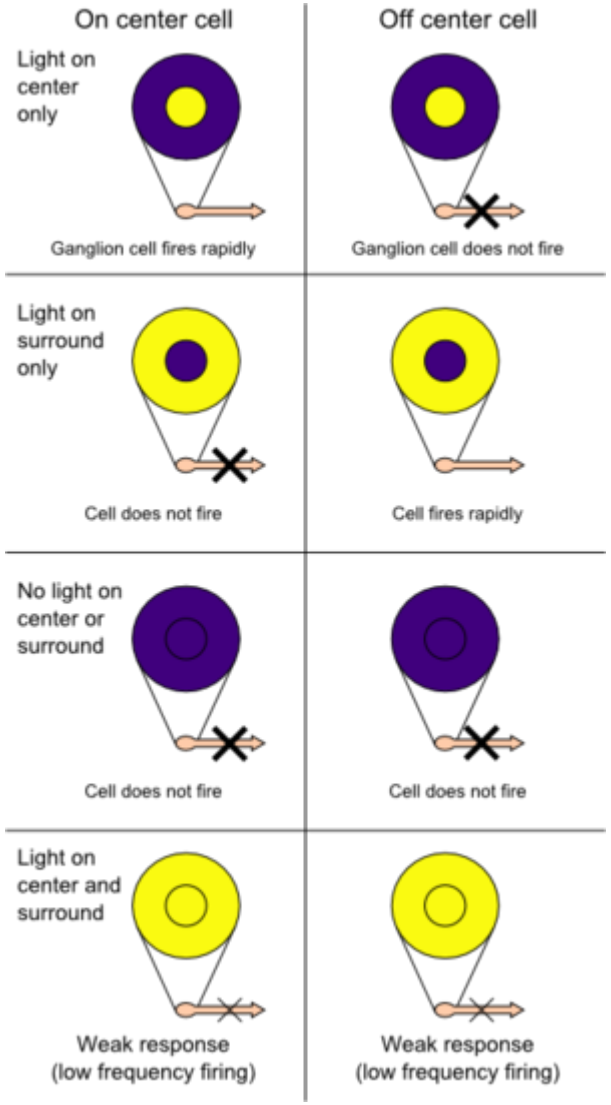
# Receptive fields

Particular part of the body surface in which a stimulus will trigger firing in sensory neurons

- Part of a hair in cochlea, piece of retina, tongue

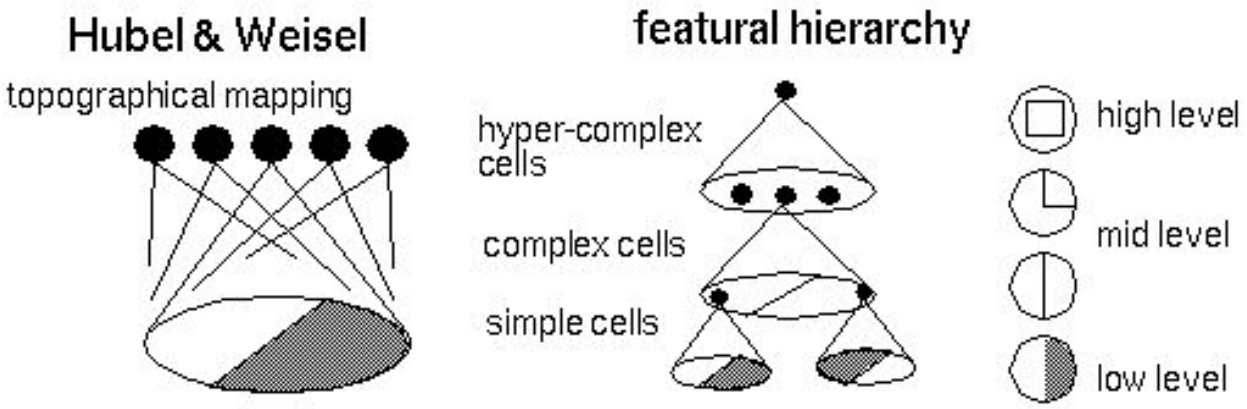
Hierarchies of neurons

- Many sensory receptors all synapsing with a single cell collectively form the receptive field of that cell
- E.g., a group of ganglion cells in retina form the receptive field for a cell in the brain



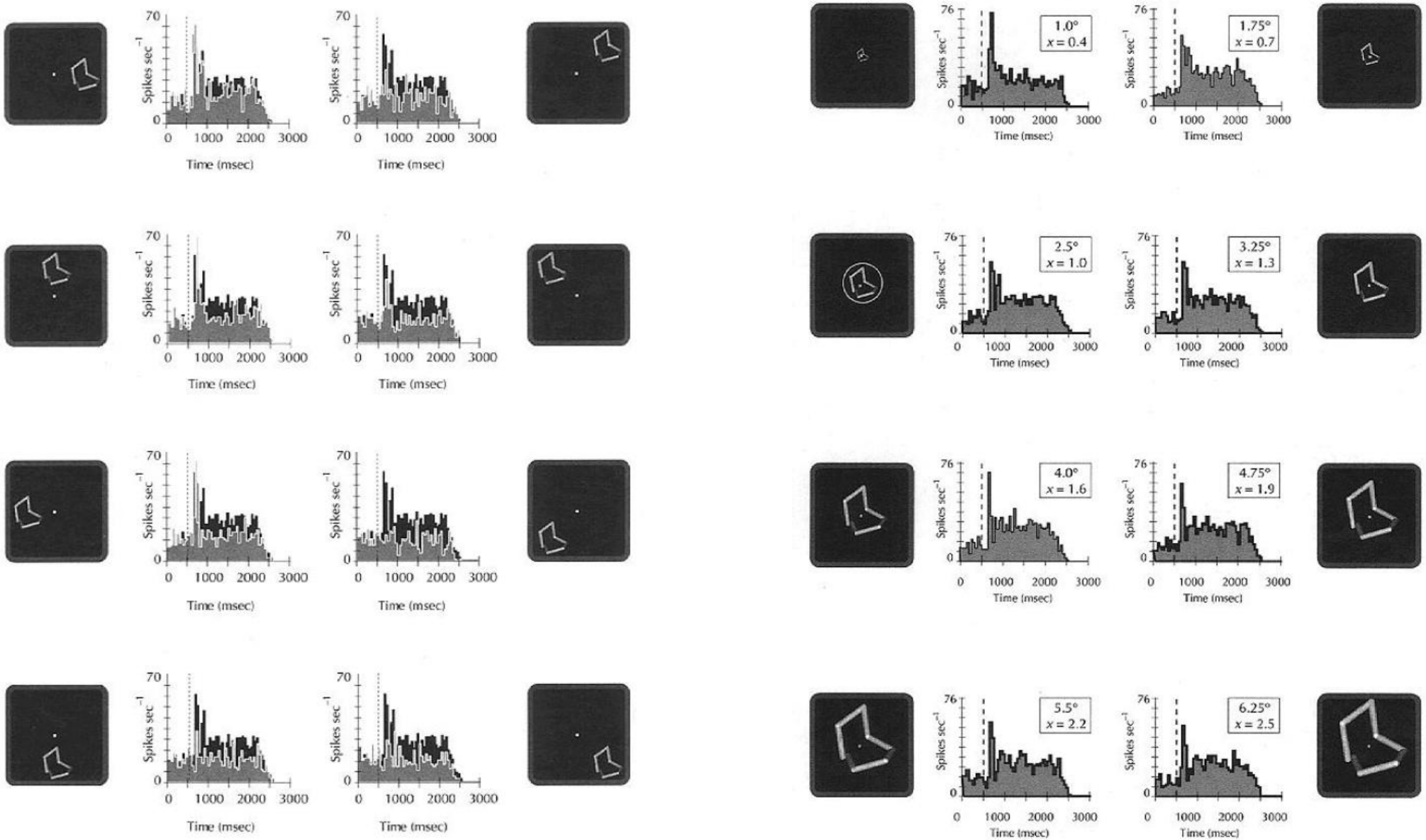

[http://en.wikipedia.org/wiki/Receptive_field]

# Receptive fields & feature hierarchies




[Hubel & Wiesel]

# Invariance to affine transforms




[Hubel & Wiesel]

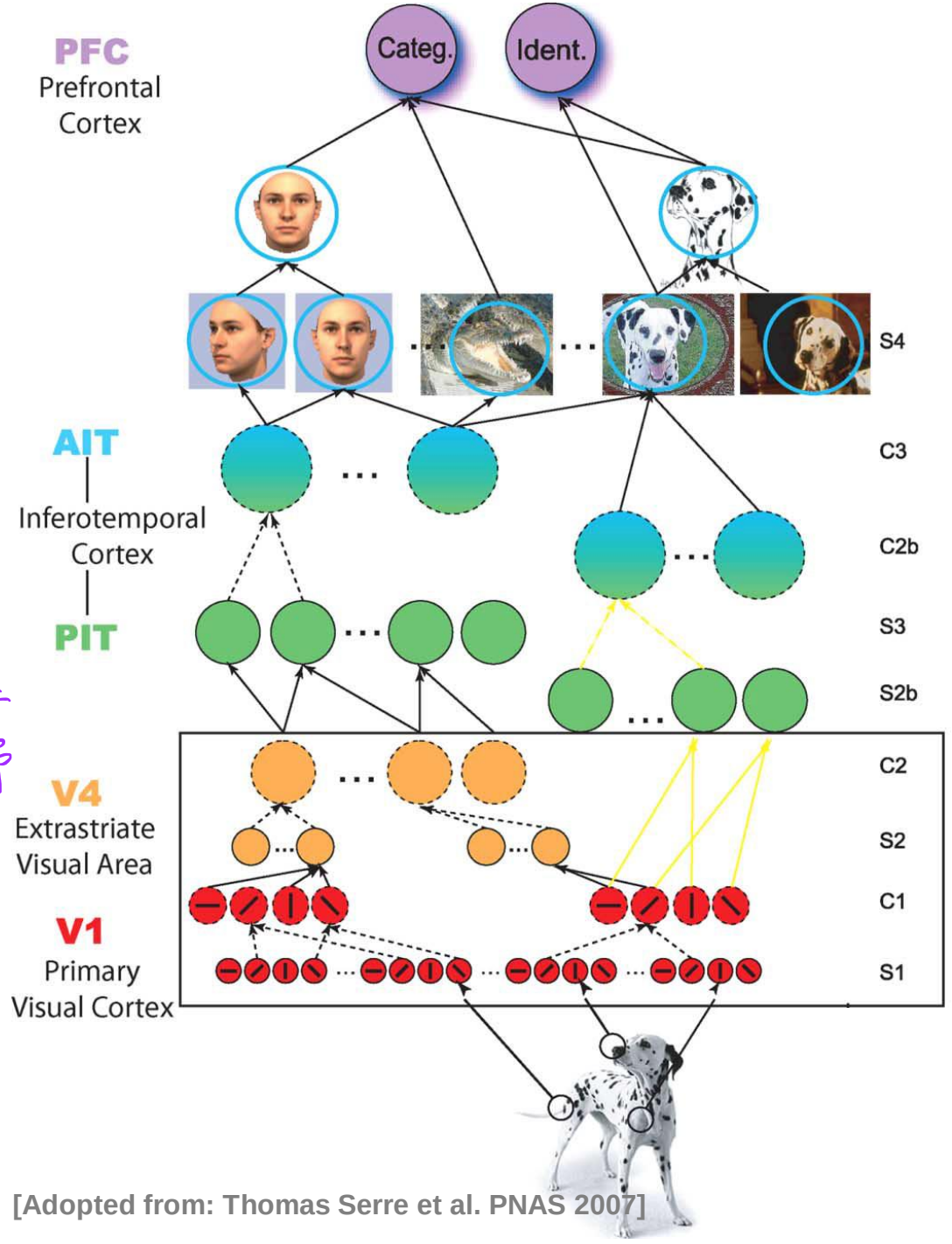
# HMAX Model

**S** (simple) cells are tuned to specific stimuli and have typically small receptive fields

**C** (complex) cells combine output from various S units to increase invariance and receptive field

增加不变性和感受视野

Many iterations of these operations allow for the construction of complex objects from low-level features




[Adopted from: Thomas Serre et al. PNAS 2007]

# HMAX Model

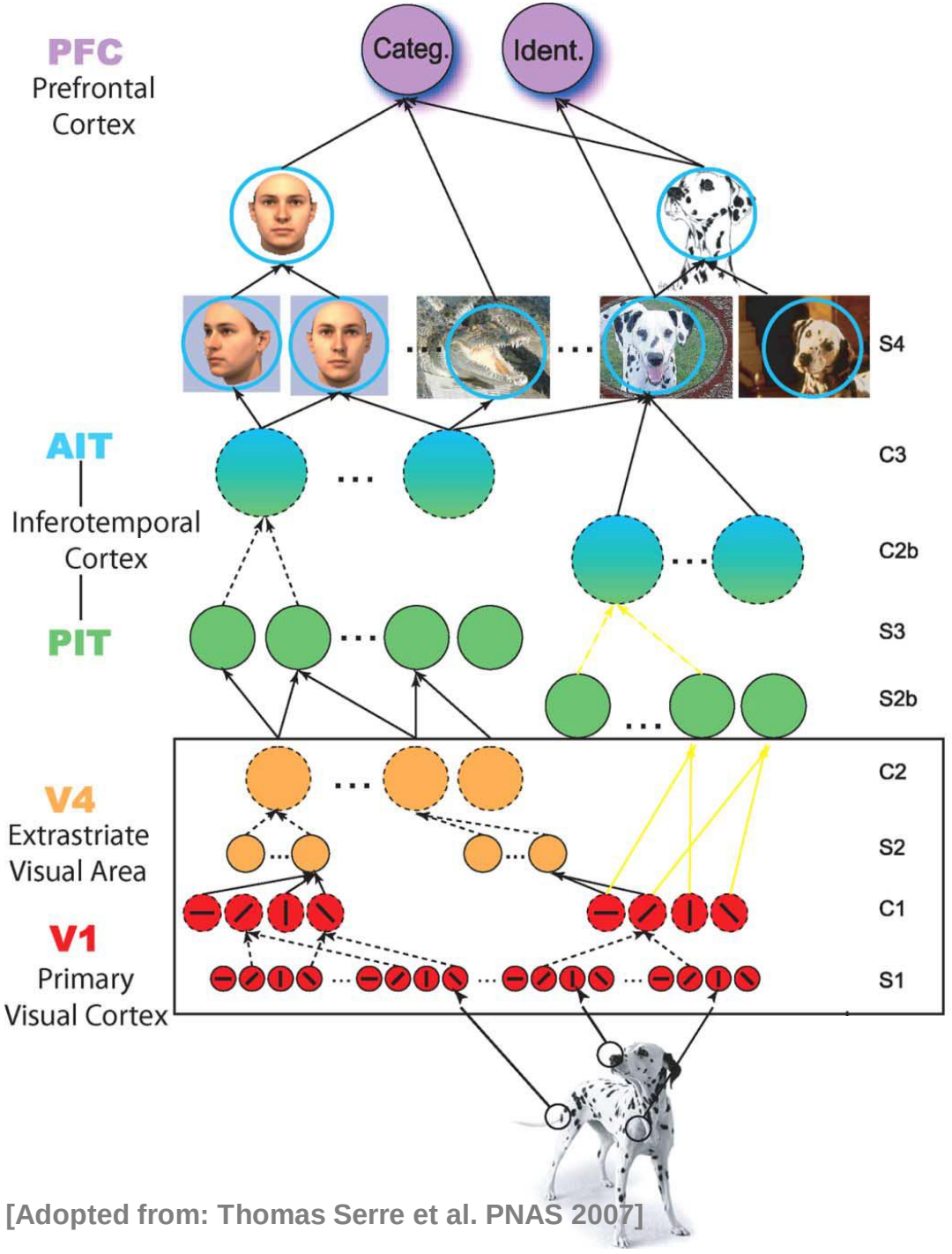
σ determine the sharpness of tuning of a particular cell.

**S**-cell response:

$$y = \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{n_{s_k}}\left(w_j - x_j\right)^2\right)$$

**C**-cell response:

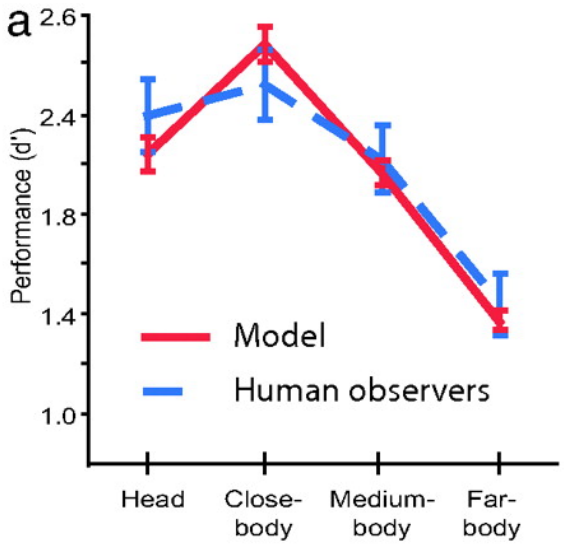
$$y = \max_{j=1\ldots n_{C_k}} x_j$$

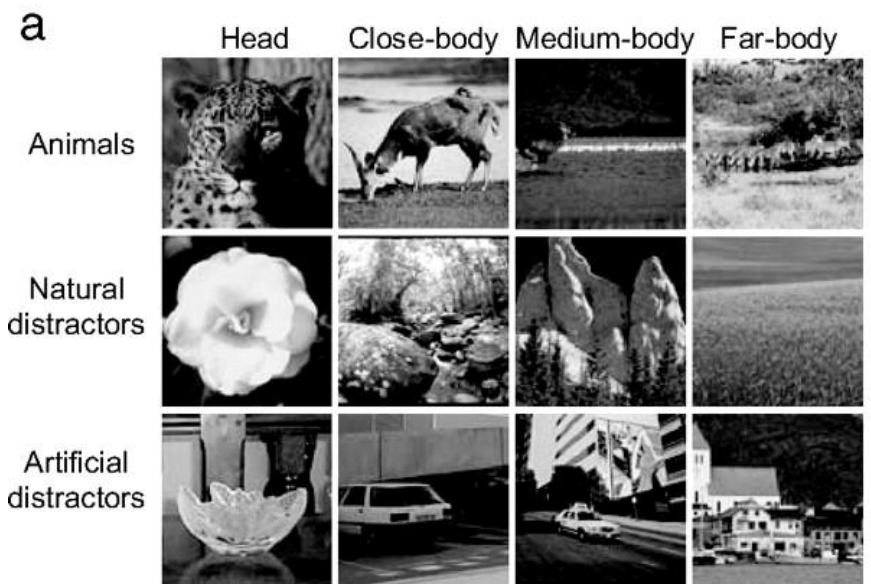


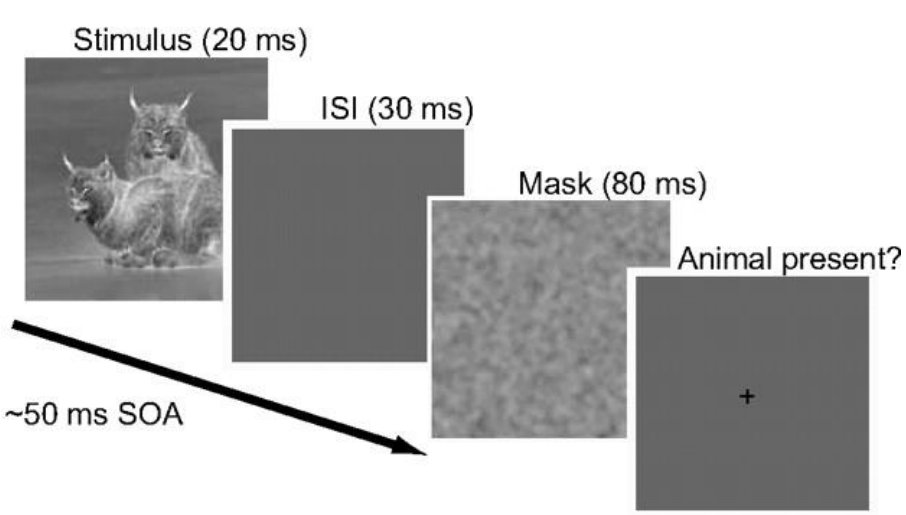

[Adopted from: Thomas Serre et al. PNAS 2007]

# Results

## ‘Solves’ the categorization problem - if mask forces feedforward processing



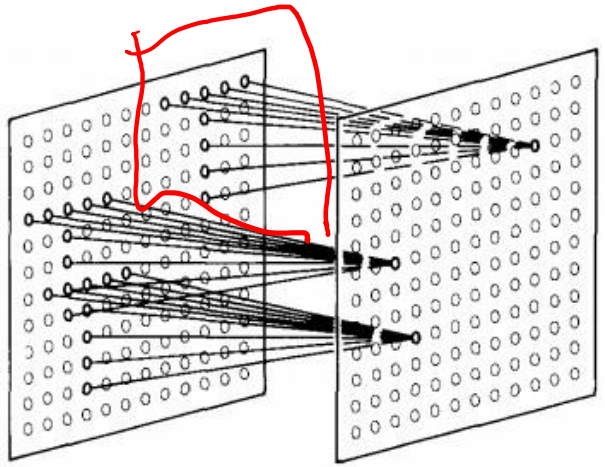



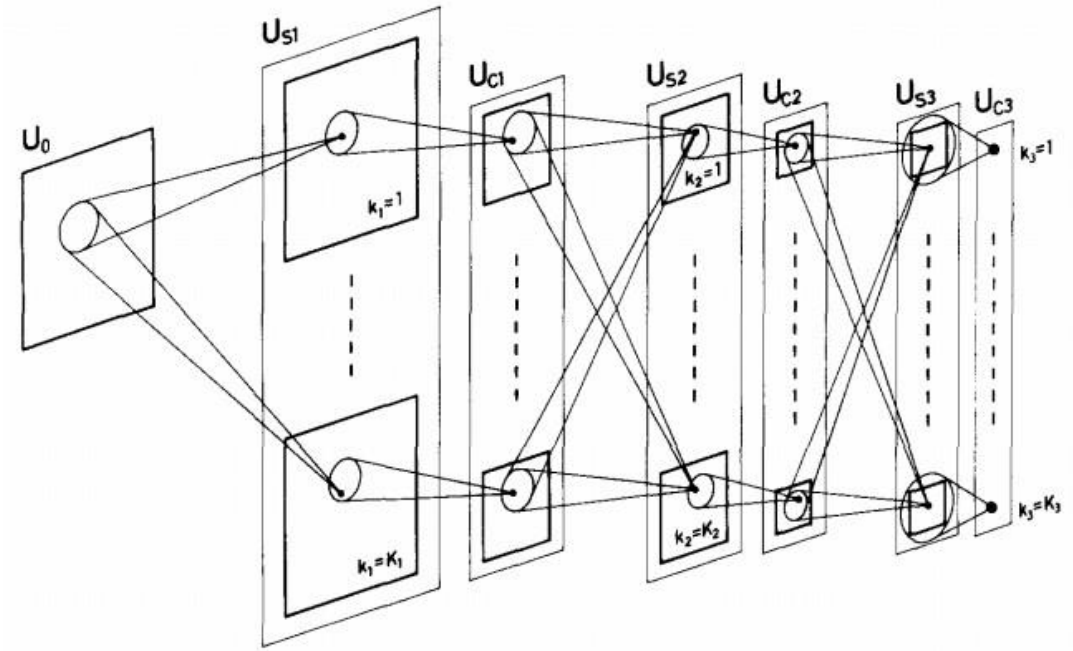



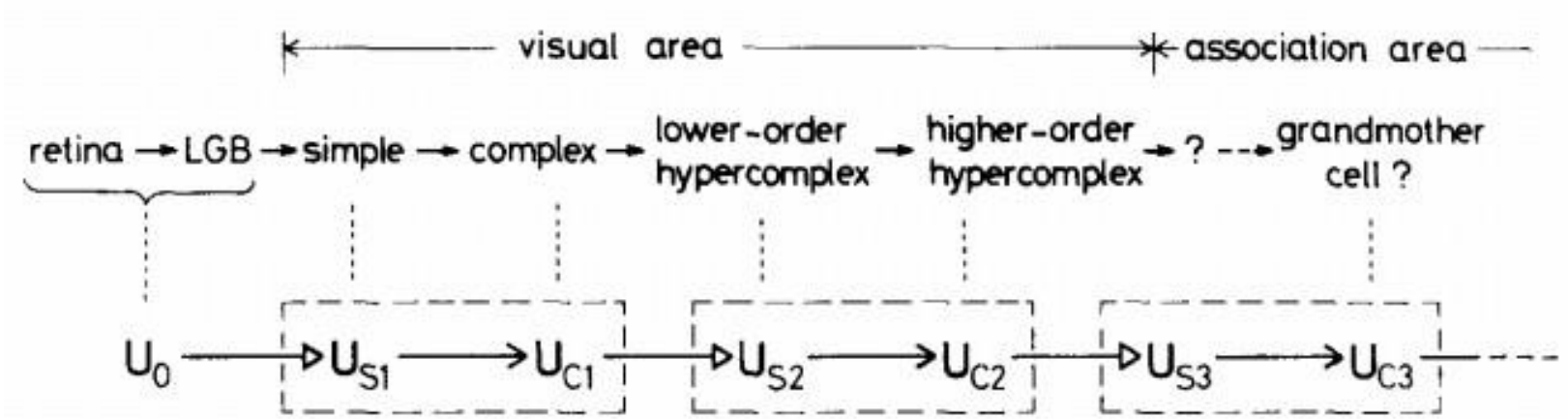

[Thomas Serre et al. PNAS 2007]

# Convolutional Neural Networks
# -
# A very brief history

# Neurocognitron

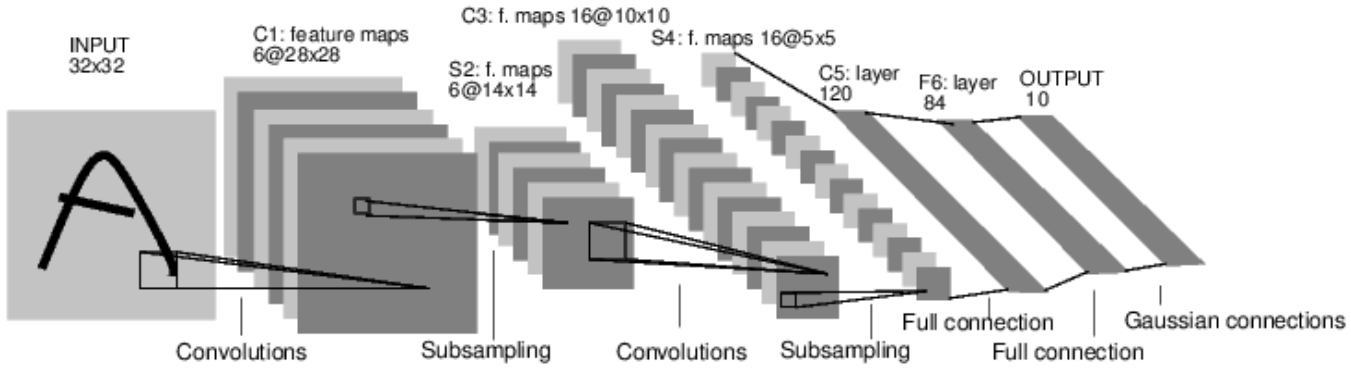




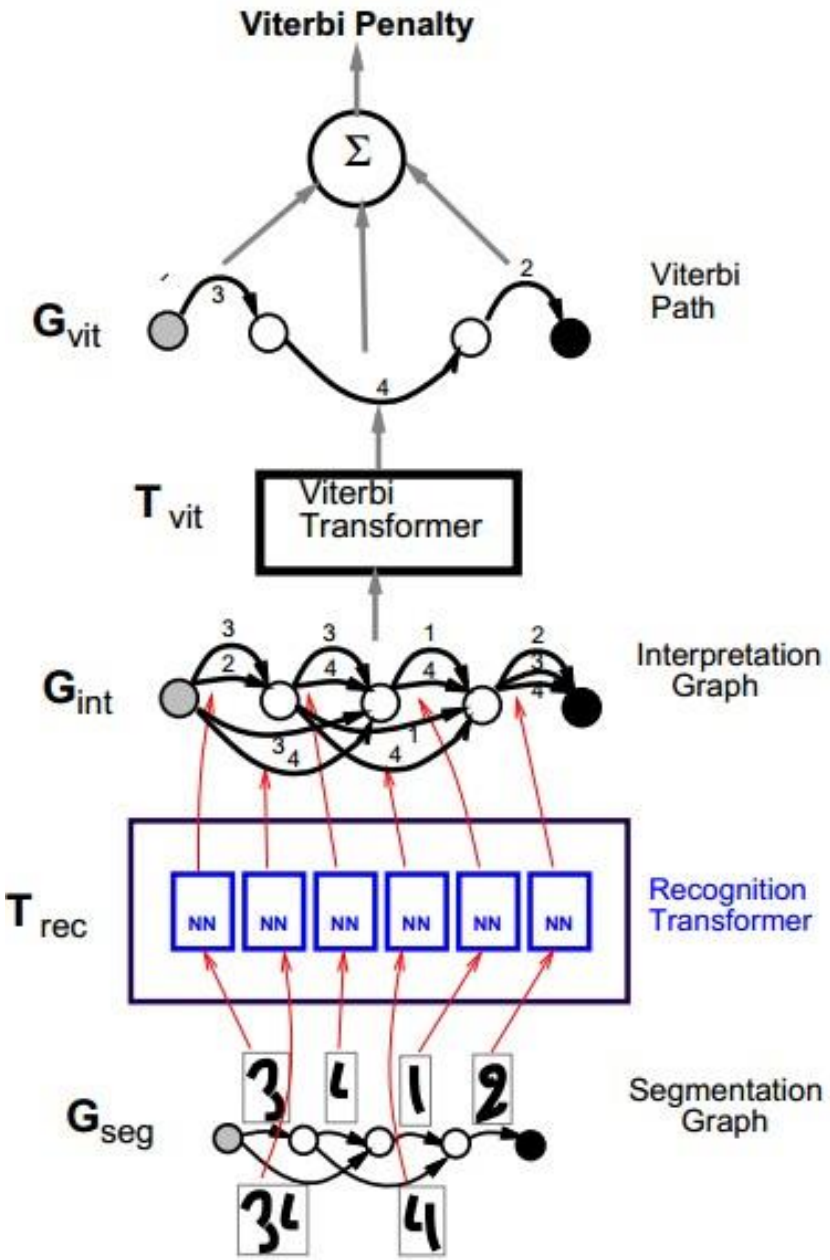




*[Fukushima. Neurocognitron. J. Biol. Cybernetics 1980.]*

# LeNet-5



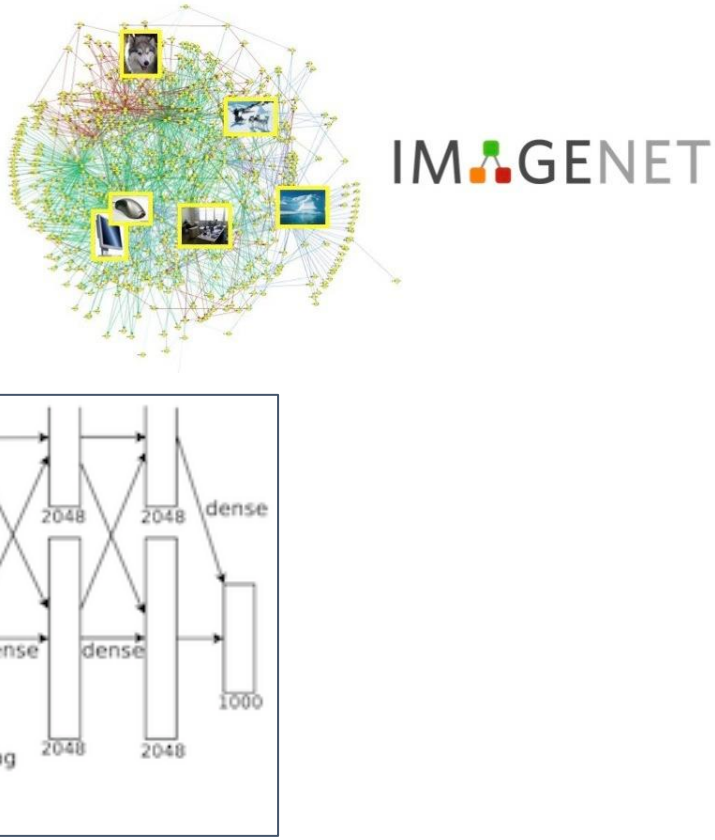

**Gradient-based learning applied to document recognition**
*[LeCun, Bottou, Bengio, Haffner 1998]*

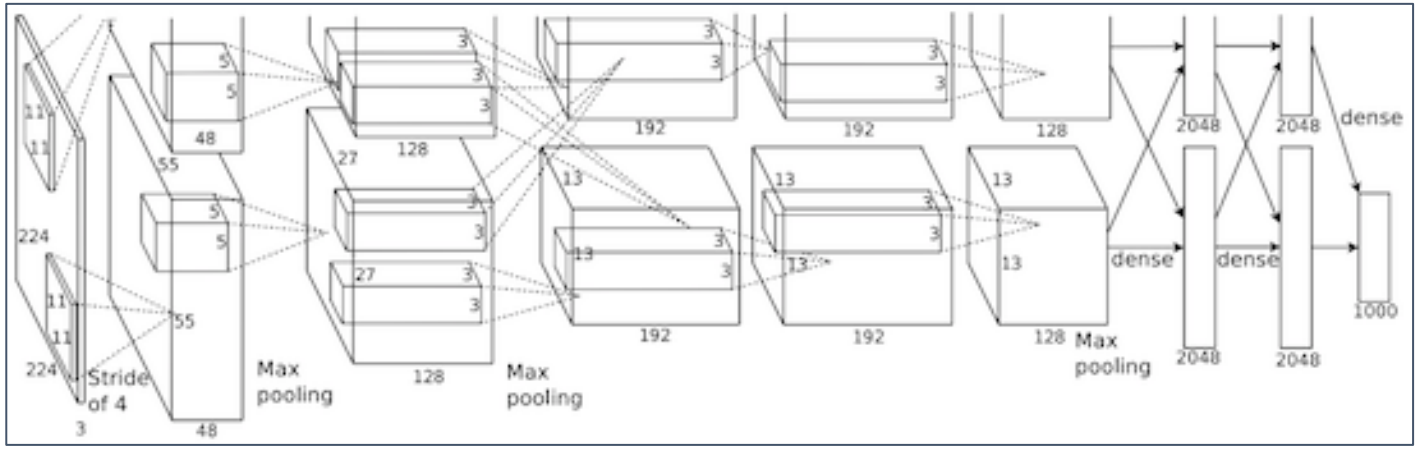


# ImageNet - 2012

**ImageNet Classification with**

**Deep Convolutional Neural Networks**

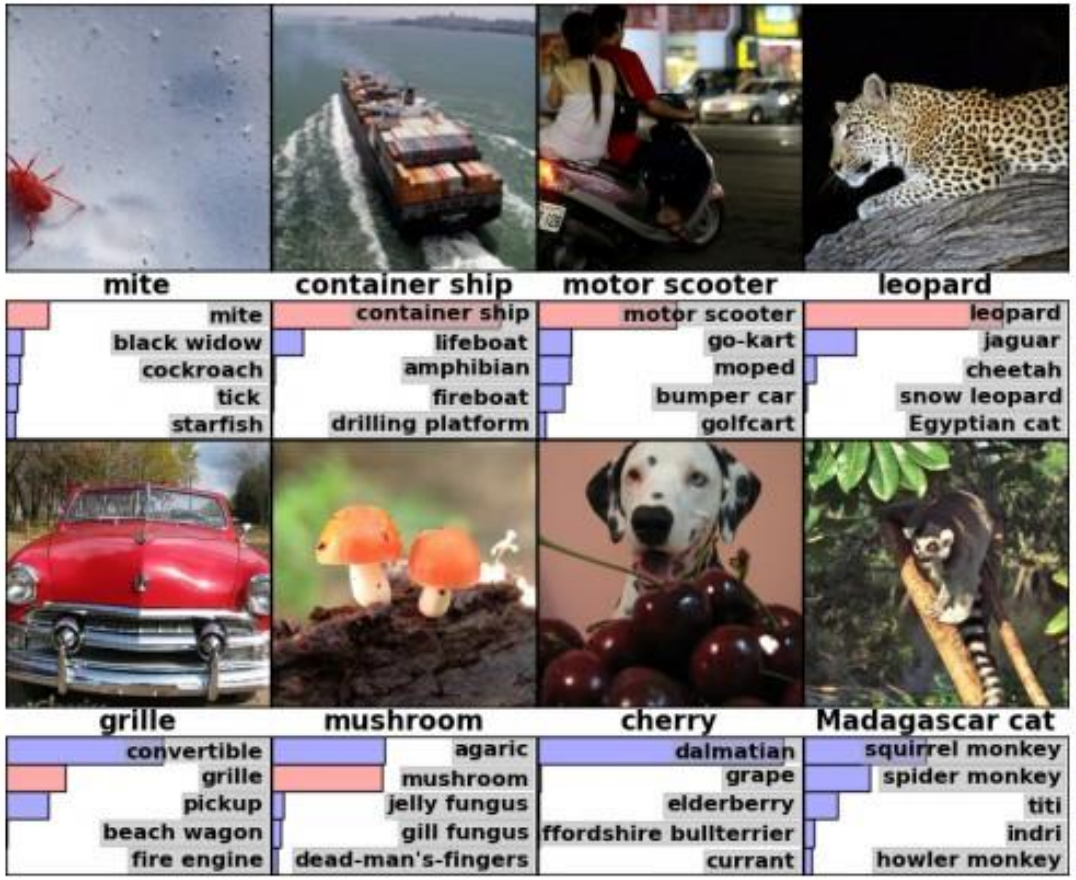



*[Krizhevsky, Sutskever, Hinton, 2012]*

# CNNs @ Large Scale

Classification

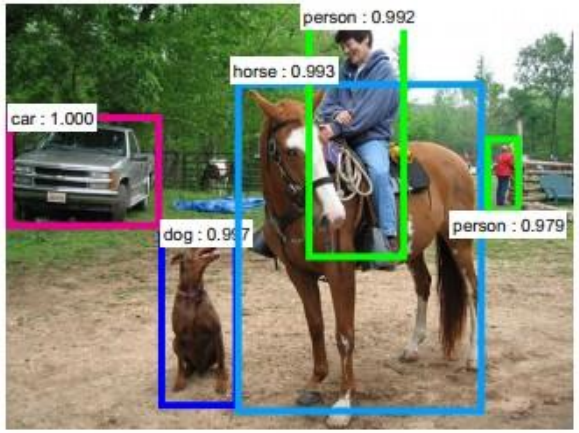


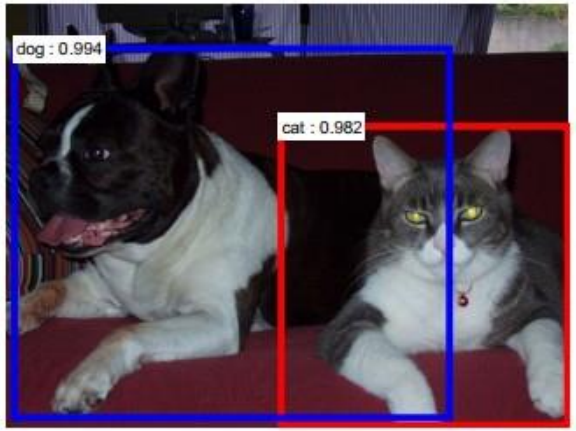

Retrieval

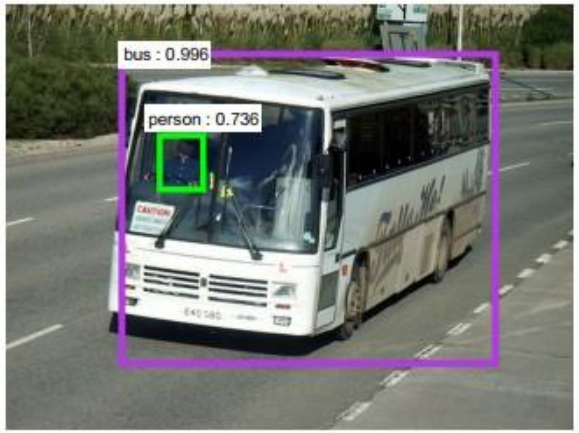
*[Krizhevsky 2012]*

# Fine-Grained Prediction

Detection



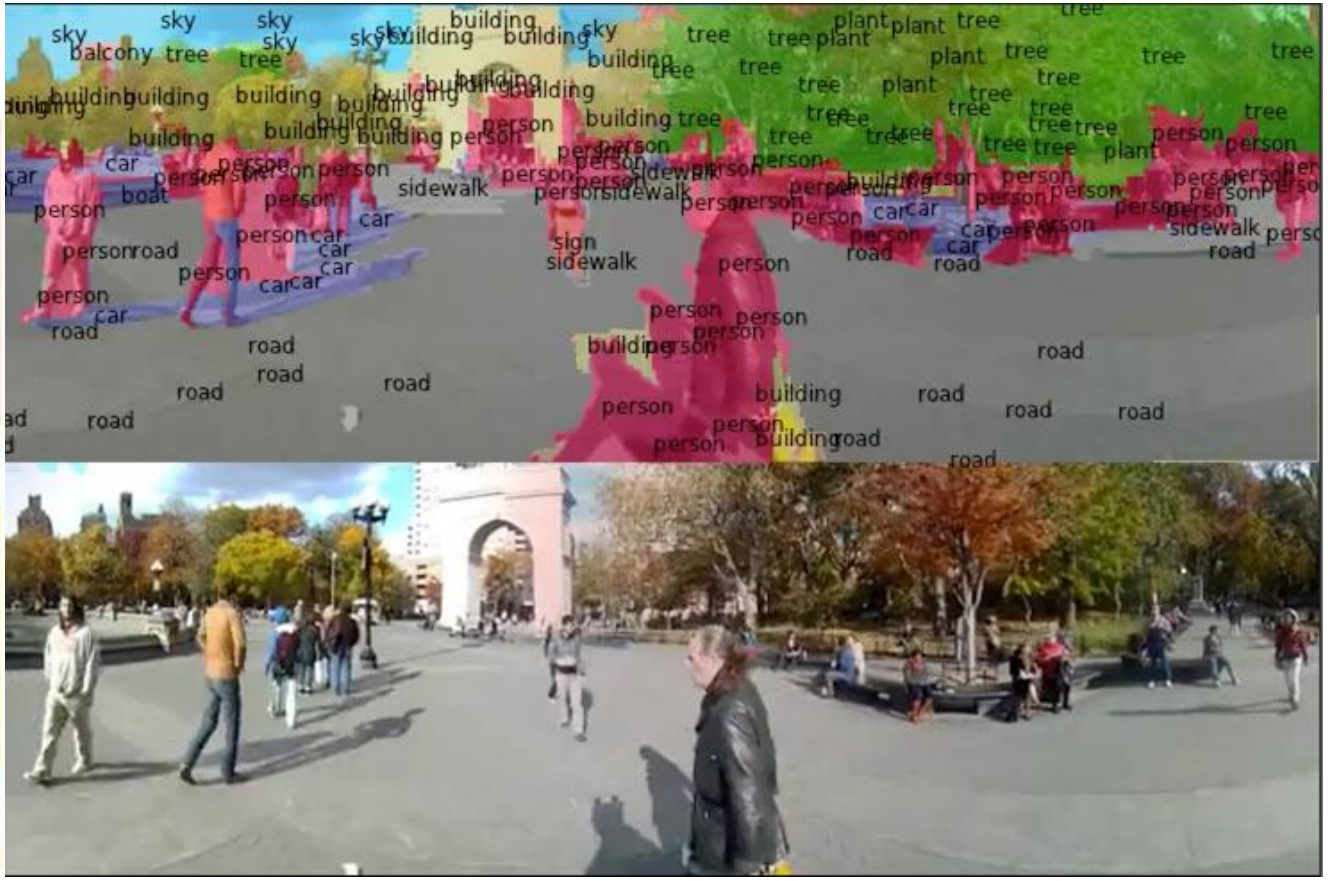





*[Faster R-CNN: Ren, He, Girshick, Sun 2015]*

Segmentation

Semantic : each object get labled.

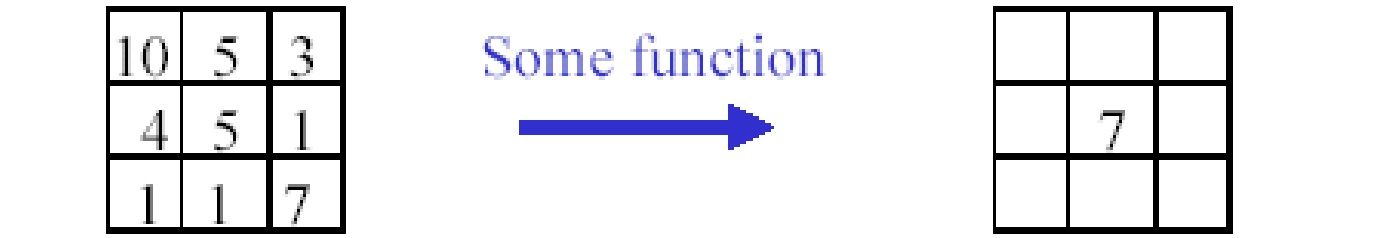



*[Farabet et al., 2012]*

# Convolutions

# What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.




Local image data

Modified image data

# Linear Transforms

Given a transform $T$, a function $f$, defining a translation or shift operation and input vectors $\boldsymbol{u}, \boldsymbol{v}$ scalars $\alpha, \beta$

Definition: Transform $T$ is *linear* if,

$$T(\alpha \boldsymbol{u} + \beta \boldsymbol{v}) = \alpha T(\boldsymbol{u}) + \beta T(\boldsymbol{v})$$

Definition: A transform $T$ is *invariant* to $f$ if,

for classification, we do want invariant.

$$T\big(f(\boldsymbol{u})\big) = T(\boldsymbol{u})$$

Definition: A transform $T$ is *equivariant* to $f$ if,

$$T\big(f(\boldsymbol{u})\big) = f\big(T(\boldsymbol{u})\big)$$

Any linear, shift-equivariant transform $T$ can be written as a convolution

# Linear Filtering

Operations are "shift-invariant" if *K* does NOT depend on (*i*,*j*): using same weights everywhere!

Linear operations can be written:

$$I'(i,j) = \sum_{(m,n)\in N(i,j)} K(\cancel{i,j},m,n)I(i+m,j+n)$$

kernel window patch

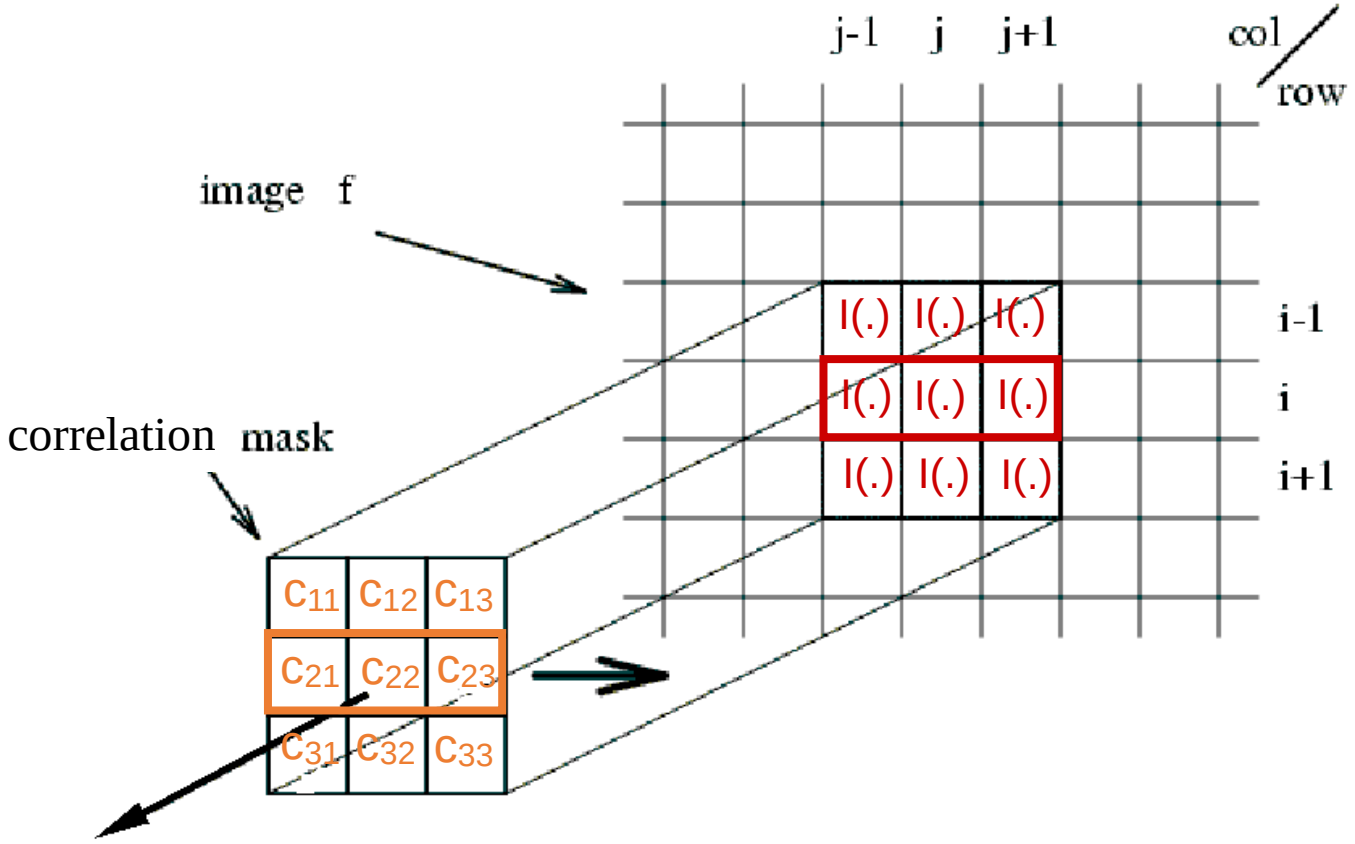
*I* is the input image; *I'* is the output of the operation.

*k* is the *kernel* of the operation. *N*(*m*,*n*) is a neighbourhood of (*m*,*n*).

# Correlation

correlation is the techneque for template matching.

(e.g. template matching)




Using inner product to compute similarities between two patches.
Same thing hold true for gray scale images.

If the mask's pixel is the same as another patch, the value of product will be large and that is a high correlation

$$o(i,j) = c_{11}\, I(i-1,j-1) + c_{12}\, I(i-1,j) + c_{13}\, I(i-1,j+1) + c_{21}\, I(i,j-1) + c_{22}\, I(i,j) + c_{23}\, I(i,j+1) + c_{31}\, I(i+1,j-1) + c_{32}\, I(i+1,j) + c_{33}\, I(i+1,j+1)$$

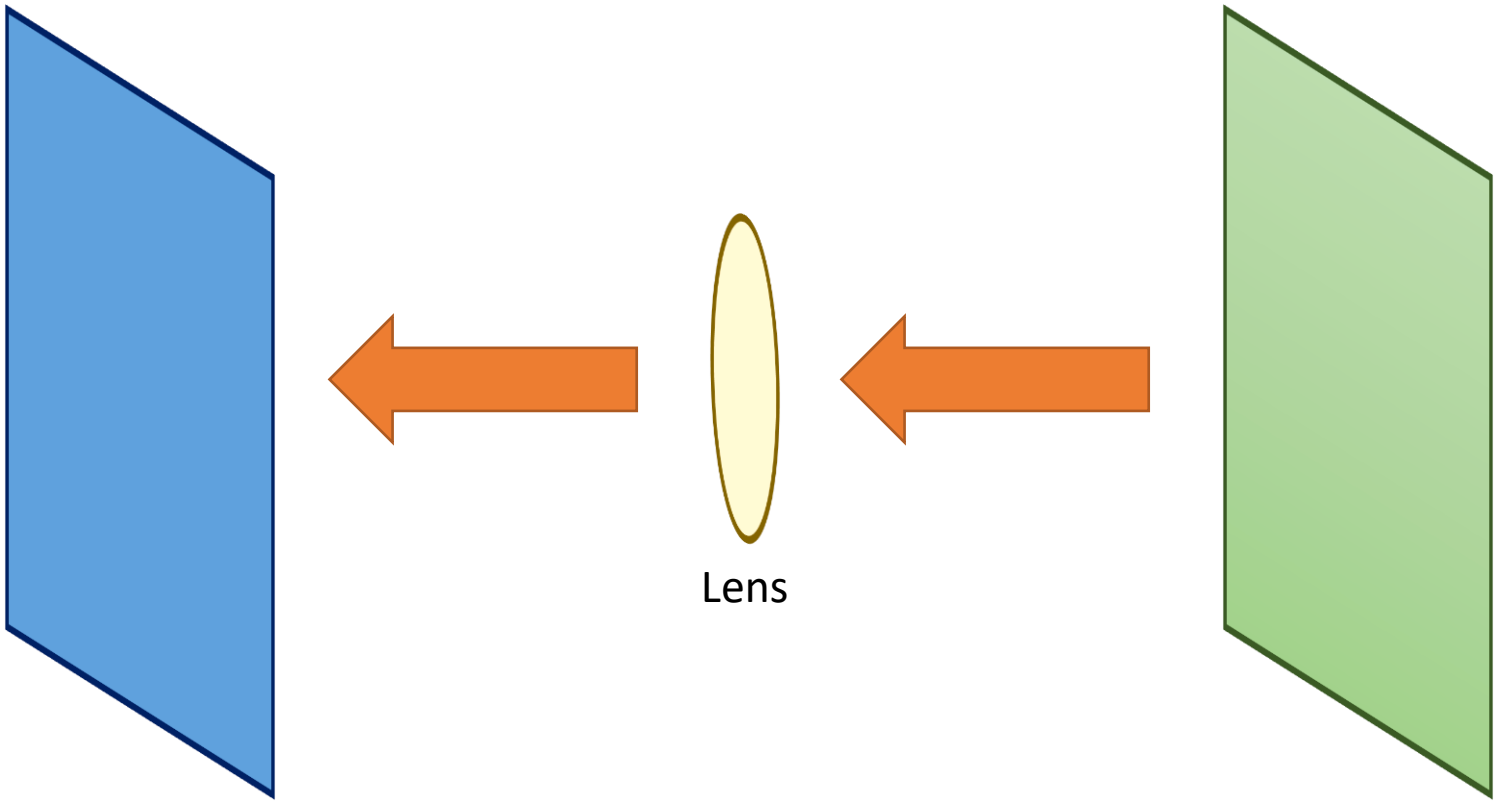
# Correlation

$$I'(i, j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) I(i+m, j+n)$$

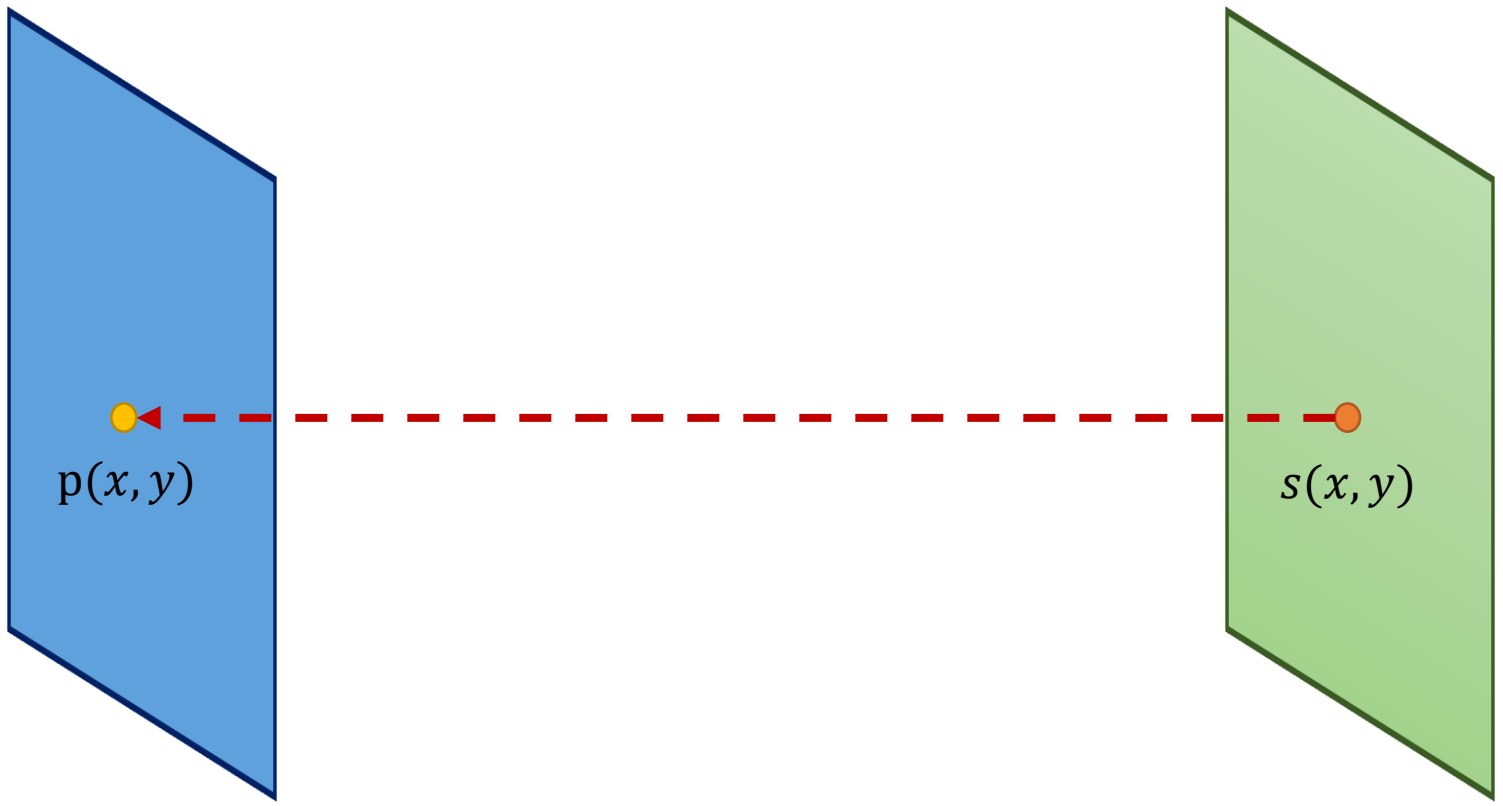
# Image Convolution - Example

Convolution is the same as correlation, just with flip kernel.

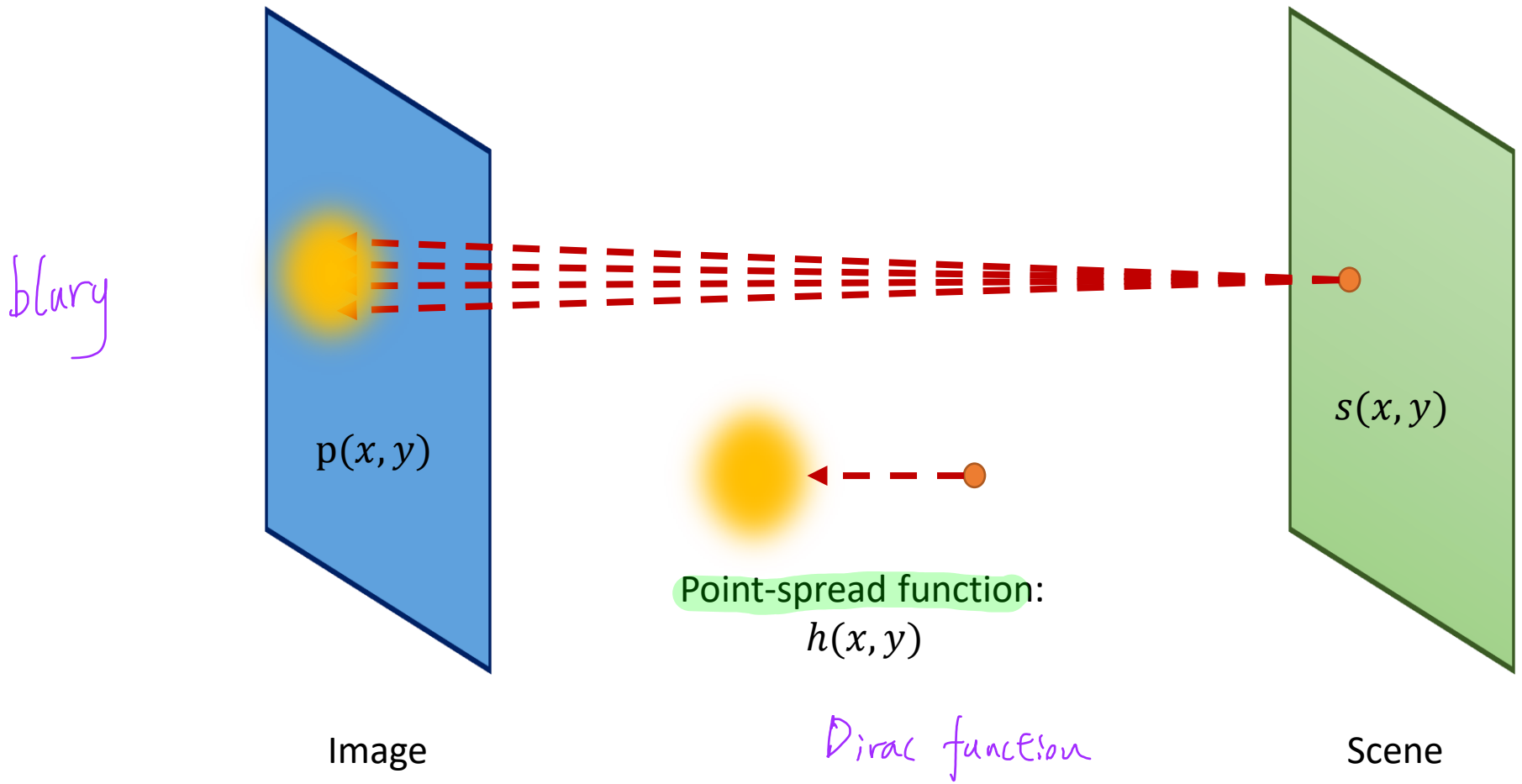
Convolution : additive function



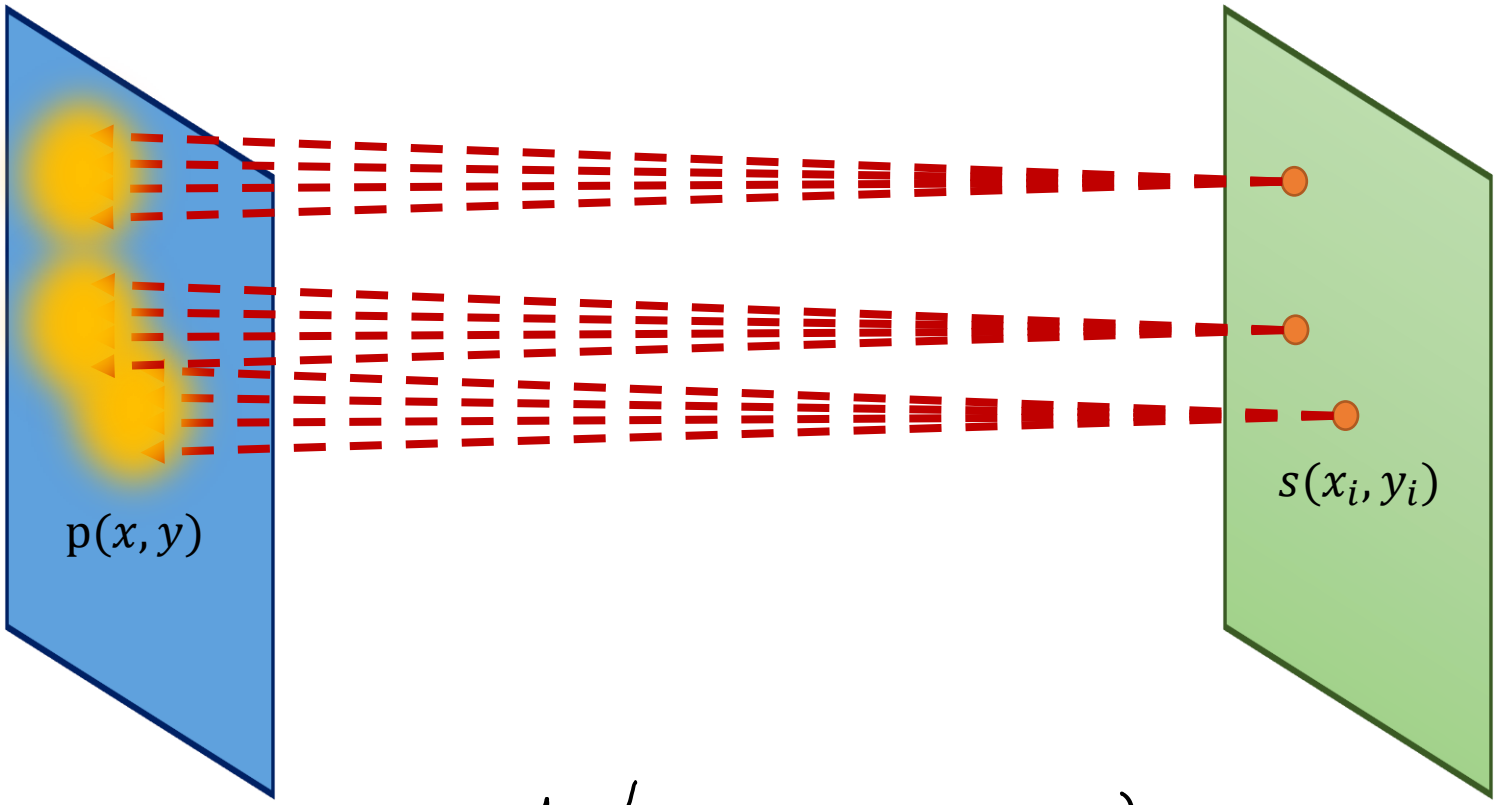
# Image Convolution - Example



# Image Convolution - Example



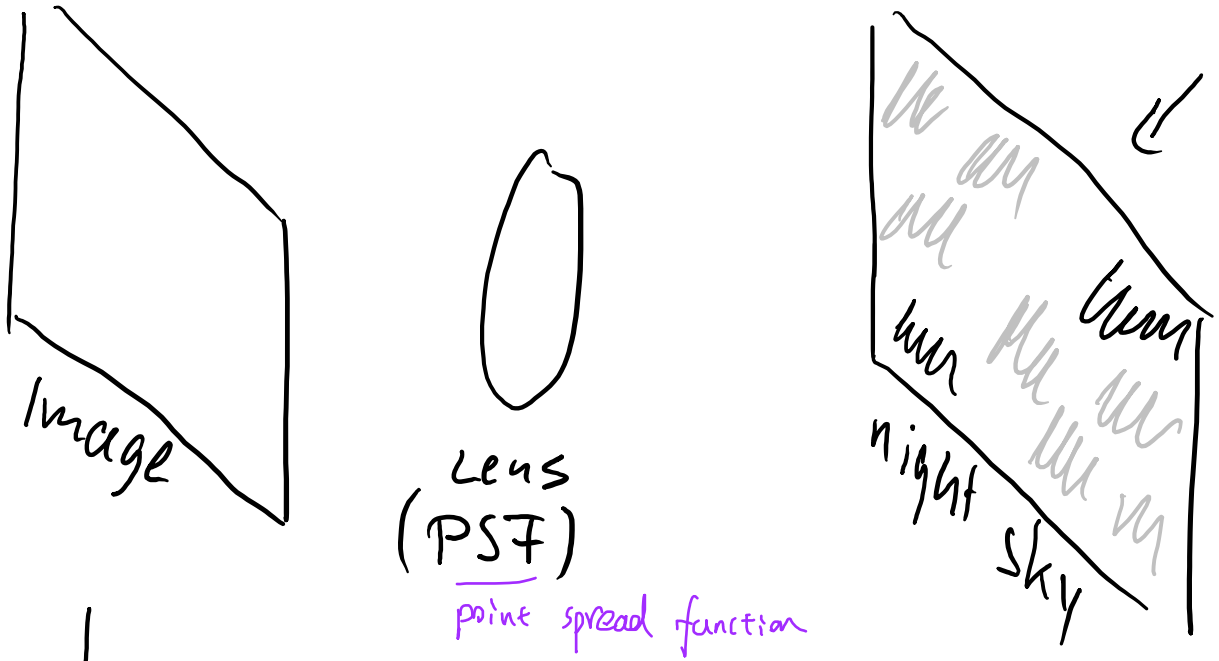
# Image Convolution - Example




$$\sum_i s_i \, h(x - x_i, y - y_i)$$

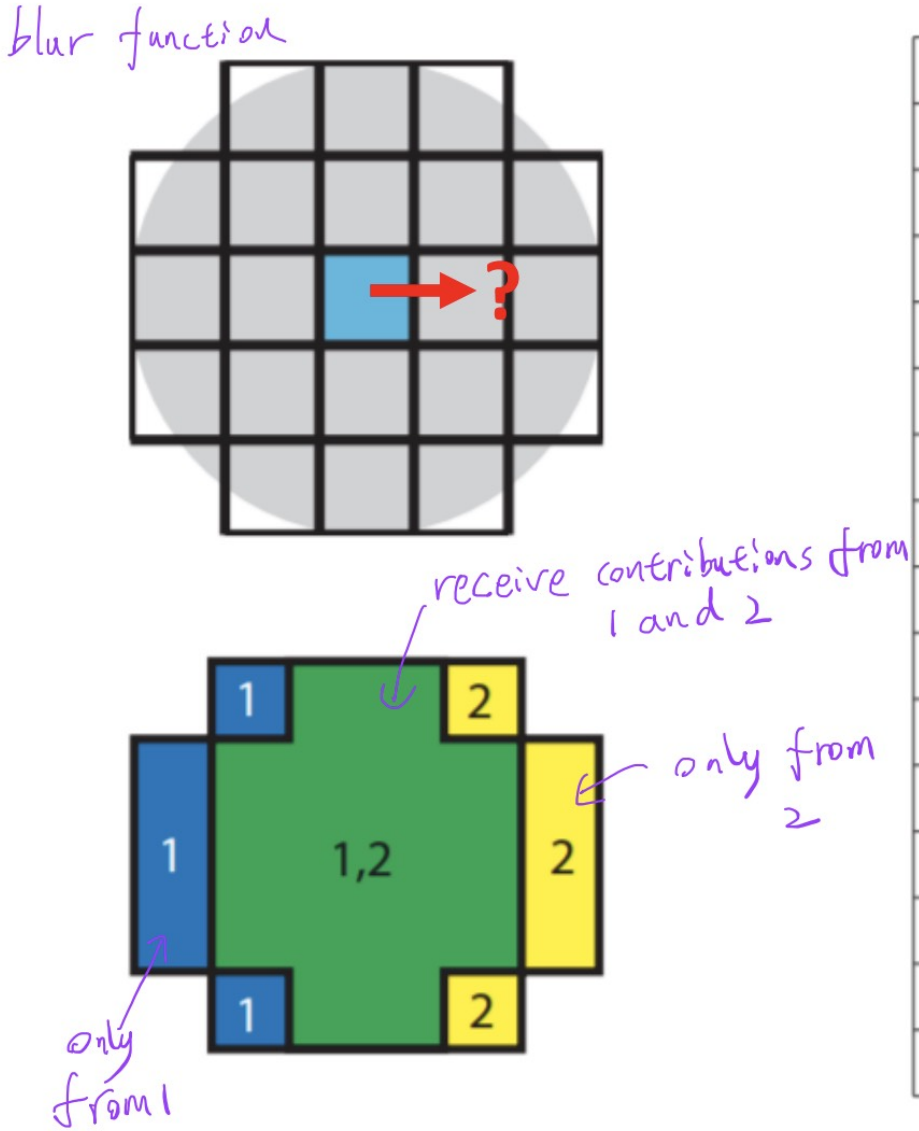
Sampling

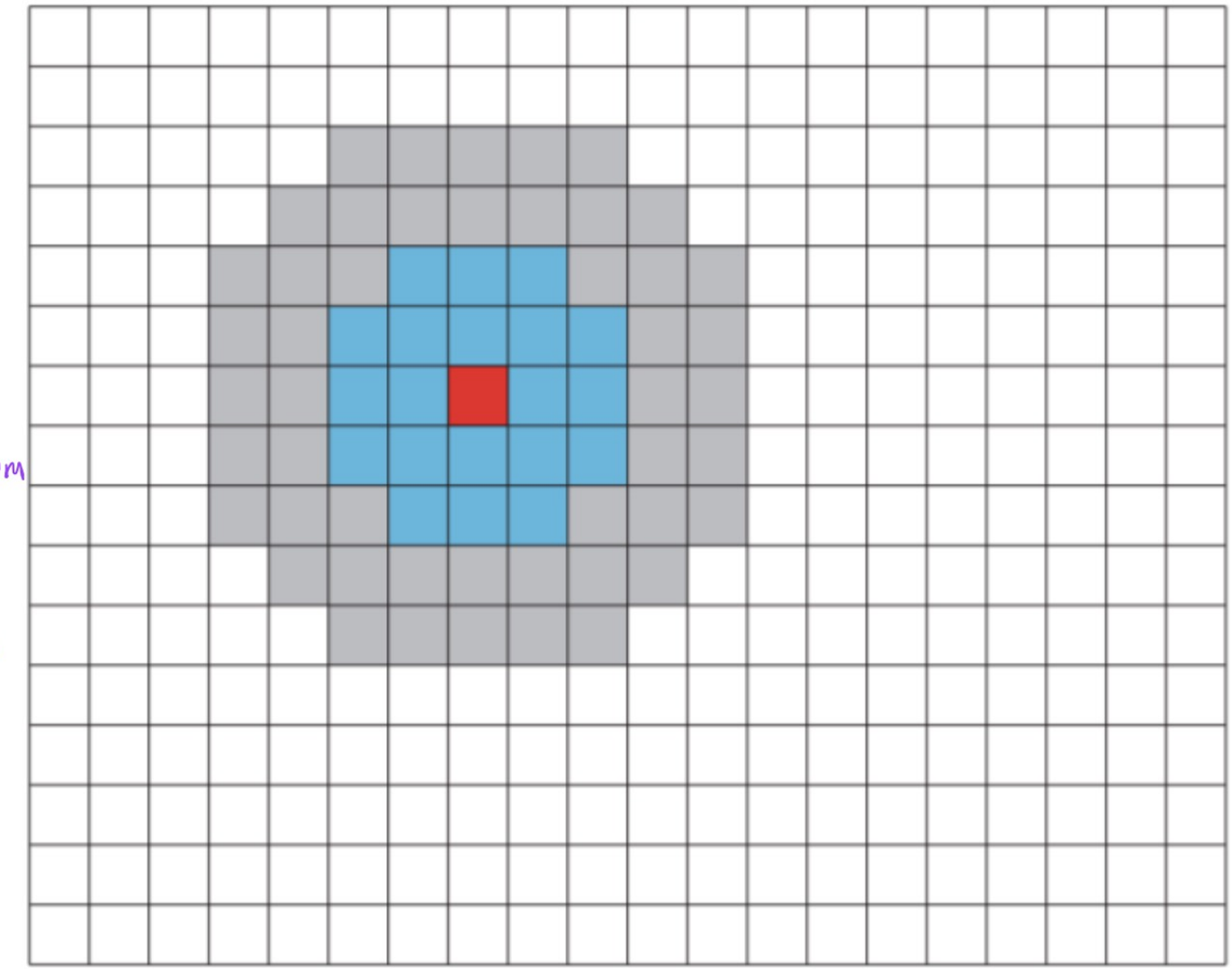
# Convolution - continous case:




$$\int \underbrace{m(u,v)}_{\text{brightness at } u,v}\, h(x-u, y-v)\, du\, dv$$

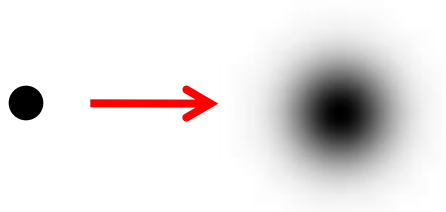
# Image Convolutions as Image Filter



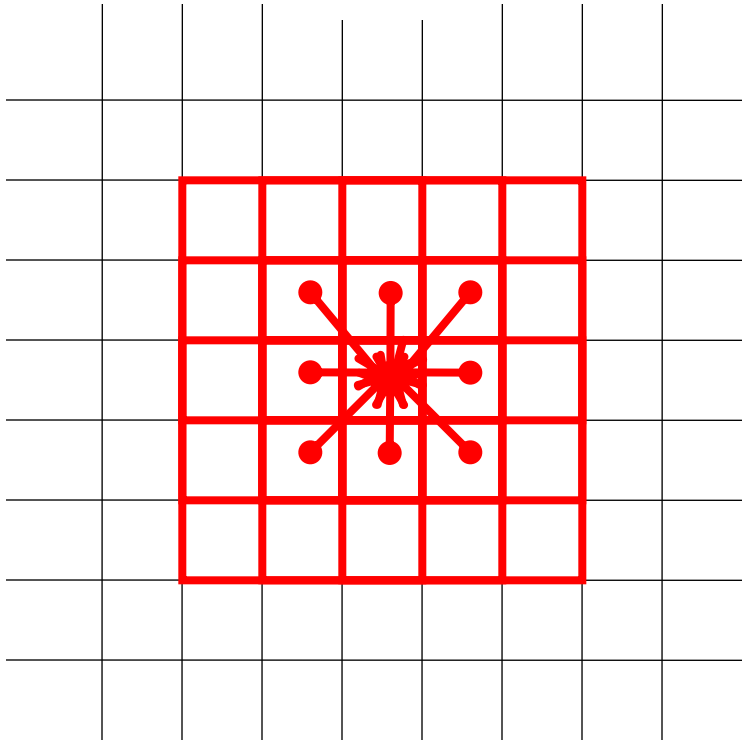
[image source: Carlo Tomassi]

# Convolution
(e.g. point spread function)



Kernel

| | | |
|---|---|---|
| K(-1,-1) | K(0,-1) | K(1,-1) |
| K(-1,0) | K(0,0) | K(1,0) |
| K(-1,1) | K(0,1) | K(1,1) |



k is flipped

$$I'(x,y)=K(1,1)I(x-1,y-1) + K(0,1)I(x,y-1) + K(-1,1)I(x+1,y-1)$$
$$+ K(1,0)I(x-1,y) + K(0,0)I(x,y) + K(-1,0)I(x+1,y)$$
$$+ K(1,-1)I(x-1,y+1) + K(0,-1)I(x-1,y) + K(-1,-1)I(x+1,y+1)$$

# Correlation

$$I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) I(i+m, j+n)$$

# Convolution

$$I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) I(i-m, j-n)$$

$$= \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(-m,-n) I(i+m, j+n)$$

convolution, these kind of linear filtering operations are basically gonna become our weight matrix in CNN

In CNN, we basically try to learn the weights of the kernel. The weights (patch) we slide over the image are learn from data.

So if *K(i,j) = K(-i, -j)*, then Correlation == Convolution

=> kernel is symmetric

# Convolution as Matrix multiplication

Convolution operation on finite signals and kernels (i.e., images) can be implemented via matrix-vector multiplication:

$$I(u) = 0 \text{ for } u \notin [1:n], \qquad K(u) = 0 \text{ for } u \notin [1:m], m \leq n$$

$$(I * K) = \begin{pmatrix} k_1 & 0 & \cdots & 0 \\ k_2 & k_1 & \vdots & \vdots \\ k_3 & k_2 & k_1 & \\ \vdots & k_3 & \vdots & \\ 0 & \vdots & 0 & k_m \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ \vdots \\ I_n \end{pmatrix}$$

n

n+m-1

n+m-1 x n

n x 1

# Differentiation and convolution

For discrete data, we can approximate using finite differences:

$$\frac{\partial I(x,y)}{\partial x} \approx \frac{I(x+1,y)-I(x,y)}{1}$$

Recall, for 2D function, f(x,y):

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right)$$

This is linear and shift invariant, so must be the result of a convolution.

We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1},y)-f(x_n,y)}{\Delta x}$$

| -1 | 1 |
|---|---|

(which is obviously a convolution)



[Slide: D.A. Forsyth]

# Partial derivatives of an image

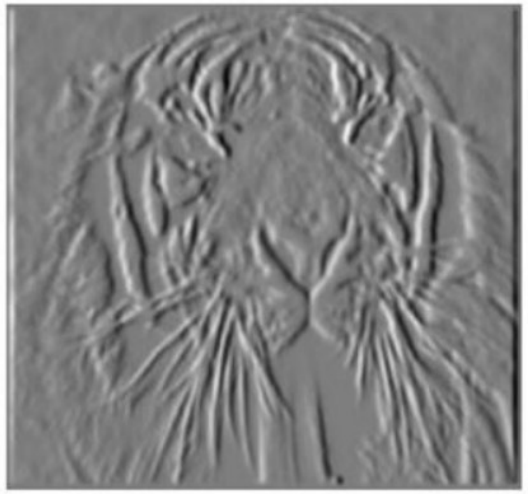
x方向上：detect vertical edges
find horizontal diff.

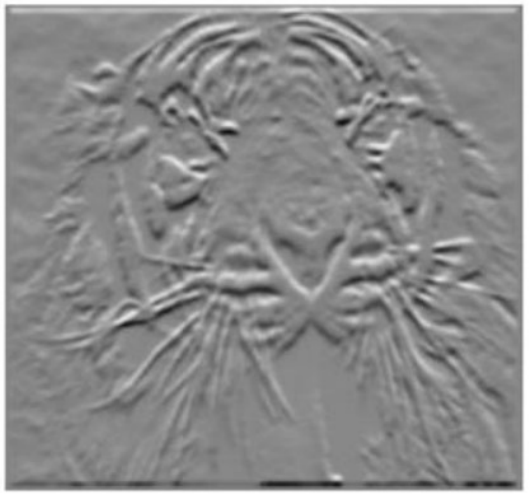
$$\frac{\partial I(x,y)}{\partial x}$$


y方向上
detect horizontal edges

$$\frac{\partial I(x,y)}{\partial y}$$

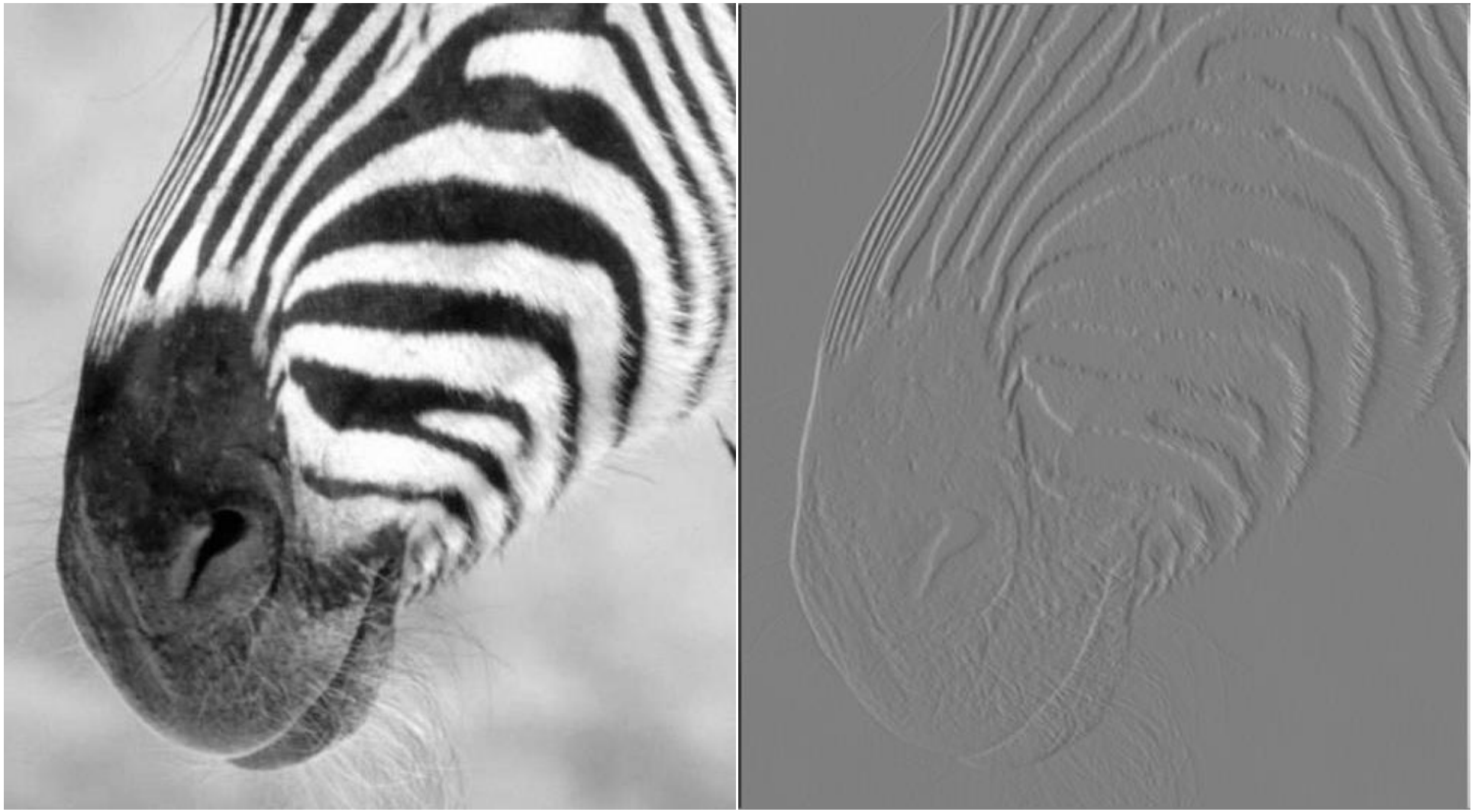
| -1 | 1 |
|---|---|





| -1 |
|---|
| 1 |

# Vertical gradients from finite differences



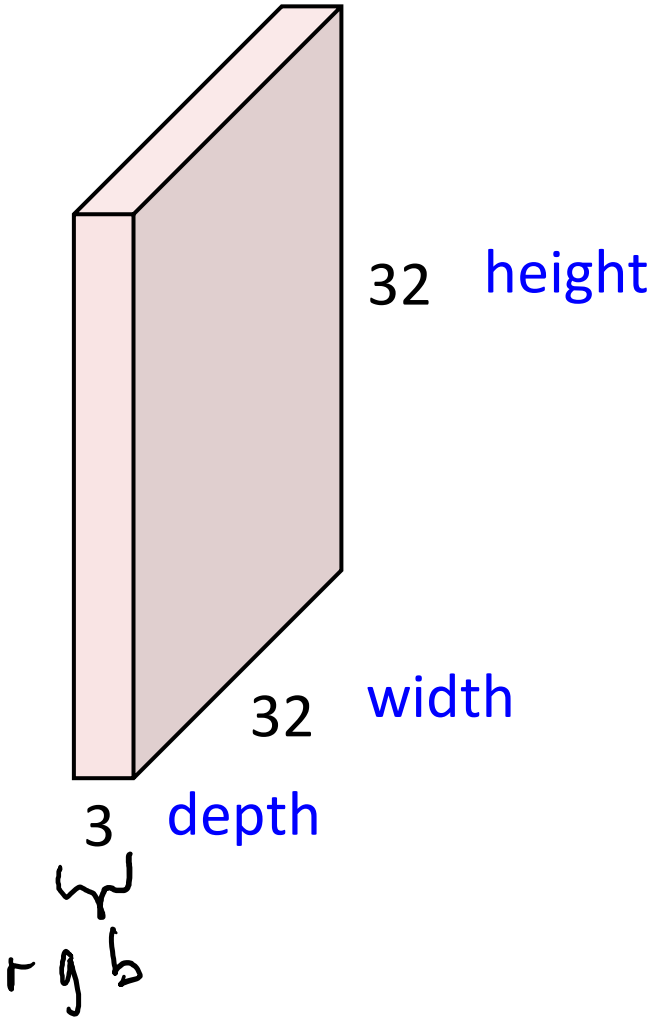


[Slide: D.A. Forsyth]

# Convolutional Neural Networks

# Convolution Layer

32x32x3 image

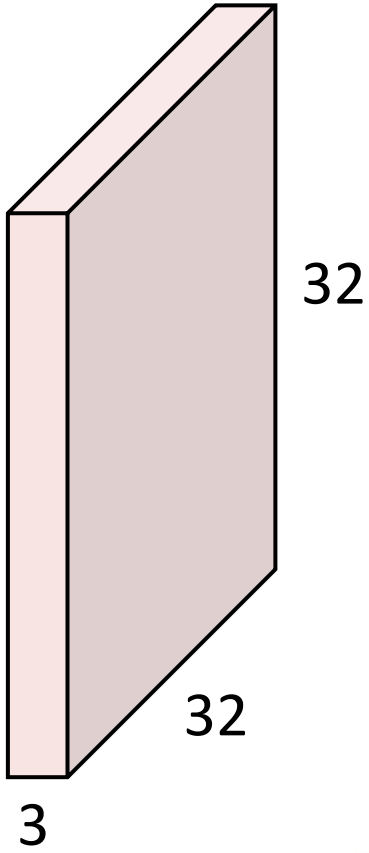




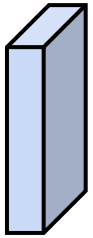

[Slide adopted from: Li, Karpathy]

# Convolution Layer

32x32x3 image




5x5x3 filter



**Convolve** the filter with the image
i.e. “slide over the image spatially, computing dot products”

filters always have to have the same depth as the input image.

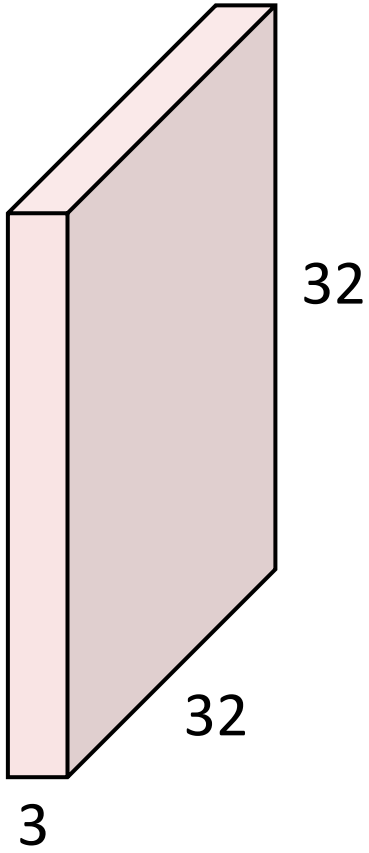


[Slides adopted from: Li, Karpathy]

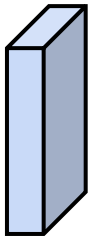
# Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image




5x5x3 filter

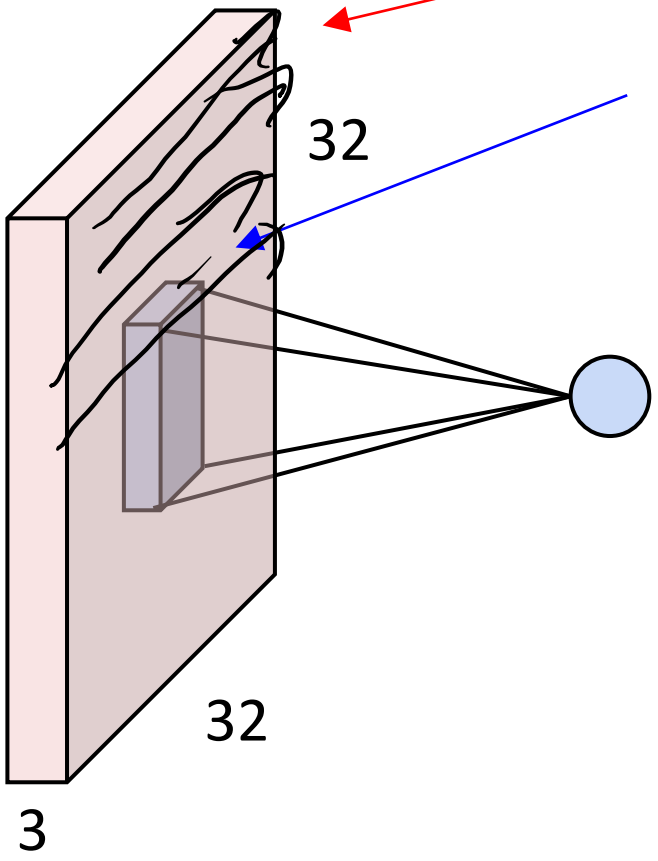


**Convolve** the filter with the image i.e. "slide over the image spatially, computing dot products"



[Slides adopted from: Li, Karpathy]

# Convolution Layer




32x32x3 image

5x5x3 filter $w$

output

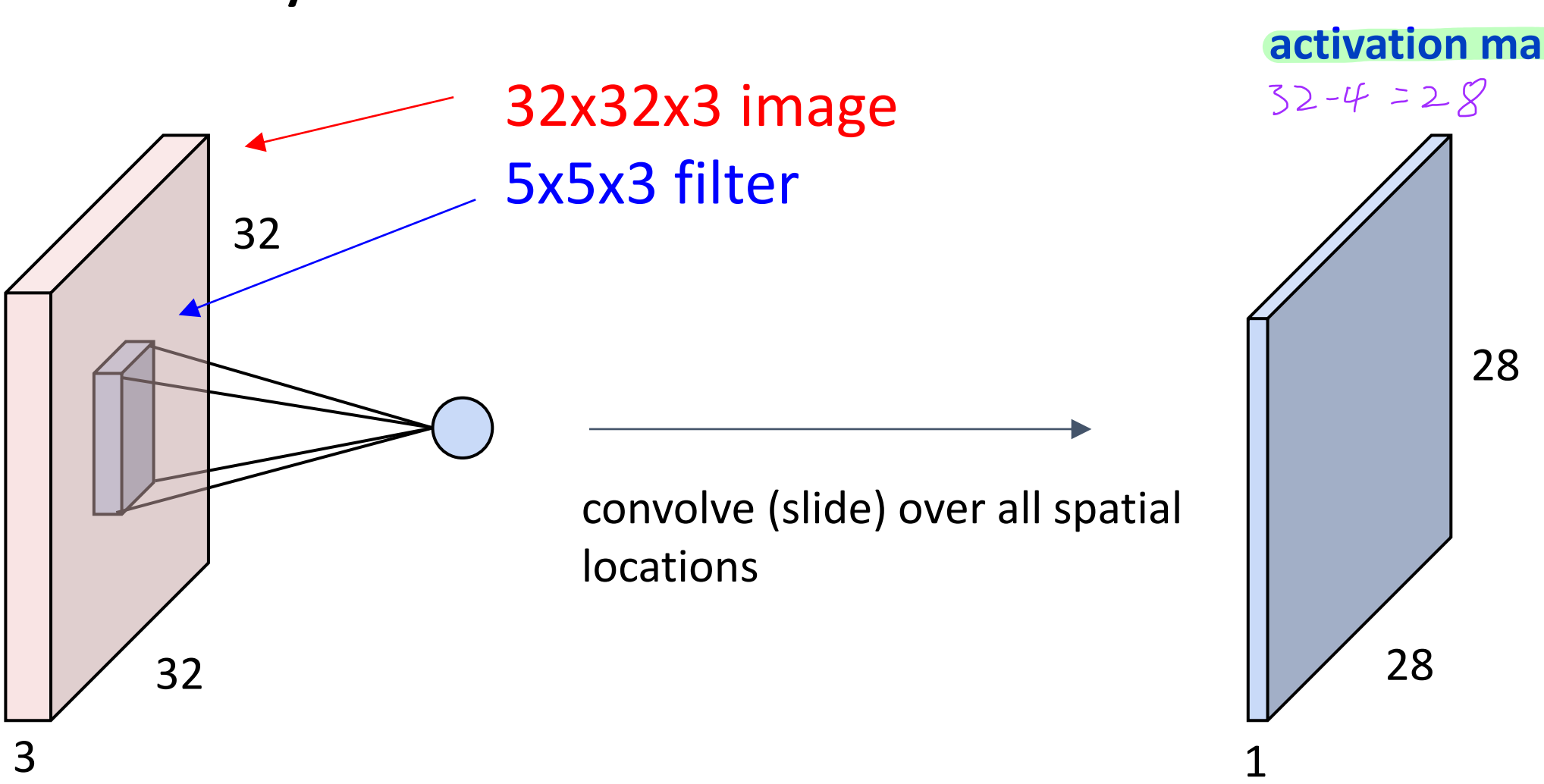
**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T z^{l-1} + b$$



[Slides adopted from: Li, Karpathy]

# Convolution Layer






[Slides adopted from: Li, Karpathy]

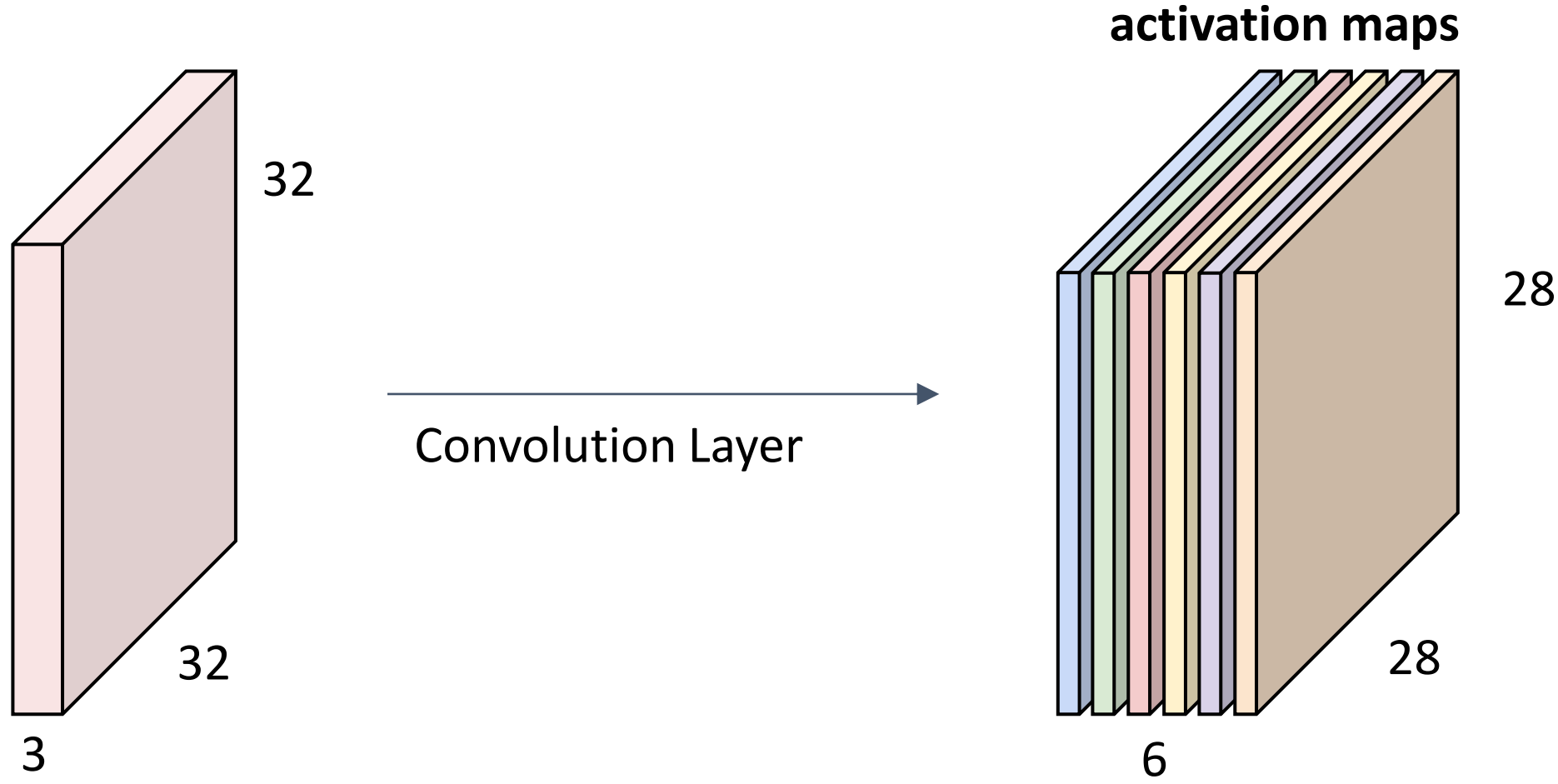
# Convolution Layer

consider a second, green filter




[Slides adopted from: Li, Karpathy]

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:




We stack these up to get a "new image" of size 28x28x6!

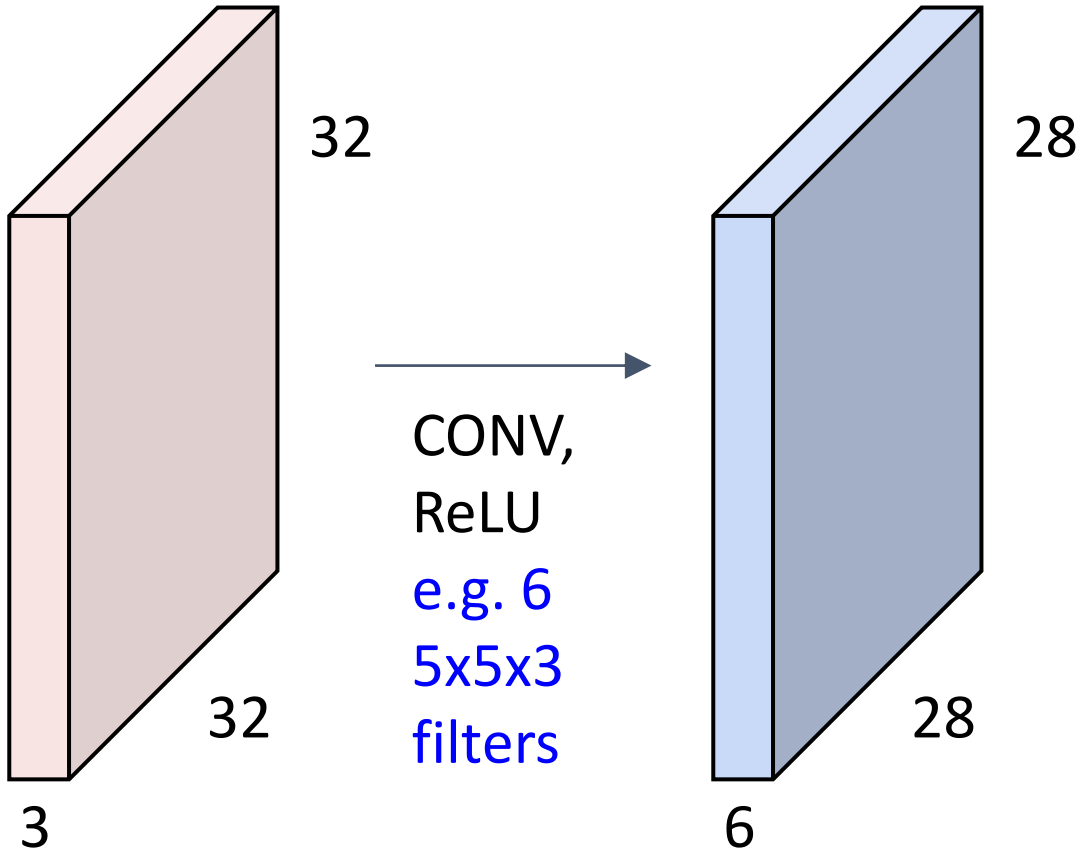
$w_i^T z^{(l-1)}$, where $i = \{1, \ldots, 6\}$



[Slides adopted from: Li, Karpathy]

**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions

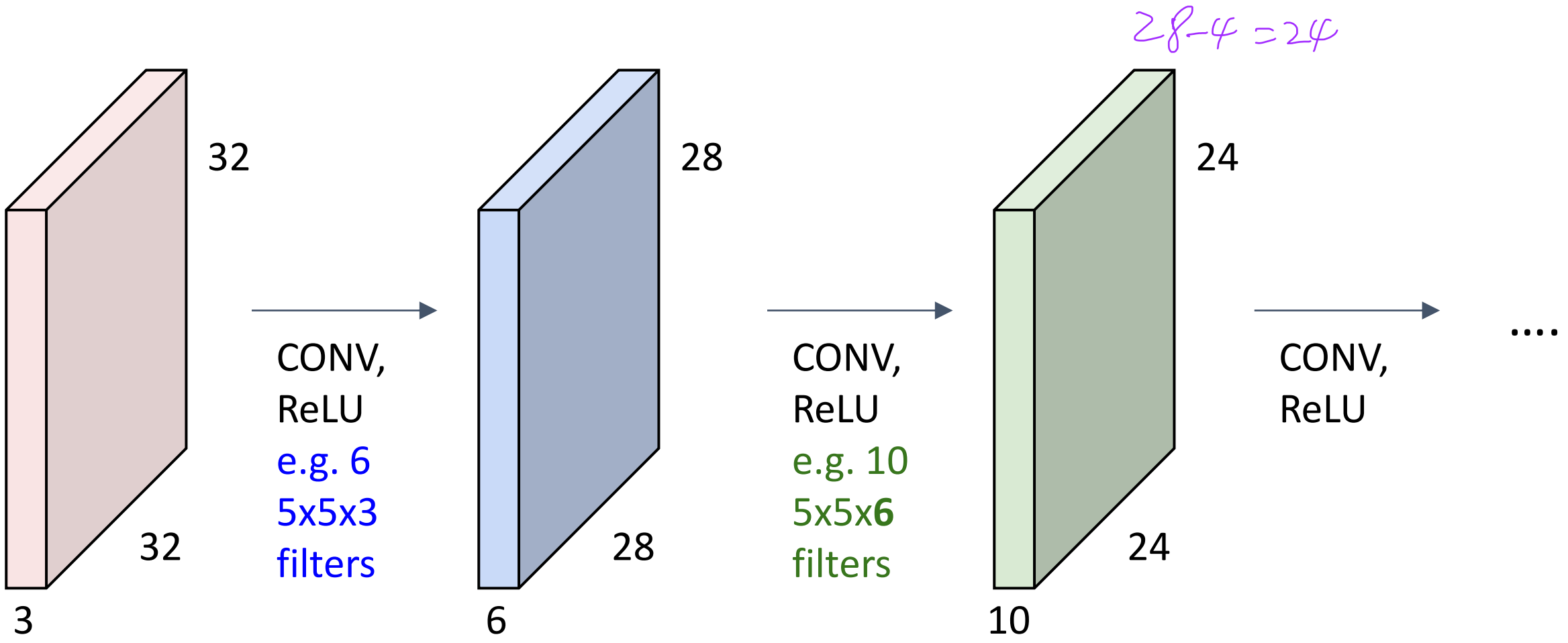
散布，穿插






[Slides adopted from: Li, Karpathy]

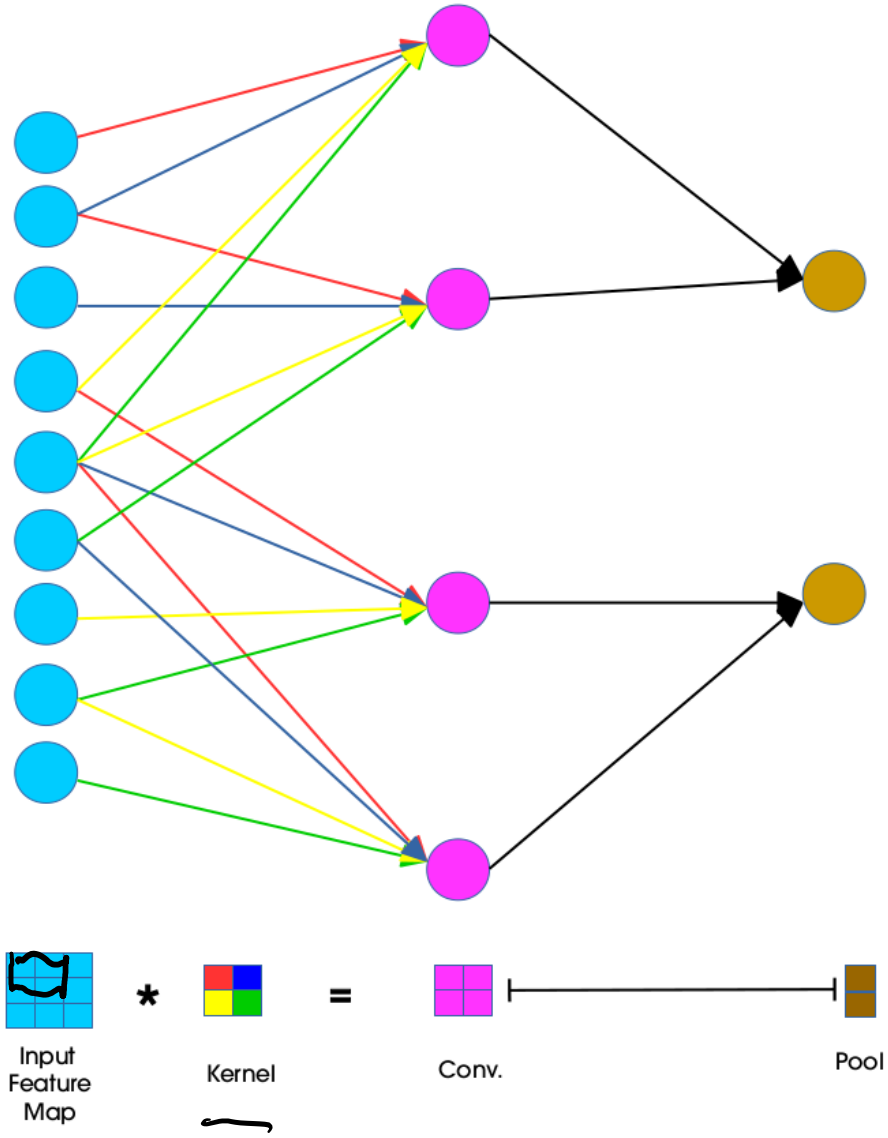
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions






[Slides adopted from: Li, Karpathy]

# Weight sharing




[image source: http://www.jefkine.com/]

# CNN-layer - Derivation

Will be on the (digital) whiteboard

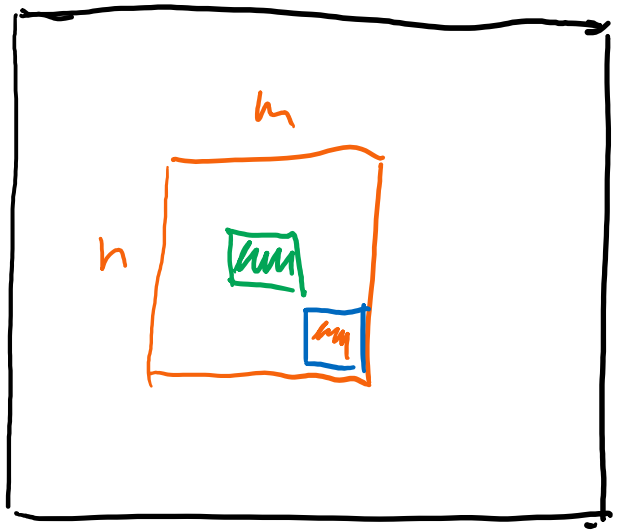
# CNN - layer, forward pass

$$z_{ij}^{(l)} = \underbrace{\omega^{(l)} * z^{(l-1)} + b}_{\sigma(\cdot)} = \left( \sum_m \sum_n \omega_{m,n}^{(l)} \, z_{i-m,j-n}^{(l-1)} \right) + b$$

$\Sigma_{i,j}^{(l)}$

↑ we will ignore this for brevity (see last week)

assume 0,0
padded 0,0

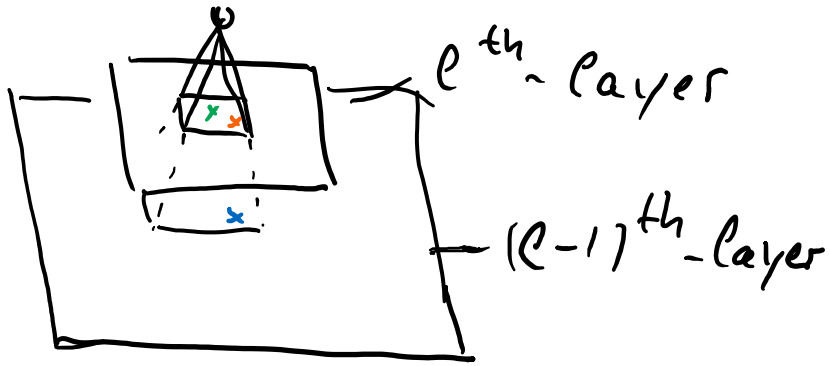



$$I'(i'_{ij}) = \omega * I(i,j)$$

$$= \sum_m \sum_n \omega_{m,n} \, I(i-m, j-n)$$

# CNN - backward pass

$$\delta_{i,j}^{(l-1)} = \frac{\partial C}{\partial z_{i,j}^{(l-1)}} = \sum_{i'} \sum_{j'} \frac{\partial C}{\partial z_{i',j'}^{(l)}} \frac{\partial z_{i',j'}^{(l)}}{\partial z_{i,j}^{(l-1)}}$$




$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} \cdot \frac{\partial}{\partial z_{i,j}^{(l-1)}} \left( \sum_{m} \sum_{n} w_{m,n}^{(l)} \, z_{i'-m,j'-n}^{(l-1)} \right) + b$$

$$w_{0,0} z_{i'-0,j'-0} + \dots + w_{m,n} z_{i'-m,j'-n} + \dots$$

$$i = i'-m \wedge j = j'-n$$

$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} \cdot w_{m,n}^{(l)} \Rightarrow \text{if } i = i'-m \wedge j = j'-n \Rightarrow m = i'-i \wedge n = j'-j$$

$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} \cdot w_{i'-i,j'-j}^{(l)} \Rightarrow \text{iter flips weights}$$

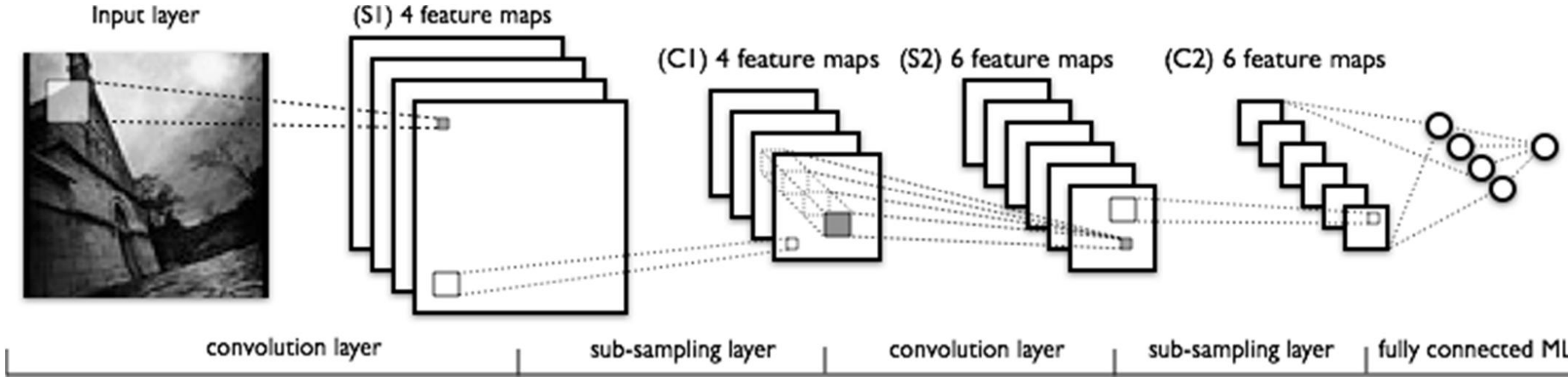
$$= \delta^{(l)} * \mathrm{ROT}_{180}(w^{(l)})$$

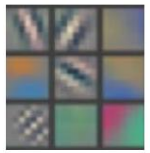
# CNN - parameter update

$$\frac{\partial C}{\partial w^{(l)}_{m,n}} = \sum_i \sum_j \frac{\partial C}{\partial z^{(l)}_{i,j}} \frac{\partial z^{(l)}_{i,j}}{\partial w^{(l)}_{m,n}} = \sum_i \sum_j \delta^{(l)}_{i,j} \frac{\partial}{\partial w^{(l)}_{m,n}} \left( \sum_m \sum_n w^{(l)}_{m,n} z^{(l-1)}_{i-m,j-n} \right) + b$$

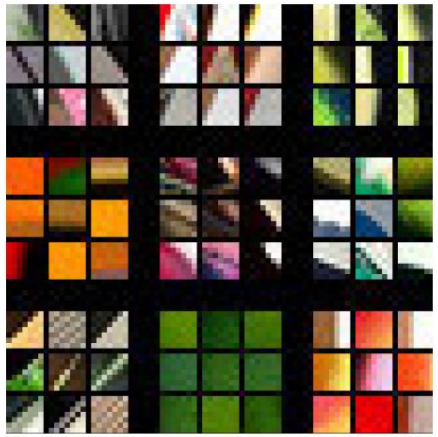
$$= \sum_i \sum_j \delta^{(l)}_{i,j} \cdot z^{(l-1)}_{i-m,j-n} = \delta^{(l)} * ROT_{180}\left(z^{(l-1)}\right)$$

# Stopped here – 3/19/2020

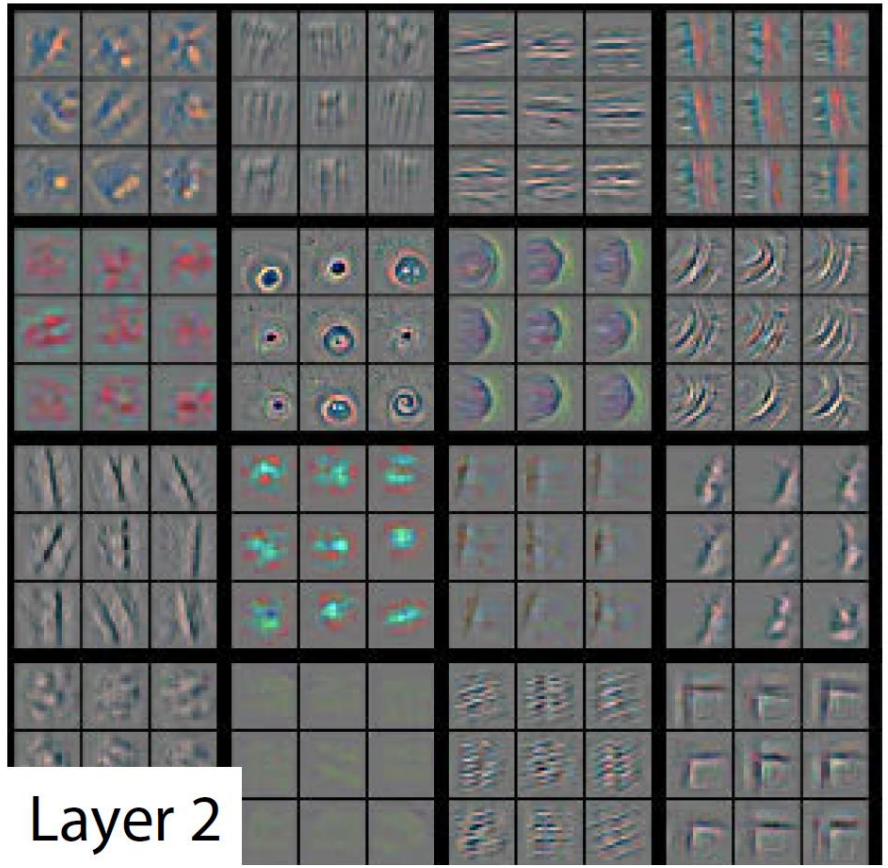


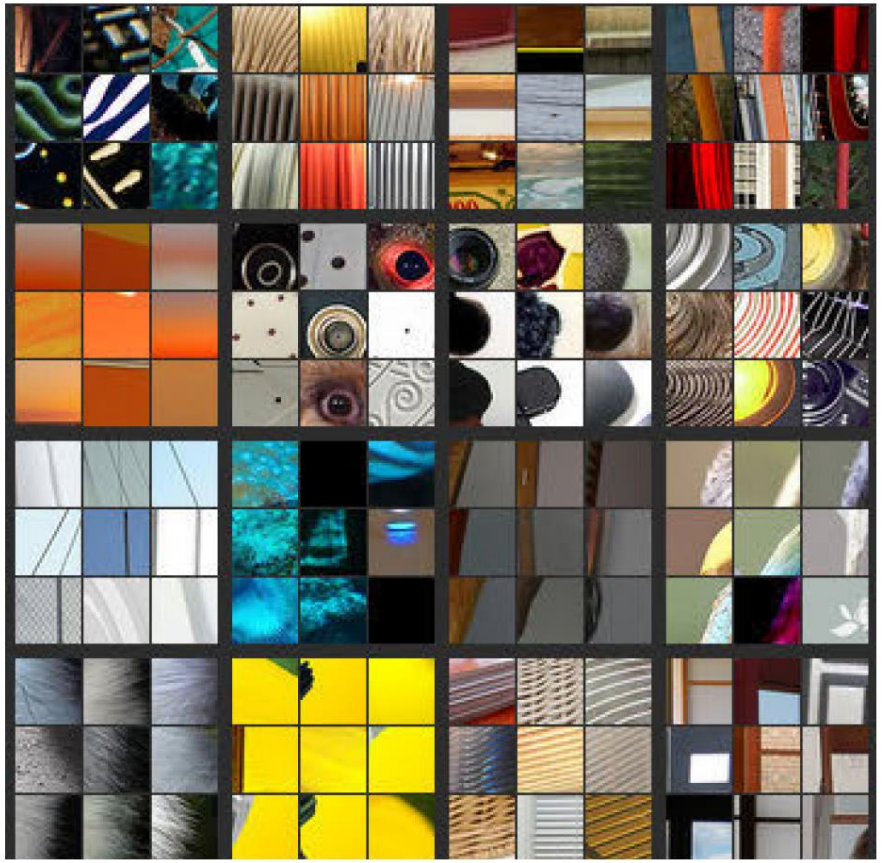
Layer 1

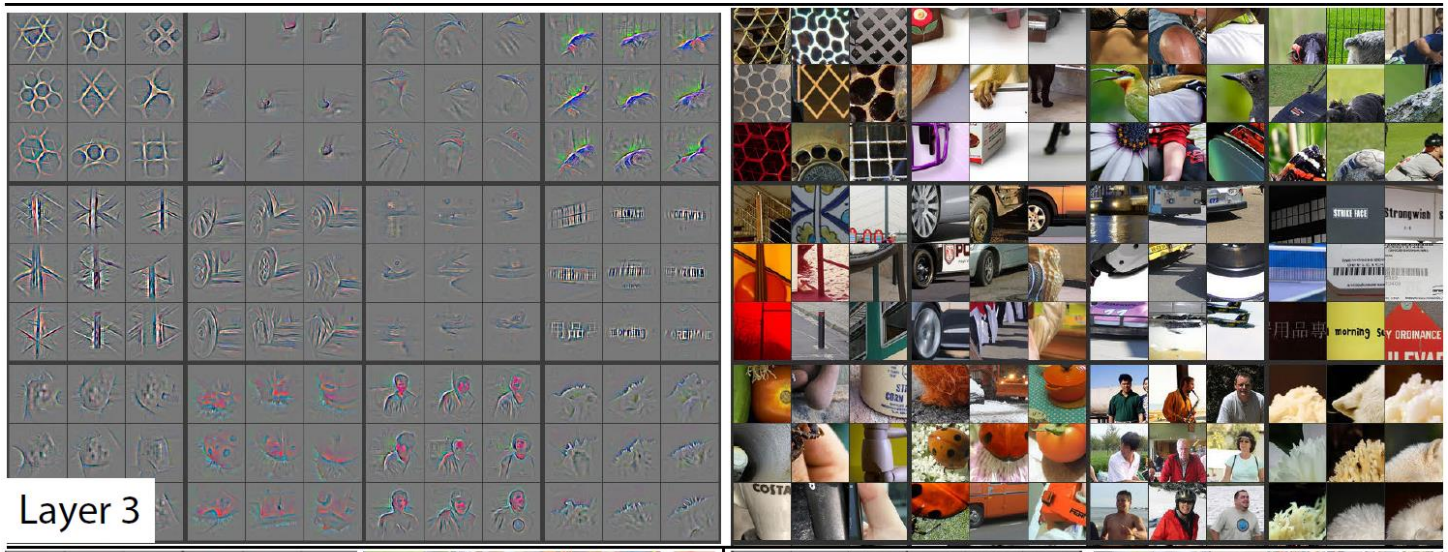


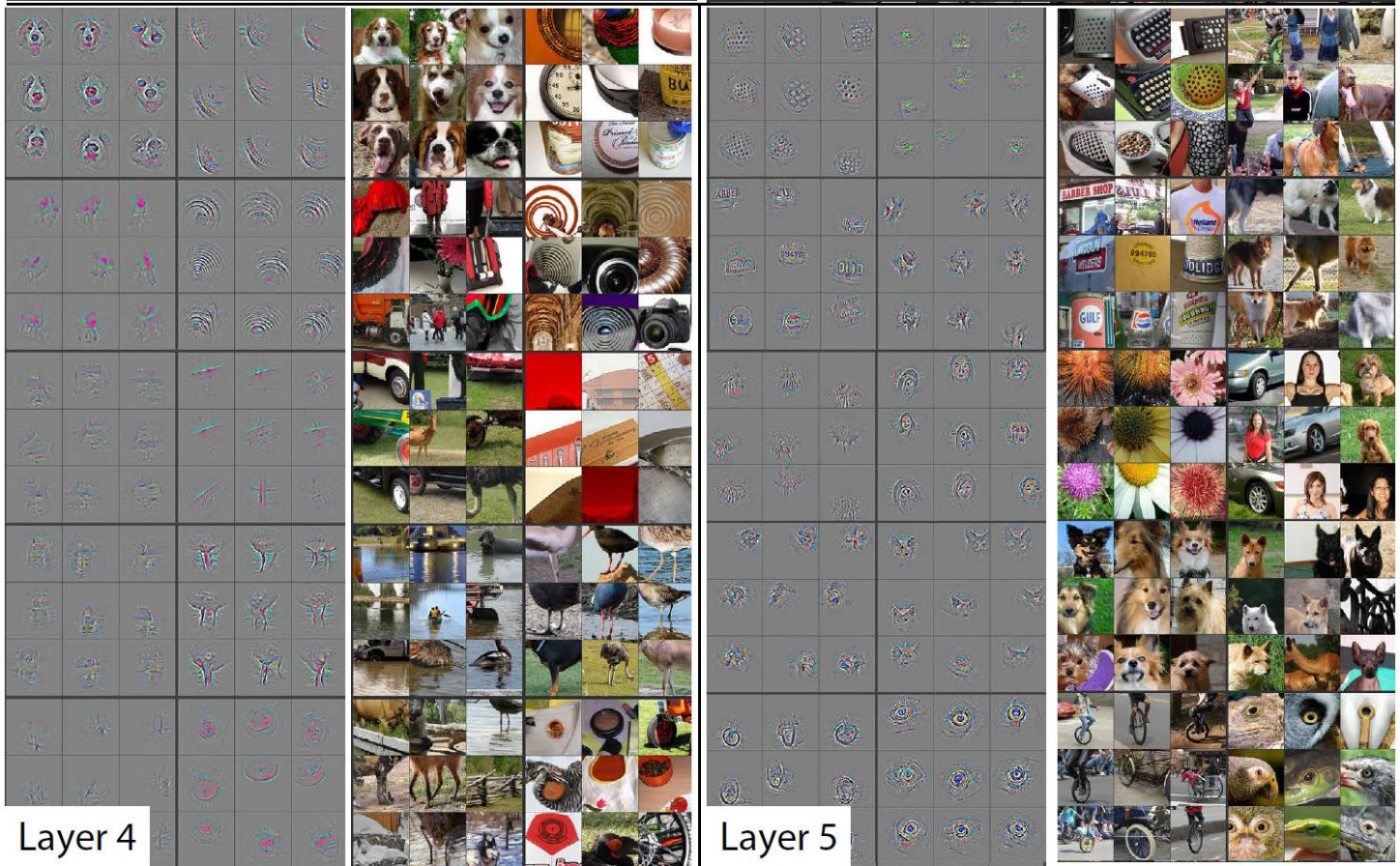
[Zeiler & Fergus ]



Layer 2



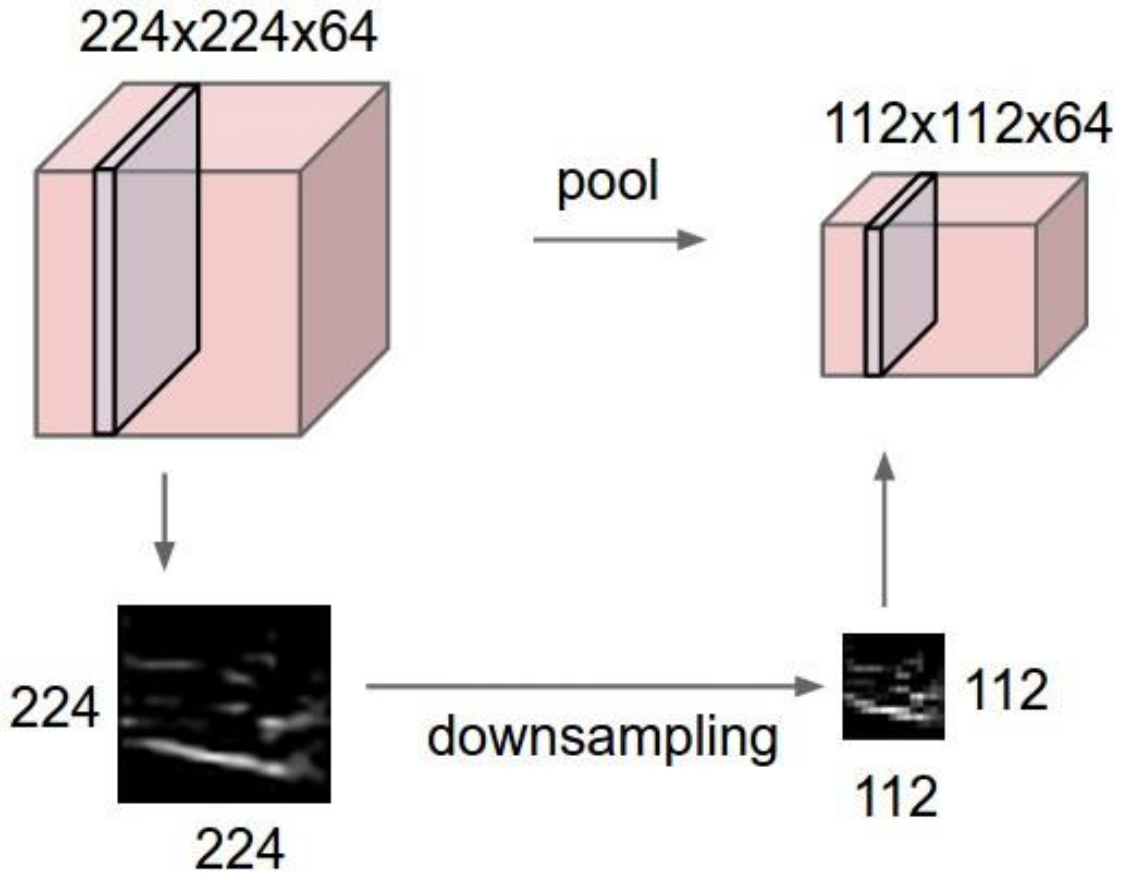



[Zeiler & Fergus ]

# Pooling layer

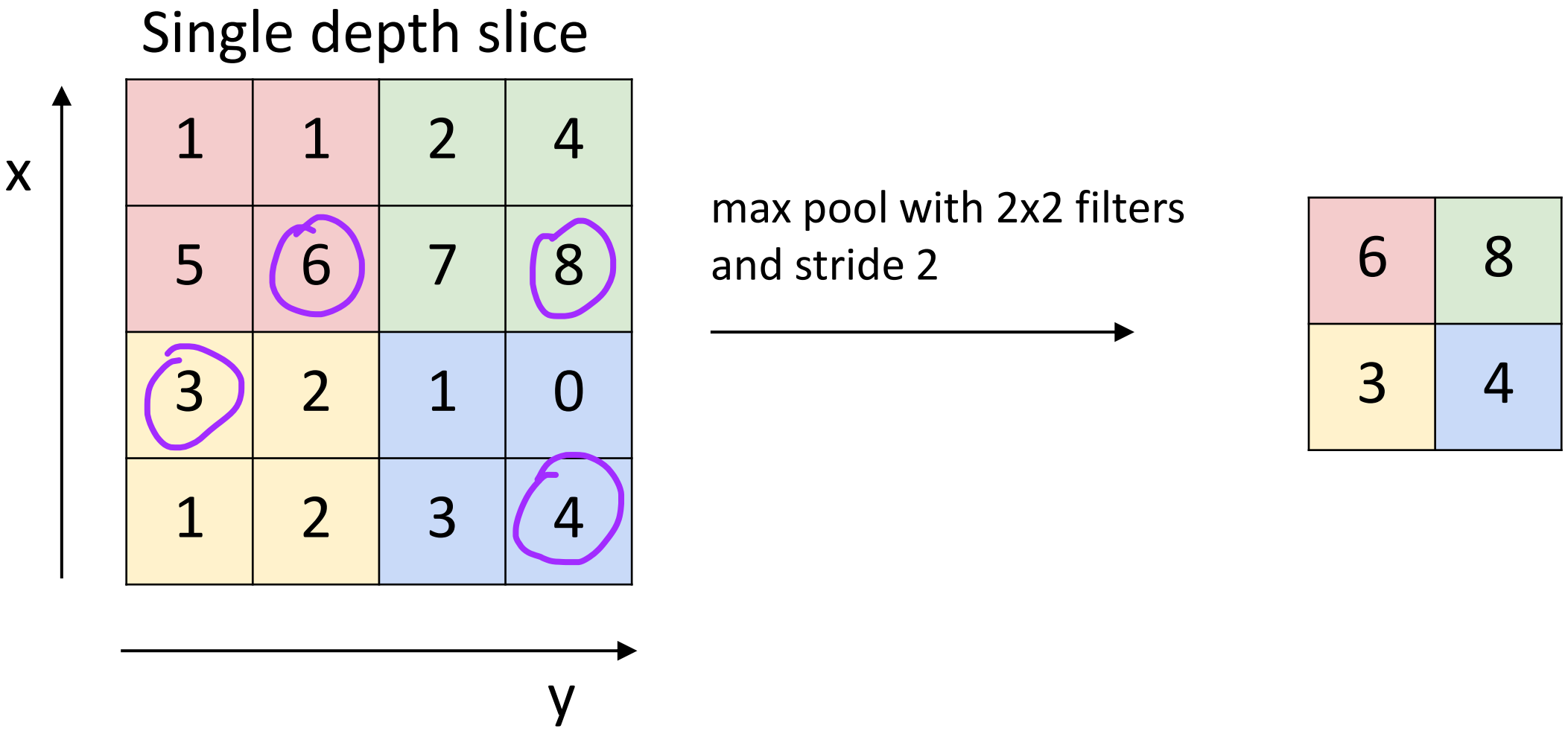
- makes the representations smaller and more manageable
- operates over each activation map independently:





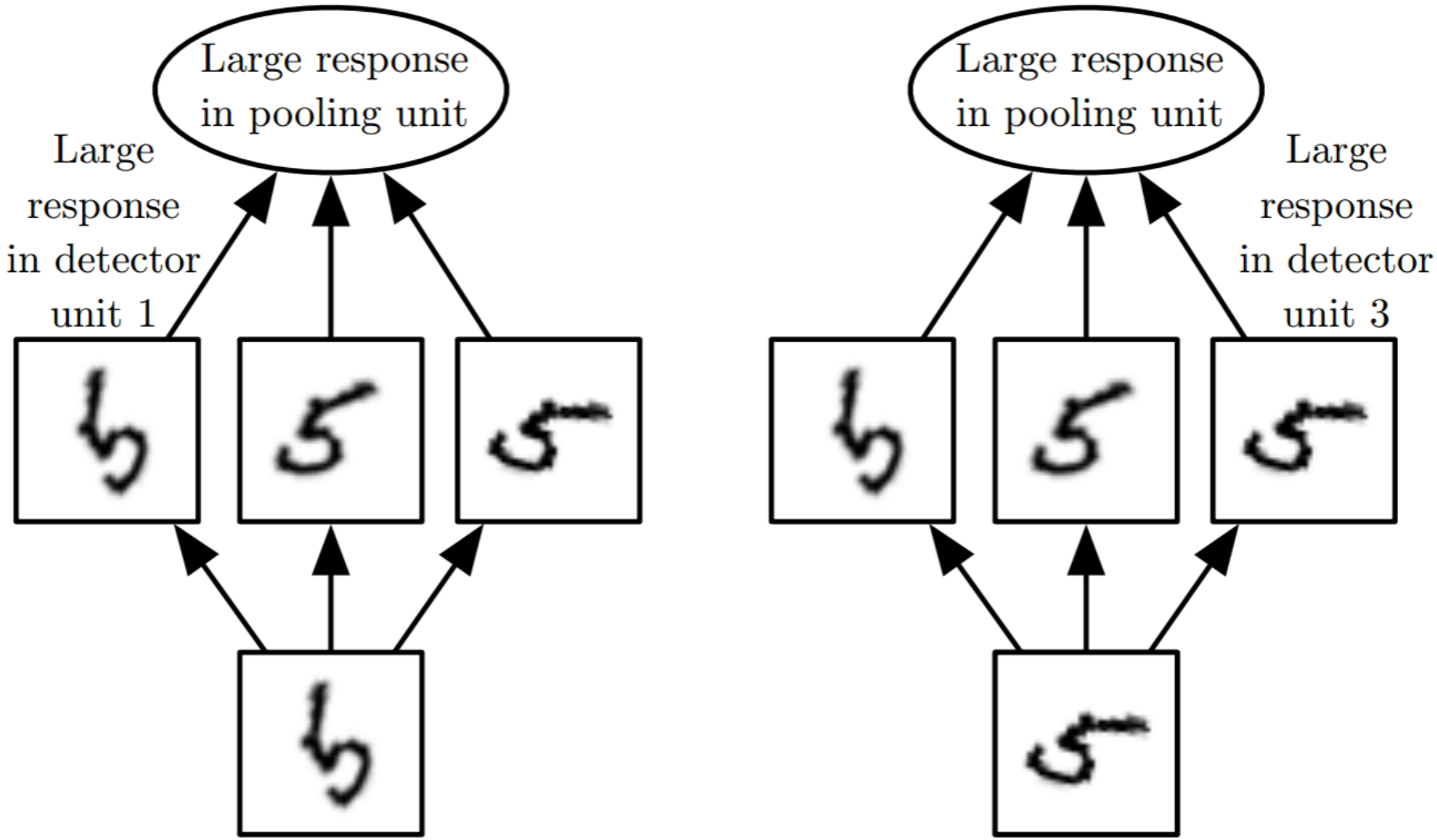

[Slides adopted from: Li, Karpathy]

# MAX POOLING





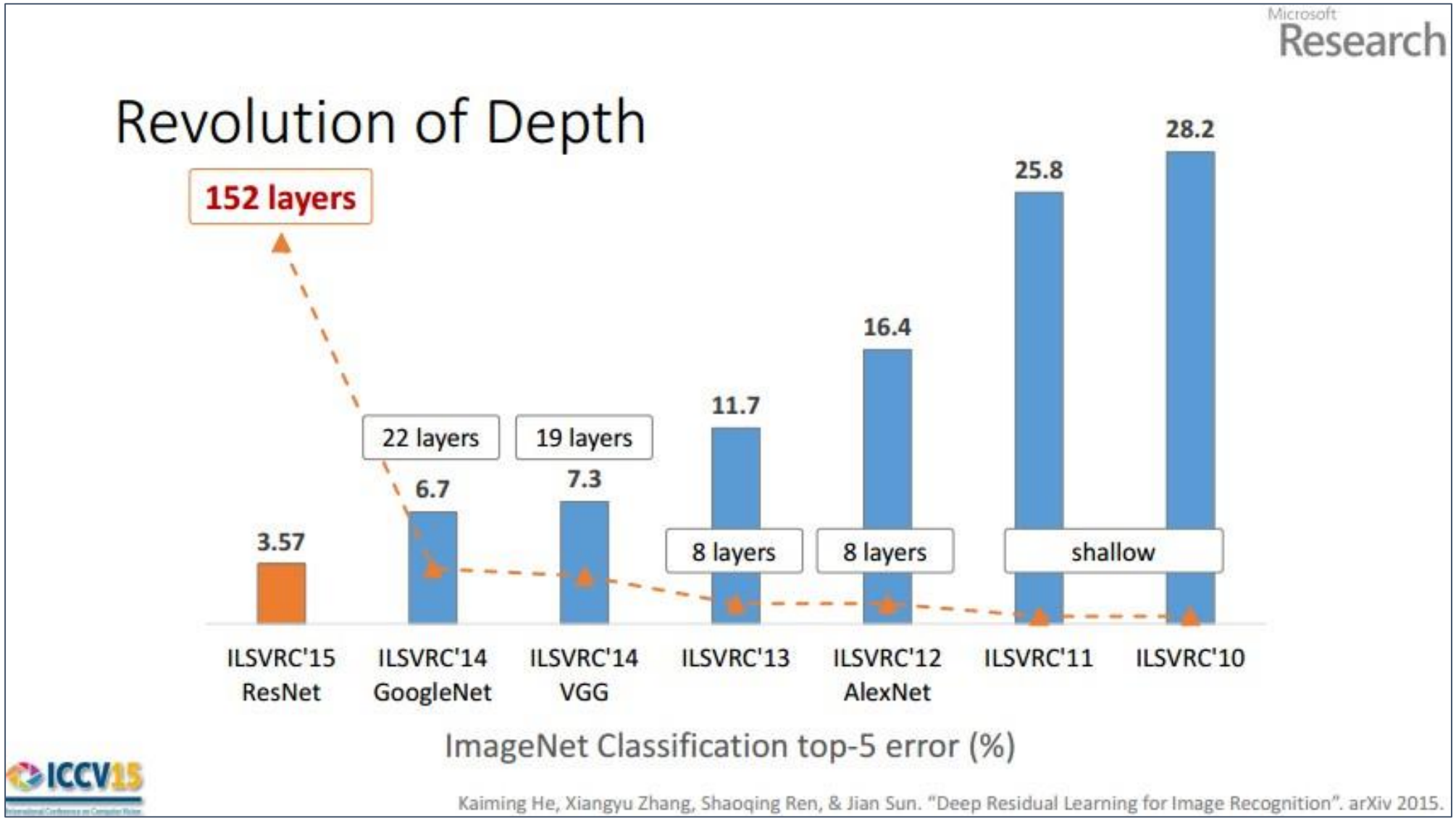

[Slides adopted from: Li, Karpathy]




[image source: deeplearningbook.org]

# Max-Pooling (Forward & Backward pass)

Will be on the (digital whiteboard)




(slide from Kaiming He's recent presentation)

# Case Study: VGG – more depth, fewer parameters

Smaller filters

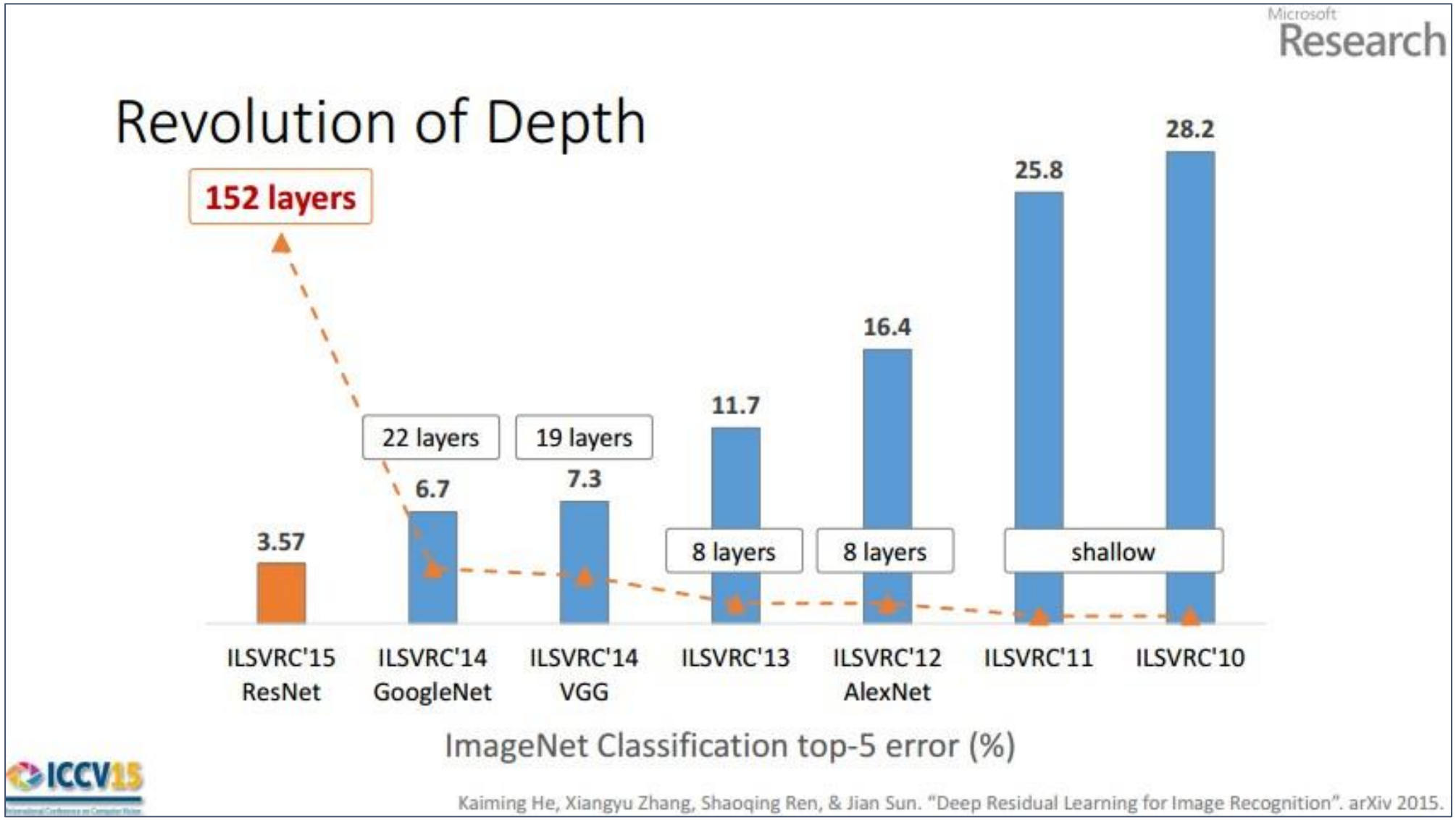
More layers

Due to depth:

- Large receptive field
- Despite smaller filters

Due to smaller filters:

- Fewer parameters



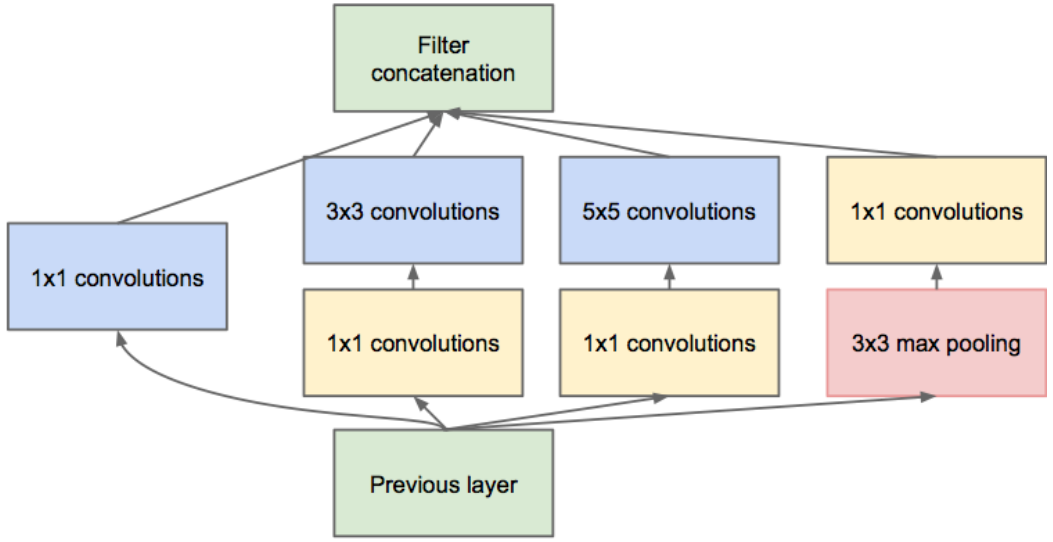

(slide from Kaiming He's recent presentation)

# Case Study: GoogLeNet

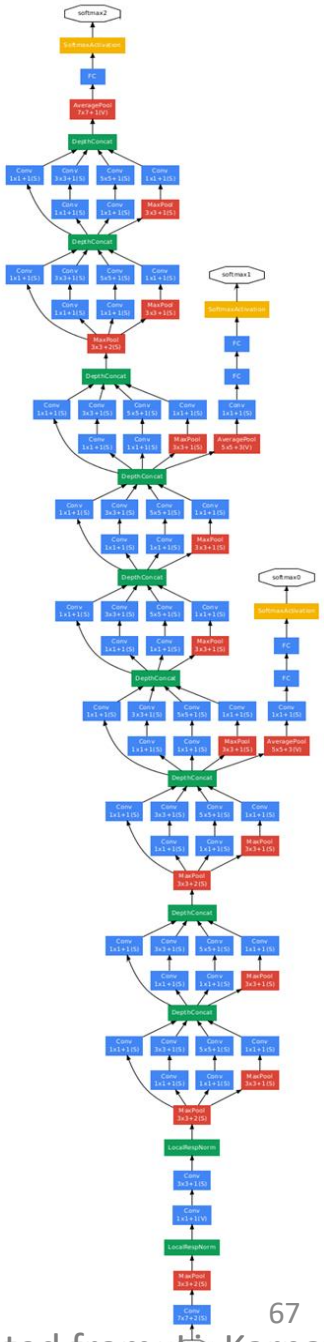
*[Szegedy et al., 2014]*

Even deeper network with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters
  - 12x less than AlexNet
  - ILSVRC 2014 winner (6.7% top 5 error)




Inception module

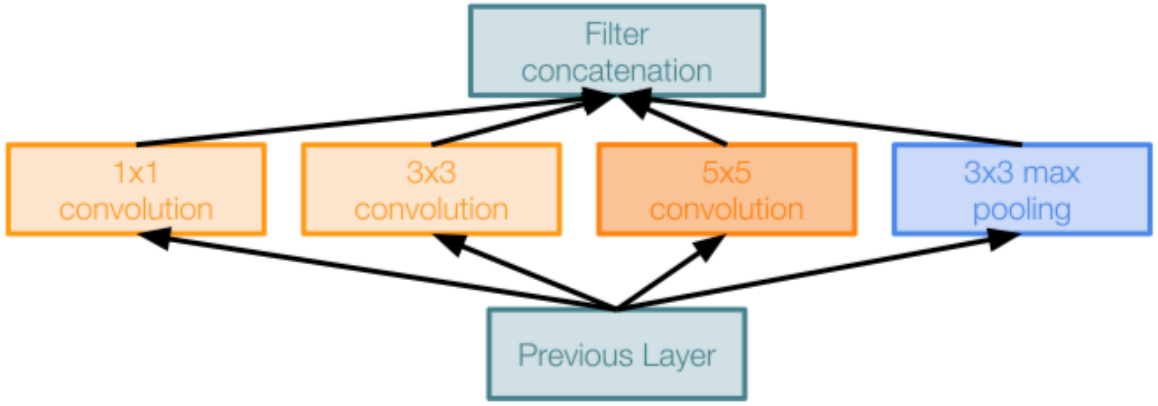




[Slides adopted from: Li, Karpathy]

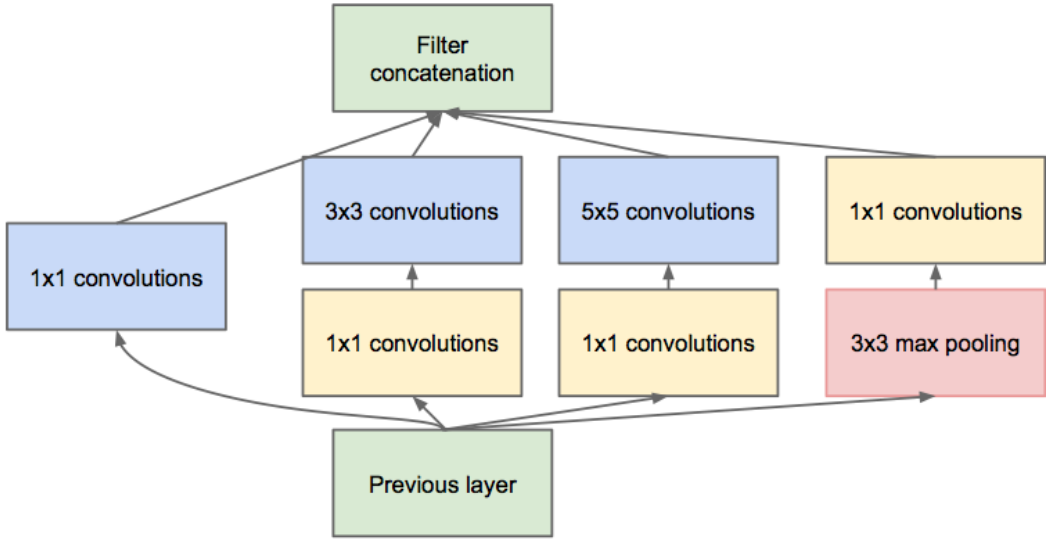
# Computational aspects

## Naïve Inception




Continuous increase in dimensionality

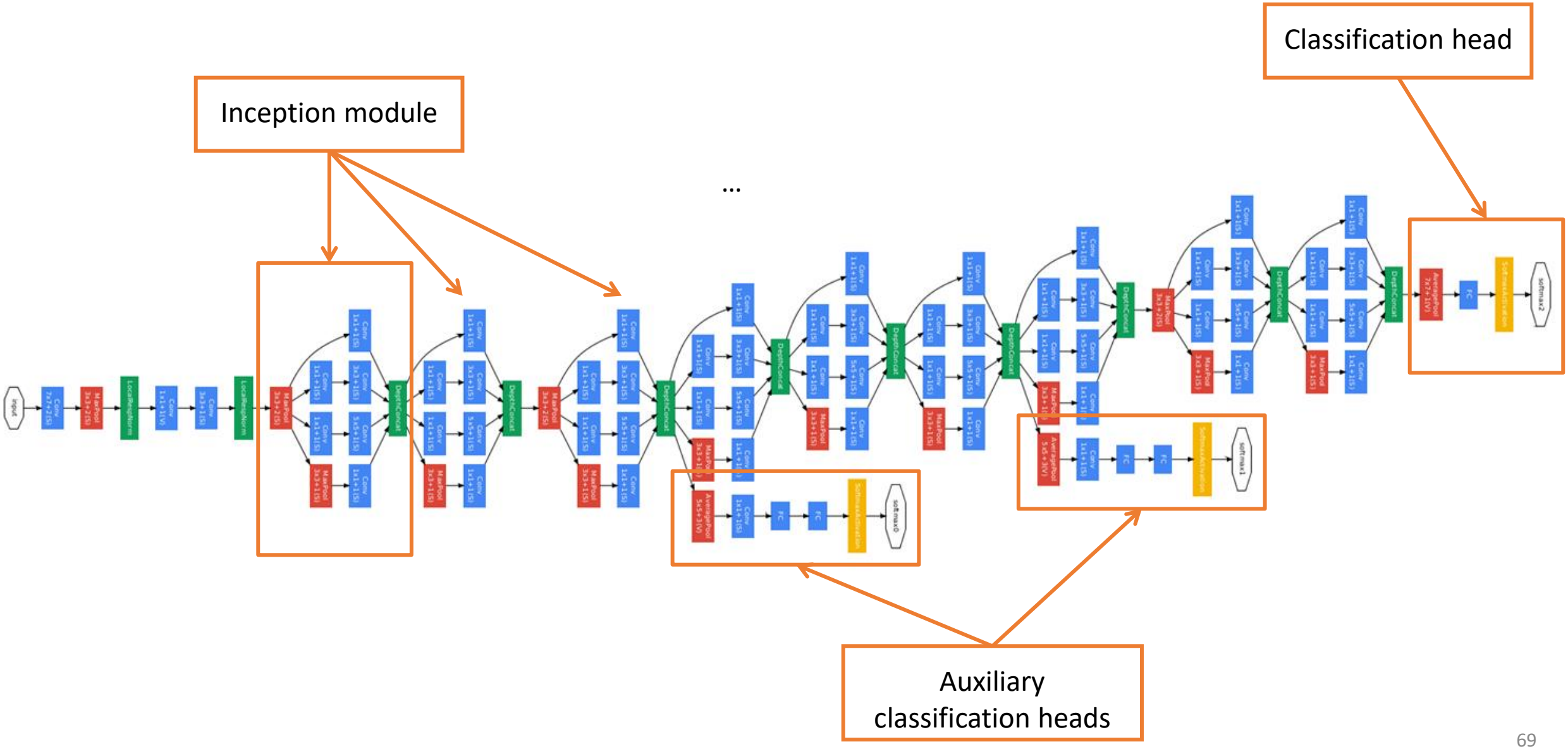
## Final Inception module

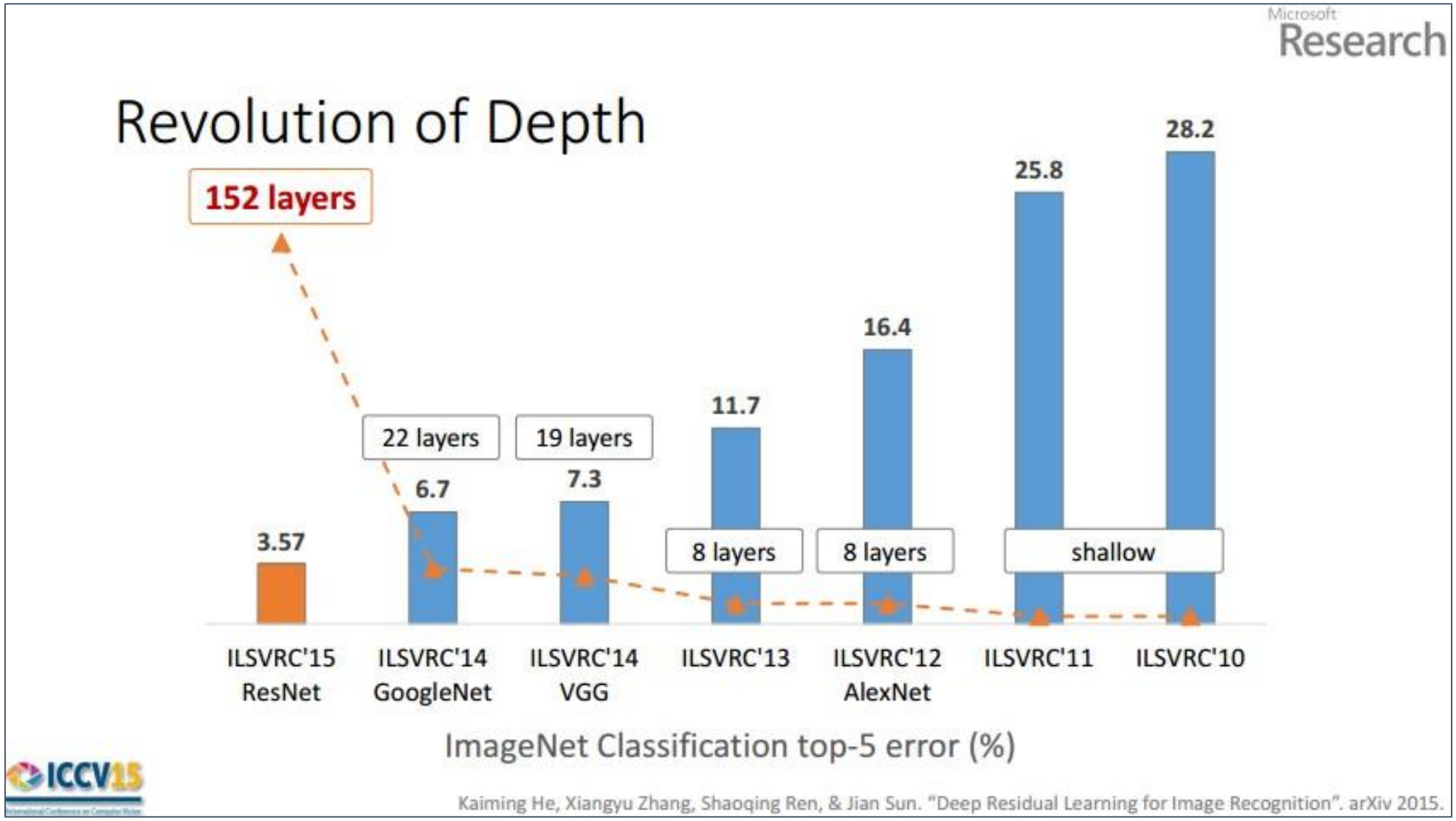



Dimensionality reduction via 1x1 convolutions

# Case Study: GoogLeNet

*[Szegedy et al., 2014]*



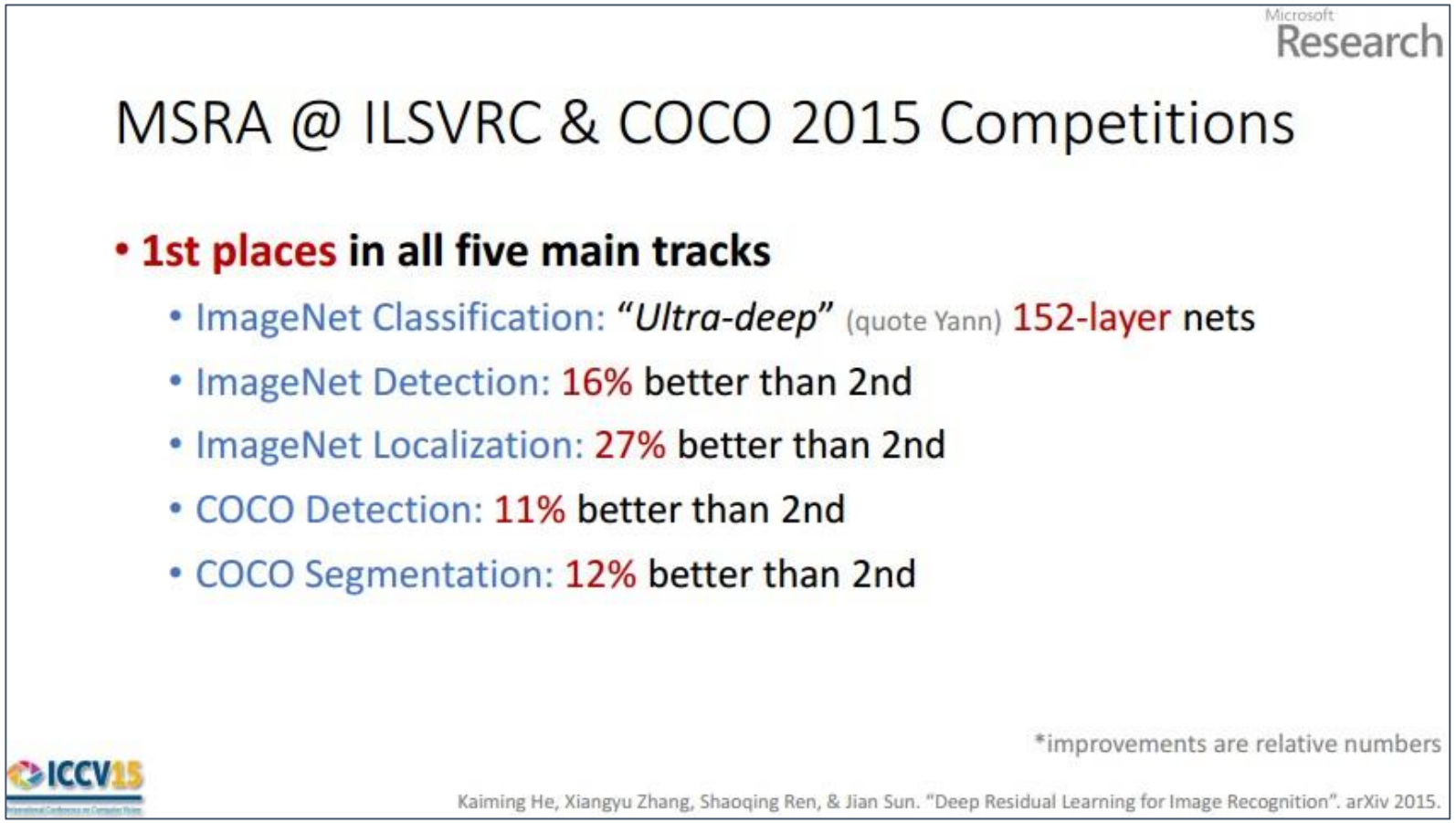



(slide from Kaiming He's recent presentation)

# Case Study: ResNet

*[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



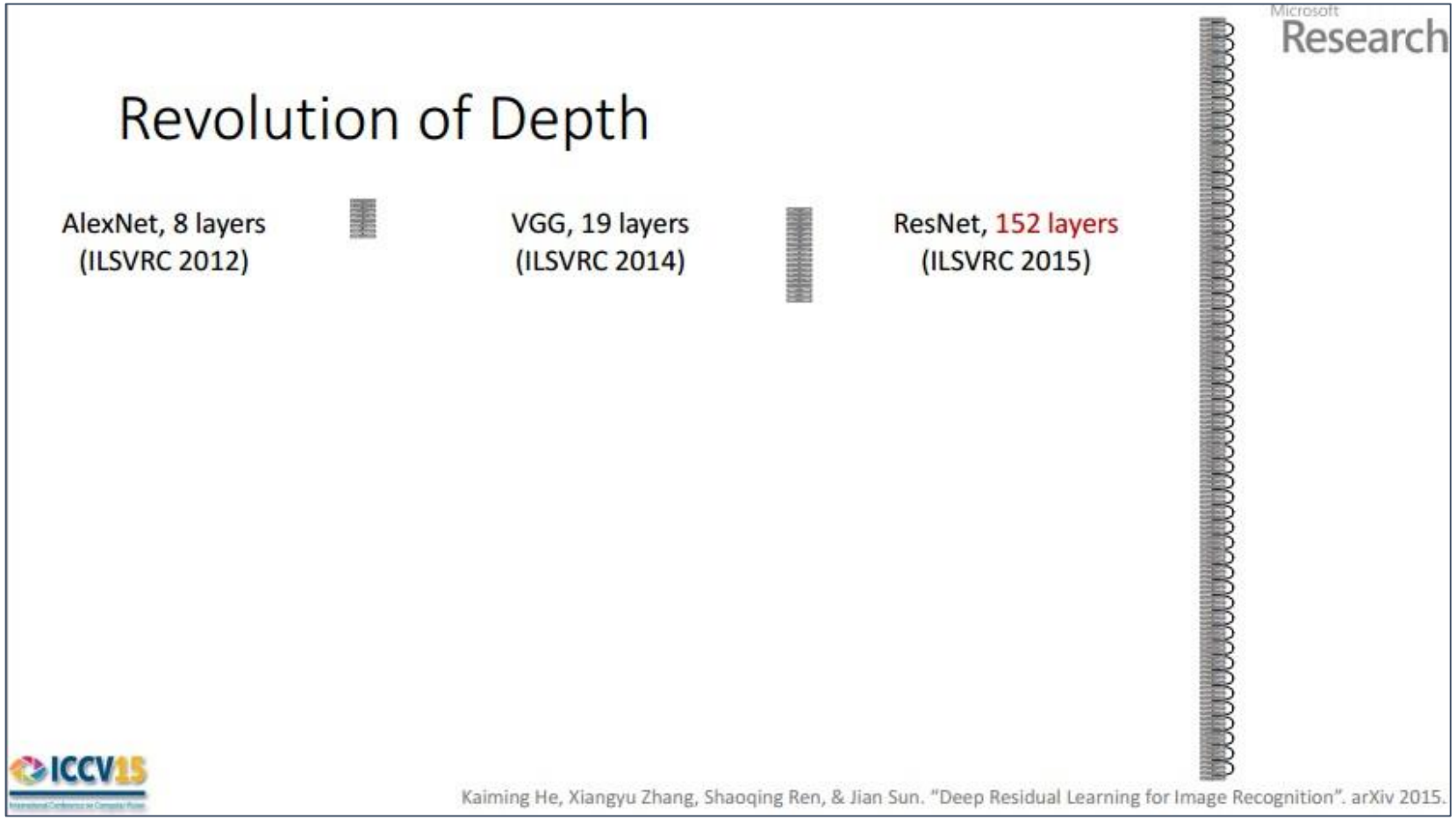

Slide from Kaiming He's recent presentation

https://www.youtube.com/watch?v=1PGLj-uKT1w

# Case Study: ResNet

*[He et al., 2015]*

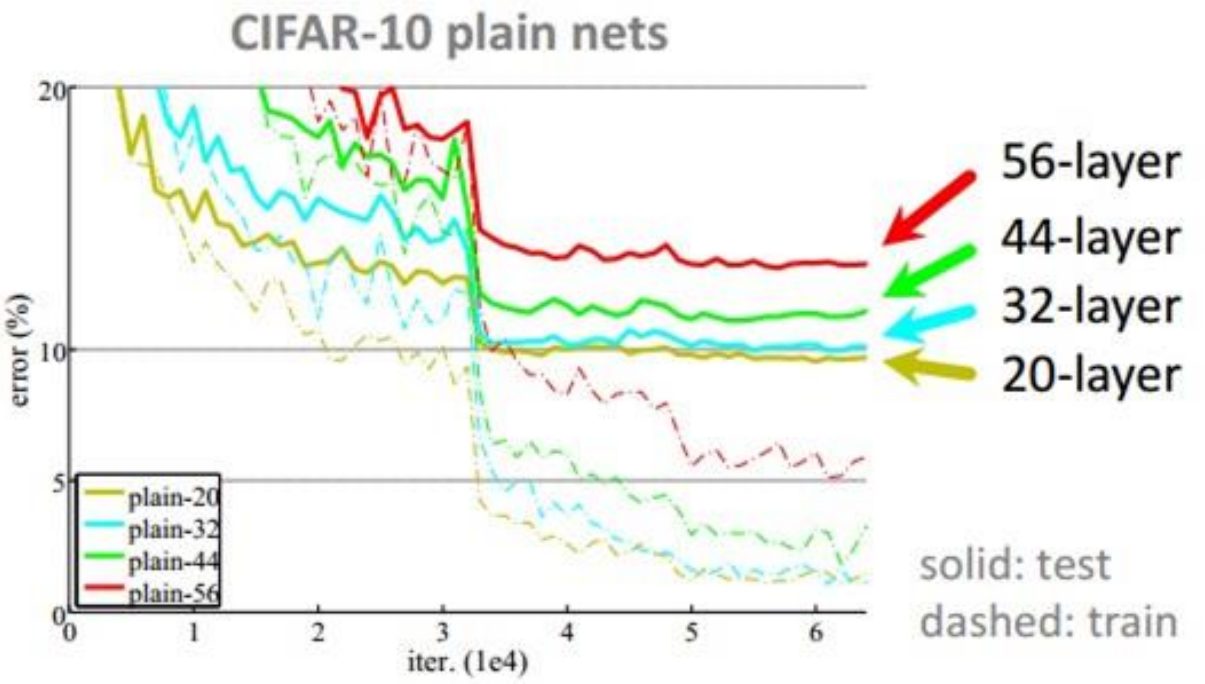
ILSVRC 2015 winner (3.6% top 5 error)




2-3 weeks of training on 8 GPU machine

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

(slide from Kaiming He's recent presentation)

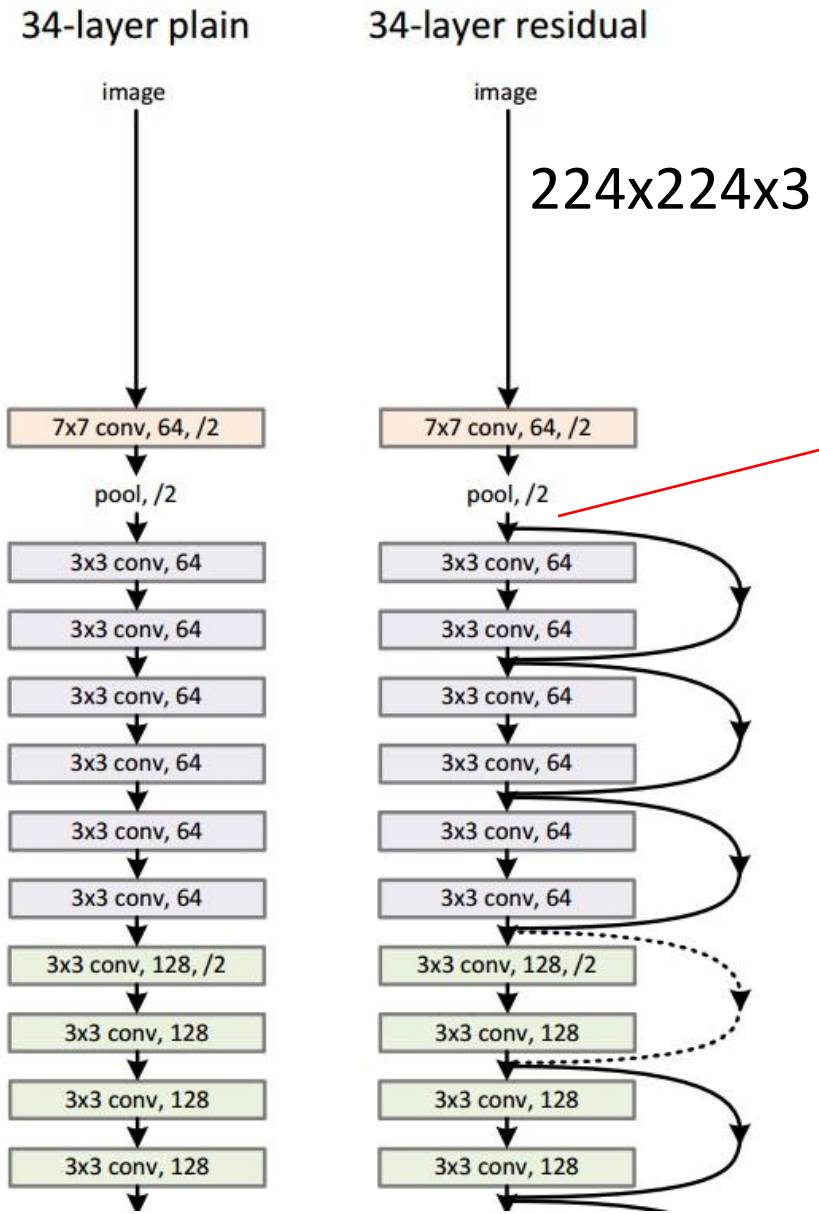



Failure to optimize

- A deeper network should perform at least as good as a shallow architecture
- “Proof”: take shallow network, copy layers and set the output to the identity map.

# Case Study: ResNet

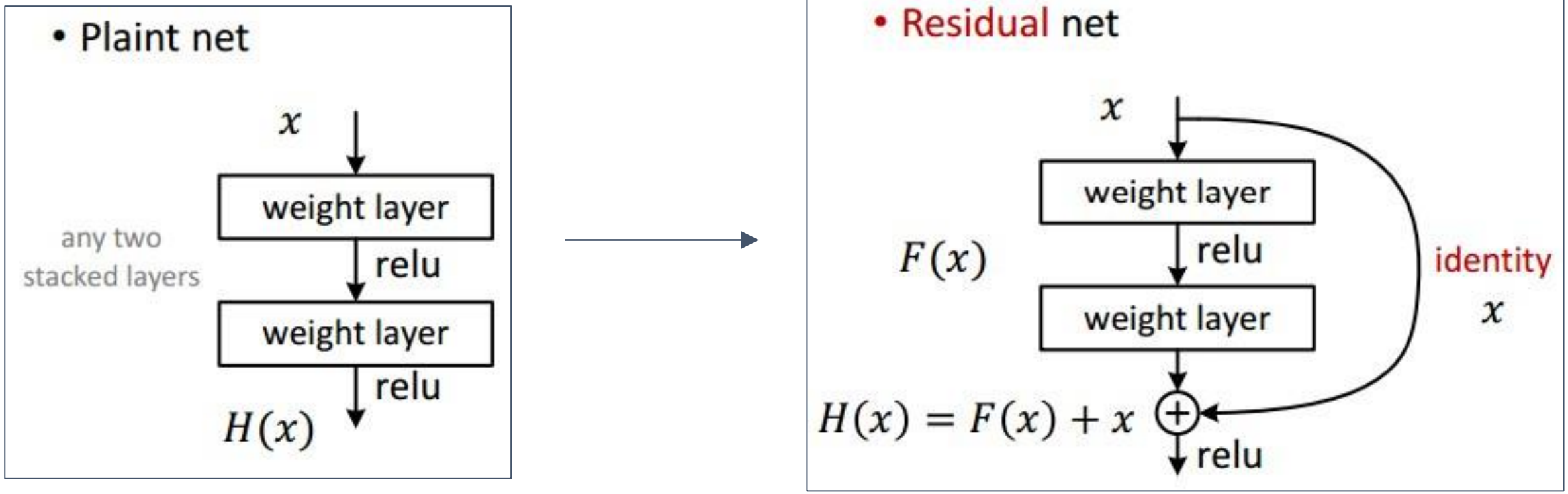
*[He et al., 2015]*



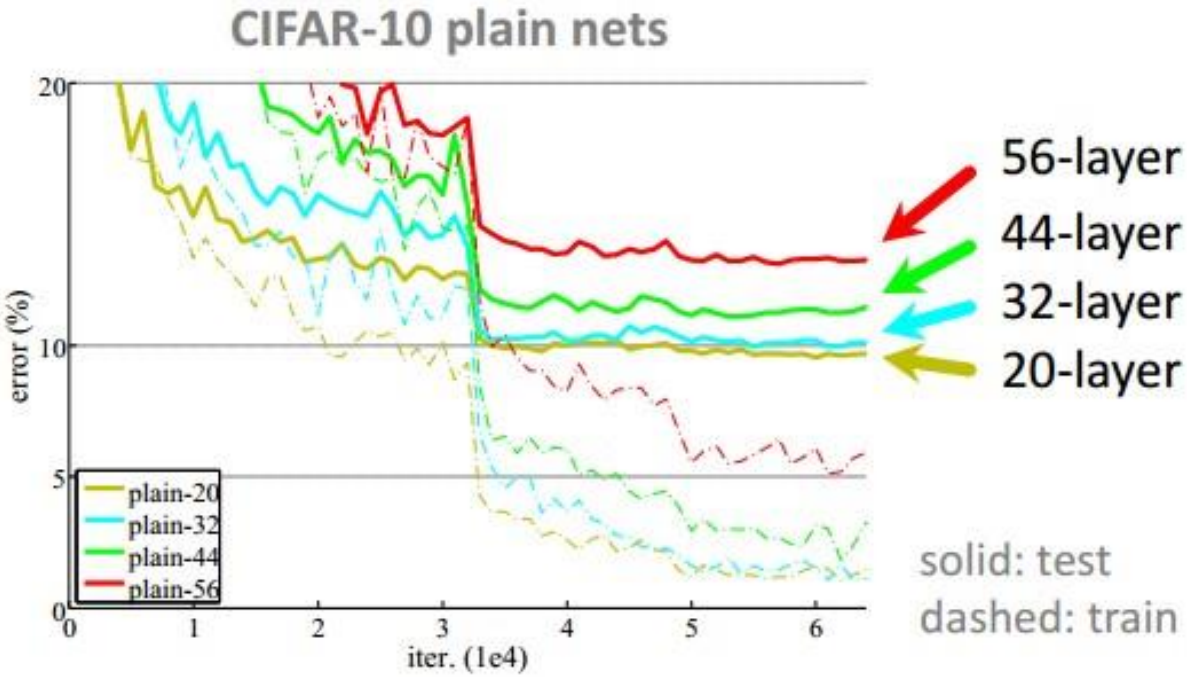

spatial dimension only 56x56!

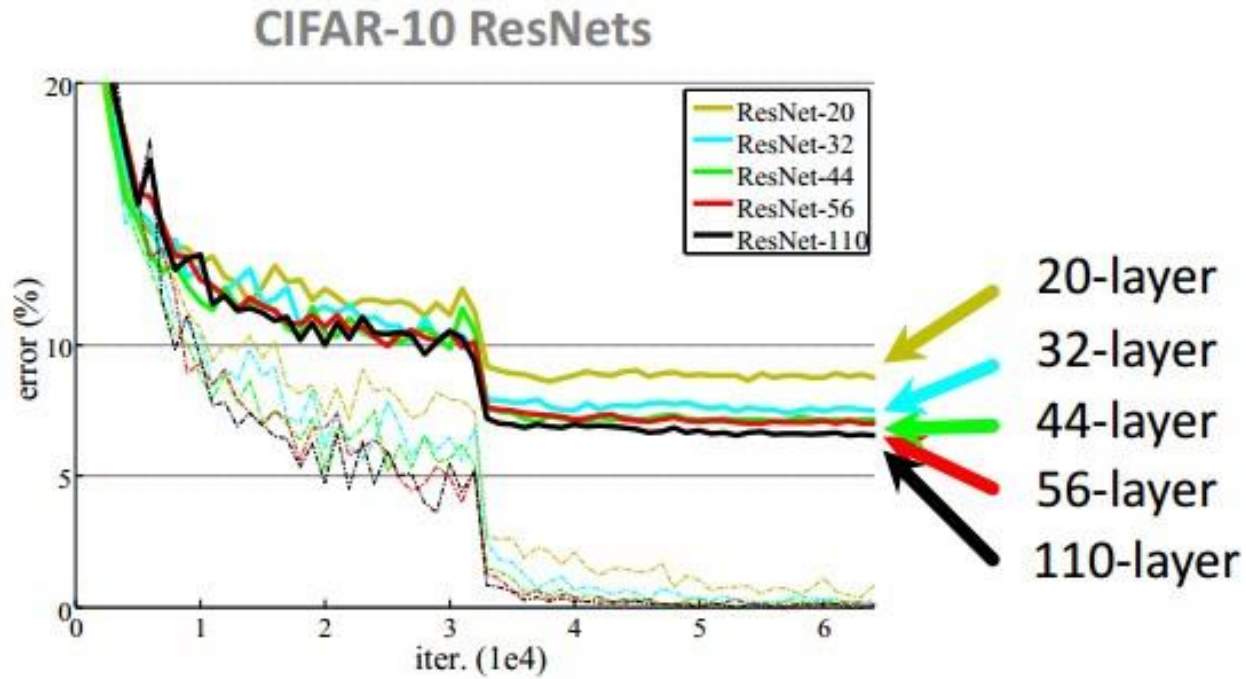
# Case Study: ResNet

*[He et al., 2015]*



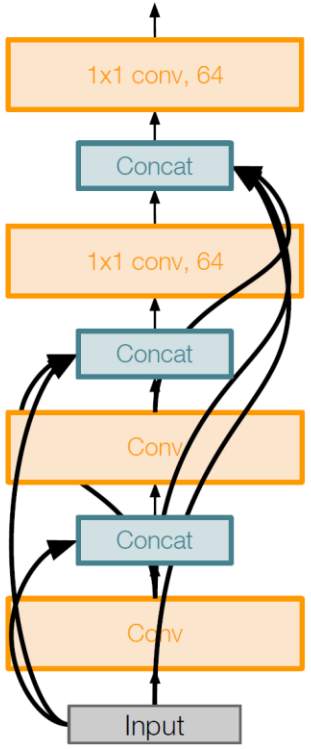
# CIFAR-10 experiments



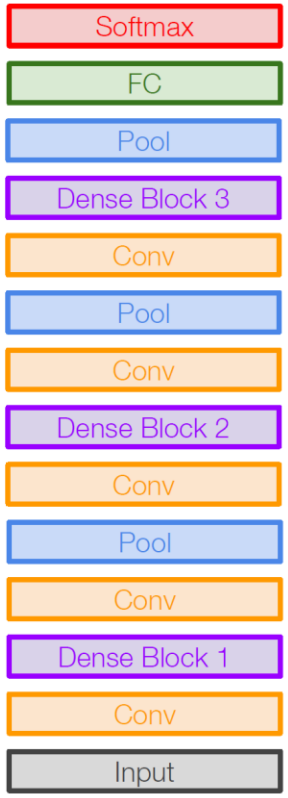



# Case Study: DenseNet and FractalNet

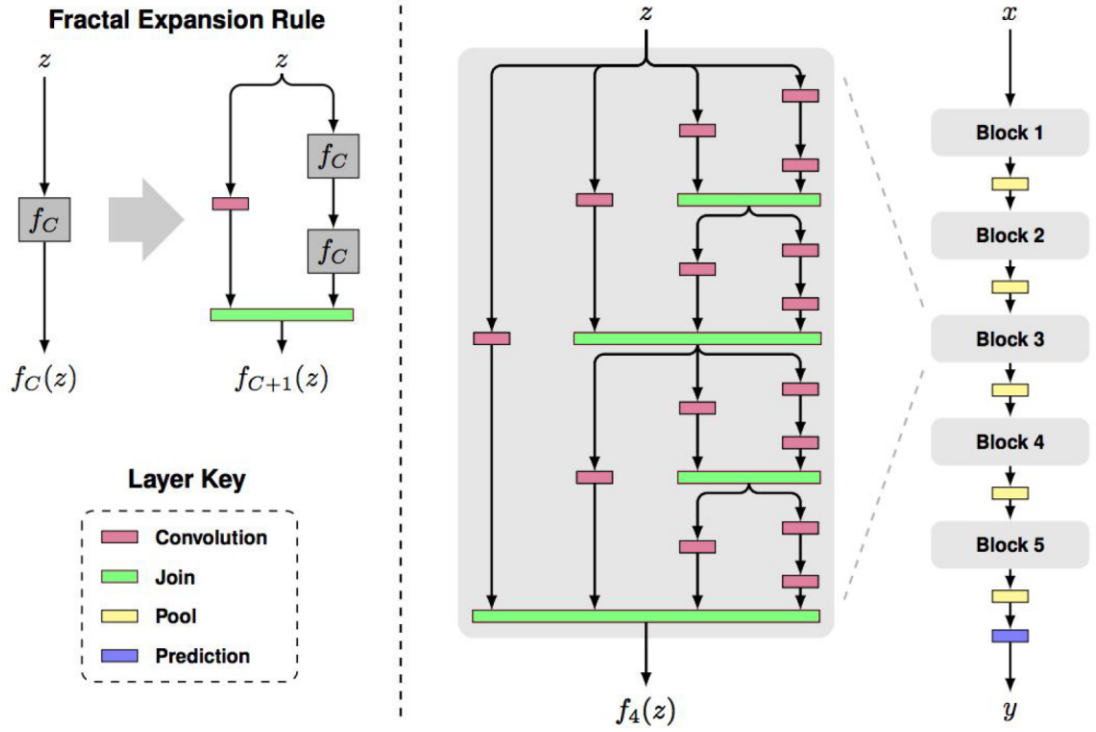
## DenseNet




Dense Block




## FractalNet




[Figures by Larsson et. al, adopted from Li et. al]

# Other CNN architectures

Wide ResNet

ResNetX

DenseNet

…

# And fully convolutional networks (see second half of this lecture)

HourGlass

U-Net

…

# Next week

Modelling of temporal data

Recurrent Neural Networks (RNNs)

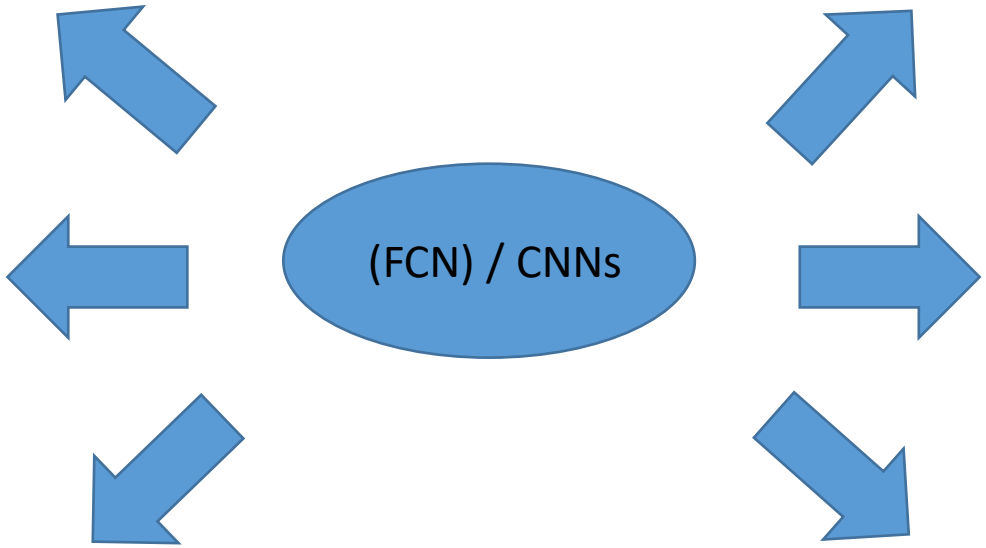
# Outlook [additional material]

# Other interesting directions for future work

**Robustness**
Adversarial attacks

**Robustness**
w.r.t. out of
distribution samples

**Optimization**
Understanding acceleration,
Momentum, etc.




**Other types of networks**
3D Convolutions
Graph Convolutions
BCNs…

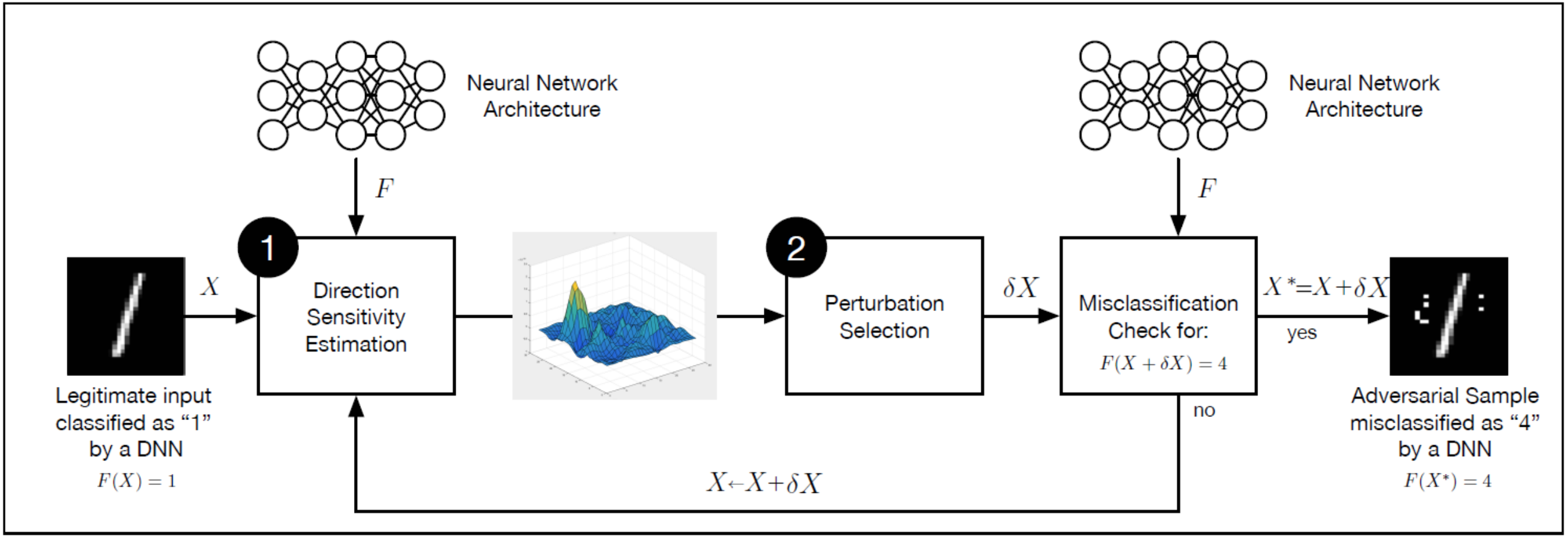
**Learning**
Understanding what and how well
NNs can learn.
Depth vs. Width
Active areas wrt hyperplanes

**Optimization**
Understanding when & why NNs converge
Landscape analysis
Trajectory based analysis
Robustness to overfitting

# Adversarial samples

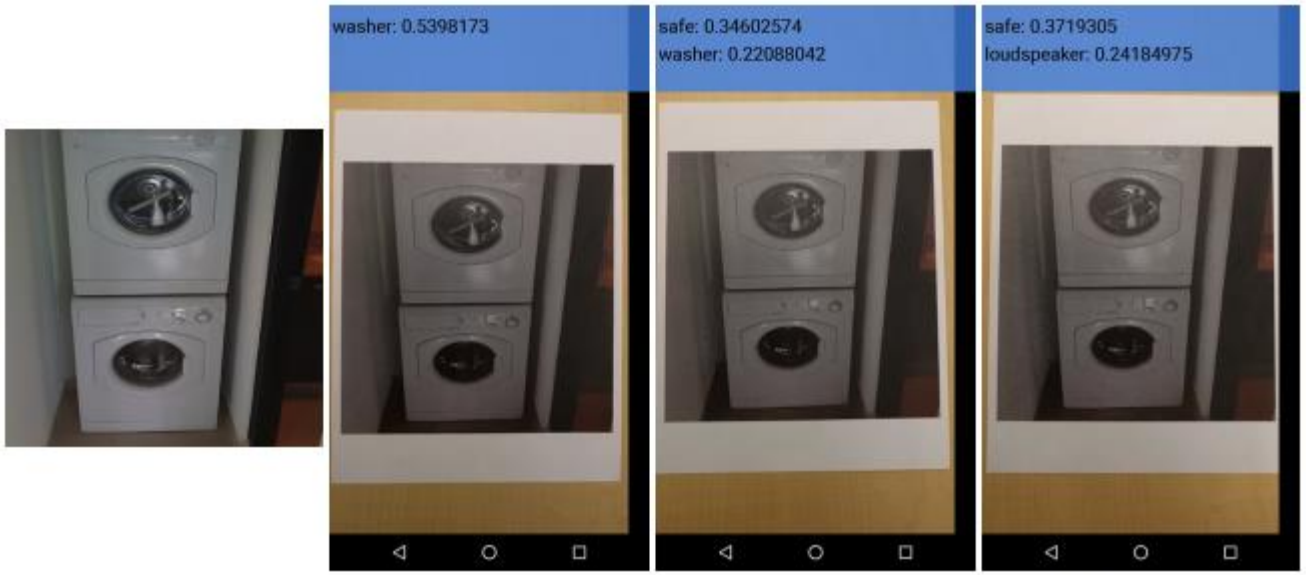

[Goodfellow et al. "Explaining and Harnessing Adversarial Examples"]

# Adversarial crafting framework




[Image source: Papernot et al.]

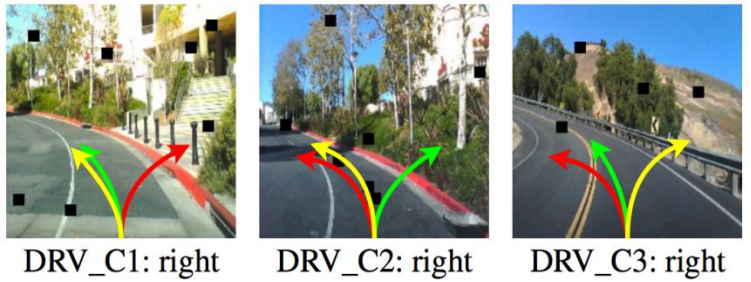
# Robust attacks via adversarial samples




(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

[Kurakin et al. "Adversarial examples in the physical world"]

Self-driving car: in each picture one of the 3 networks makes a mistake…




DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

Tape strips make network predict a 45mph sign


Robust Physical-World Attacks on Deep Learning Visual Classification, CVPR'18

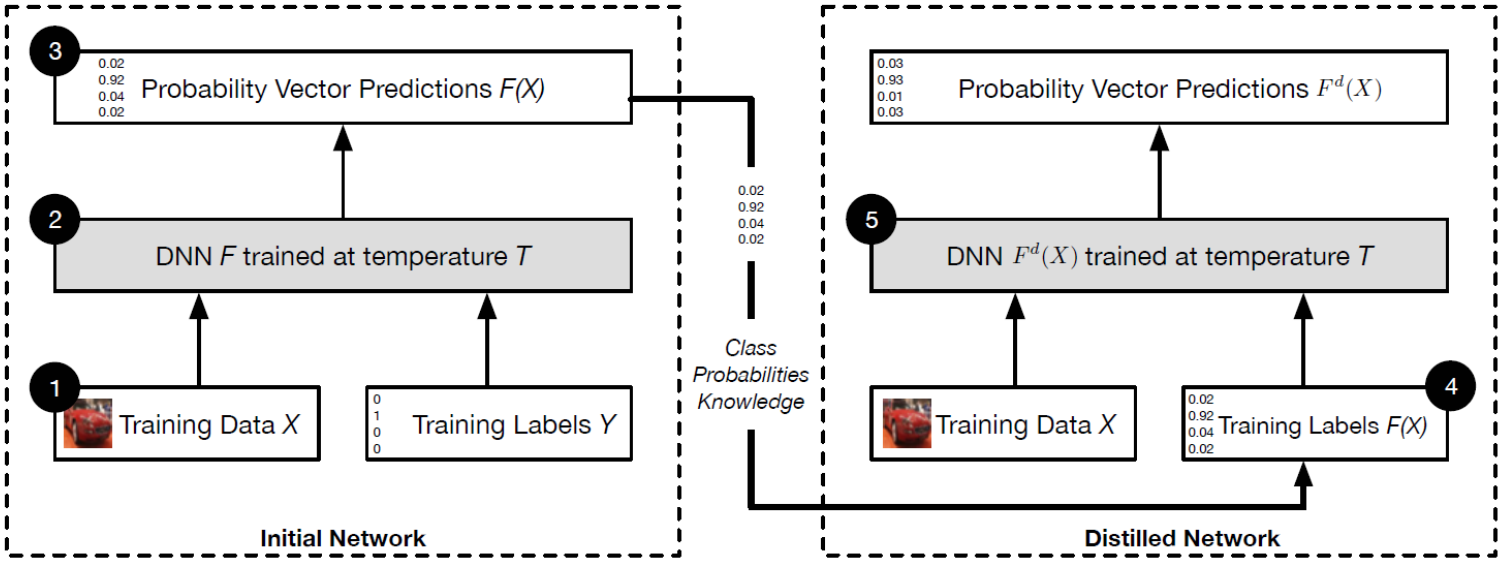
# Potential defenses - Strategies

**Adversarial training**: brute-force train with lots of adversarial samples, to increase robustness. See also:
https://github.com/openai/cleverhans/blob/master/tutorials/mnist_tutorial_tf.md

**Defensive Distillation**: train with "soft"-class probabilities instead of hard labels. Soft-probabilities stem from 2$^{nd}$ model trained on same task with hard labels. Smoothes network with respect to attack.

# Defensive Distillation

$$F(X) = \left[ \frac{e^{\frac{z_i(X)}{T}}}{\sum_{l=0}^{N-1} e^{\frac{z_l(X)}{T}}} \right]$$



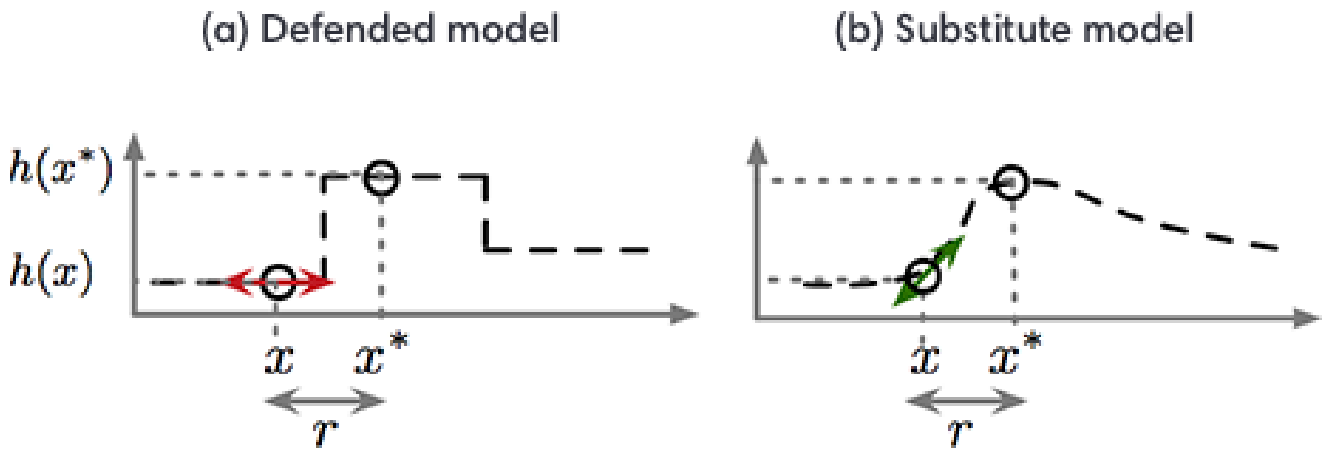
# Potential defenses - Strategies

**Adversarial training**: brute-force train with lots of adversarial samples, to increase robustness. See also:
https://github.com/openai/cleverhans/blob/master/tutorials/mnist_tutorial_tf.md

**Defensive Distillation**: output (and train with) class probabilities instead of hard labels. Probabilities stem from model trained on same task with hard labels. Smoothes network with respect to attack.

Not real solutions, more patches. New ideas and research needed!

# Failed defenses: Gradient Masking



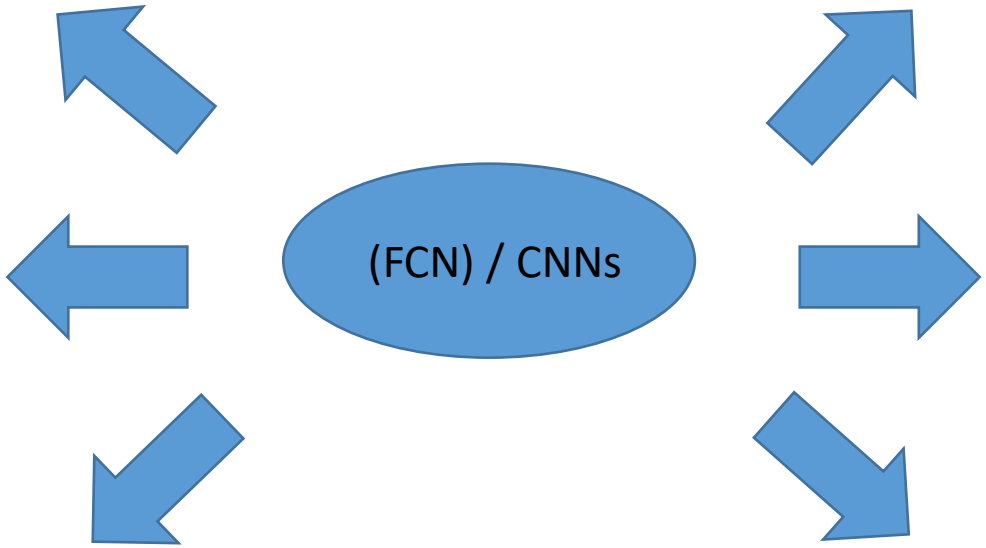

[Papernot et al. “Practical Black-Box Attacks against Machine Learning”]

# Other interesting directions for future work

**Robustness**
Adversarial attacks

**Robustness**
w.r.t. out of
distribution samples

**Optimization**
Understanding acceleration,
Momentum, etc.




**Other types of networks**
3D Convolutions
Graph Convolutions
BCNs…

**Learning**
Understanding what and how well
NNs can learn.
Depth vs. Width
Active areas wrt hyperplanes

**Optimization**
Understanding when & why NNs converge
Landscape analysis
Trajectory based analysis
Robustness to overfitting

# Further reading

**Adversarial attacks & defenses:**

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). "*Intriguing properties of neural networks*". arXiv preprint arXiv:1312.6199.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "*The limitations of deep learning in adversarial settings*," in Proceedings of the 1st IEEE European Symposium on Security and Privacy. IEEE, 2016.

I.J. Goodfellow, J. Shlens, and C. Szegedy, "*Explaining and harnessing adversarial examples*," in Proceedings of the 2015 International Conference on Learning Representations. Computational and Biological Learning Society, 2015.

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016, May). "*Distillation as a defense to adversarial perturbations against deep neural networks*". In 2016 IEEE Symposium on Security and Privacy (SP) (pp. 582-597). IEEE.

**Out of distribution samples:**

Liang, Shiyu, Li, Yixuan, and Srikant, R. "*Principled detection of out-of-distribution examples in neural networks*". In ICLR, 2018

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). "*A simple unified framework for detecting out-of-distribution samples and adversarial attacks*". In NIPS (pp. 7167-7177).

# Further reading

**Optimization:**

Maddison, C. J., Paulin, D., Teh, Y. W., & Doucet, A. (2019). "*Dual Space Preconditioning for Gradient Descent*". arXiv preprint arXiv:1902.02257.

**Understanding NNs & Gradient descent:**

Flammarion, N., & Bach, F. (2015, June). "*From averaging to acceleration, there is only a step-size*". In Conference on Learning Theory (pp. 658-695).

Gunasekar, S., Lee, J. D., Soudry, D., & Srebro, N. (2018). "*Implicit bias of gradient descent on linear convolutional networks*". In Advances in Neural Information Processing Systems (pp. 9482-9491).

**Understanding what and how NNs learn:**

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam, "*Bounding and Counting Linear Regions of Deep Neural Networks*", ICML 2018.

**Landscape analysis:**

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015, February). "*The loss surfaces of multilayer networks*". In Artificial Intelligence and Statistics (pp. 192-204).

Kawaguchi, Kenji. "*Deep learning without poor local minima.*" In Advances in neural information processing systems, pp. 586-594. 2016.

**Trajectory analysis**:

Brutzkus, A., & Globerson, A. (2017, August). "*Globally optimal gradient descent for a convnet with gaussian inputs*". In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 605-614). JMLR.

Bartlett, P., Helmbold, D. & Long, P.. (2018). "*Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks*". Proceedings of the 35th International Conference on Machine Learning, in PMLR 80:521-530.

Arora, S., Cohen, N., Golowich, N., & Hu, W. (2018). "*A convergence analysis of gradient descent for deep linear neural networks*". arXiv preprint arXiv:1810.02281.

**See also**:

http://www.offconvex.org/

# Further reading

**BCNs:**

Gal, Y., & Ghahramani, Z. (2015). "*Bayesian convolutional neural networks with Bernoulli approximate variational inference*". arXiv preprint arXiv:1506.02158.

**3D CNNs**:

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). "*Learning spatiotemporal features with 3d convolutional networks*". In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497).

**Graph convolutions, GCNs**:

Henaff, M., Bruna, J., & LeCun, Y. (2015). "*Deep convolutional networks on graph-structured data*". arXiv preprint arXiv:1506.05163.

Hechtlinger, Y., Chakravarti, P., & Qin, J. (2017). "*A generalization of convolutional neural networks to graph-structured data*". arXiv preprint arXiv:1704.08165.

**Auto-ML:**

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15), Francis Bach and David Blei (Eds.), Vol. 37. JMLR.org 2113-2122.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012.