Pattern Recognition

Deep Learning and CNNs

林彥宇 教授
Yen-Yu Lin, Professor

國立陽明交通大學 資訊工程學系
Computer Science, National Yang Ming Chiao Tung University

Some slides are modified from Yung-Yu Chuang, Sheng-Jyh Wang, Fei-Fei Li, and Justin Johnson

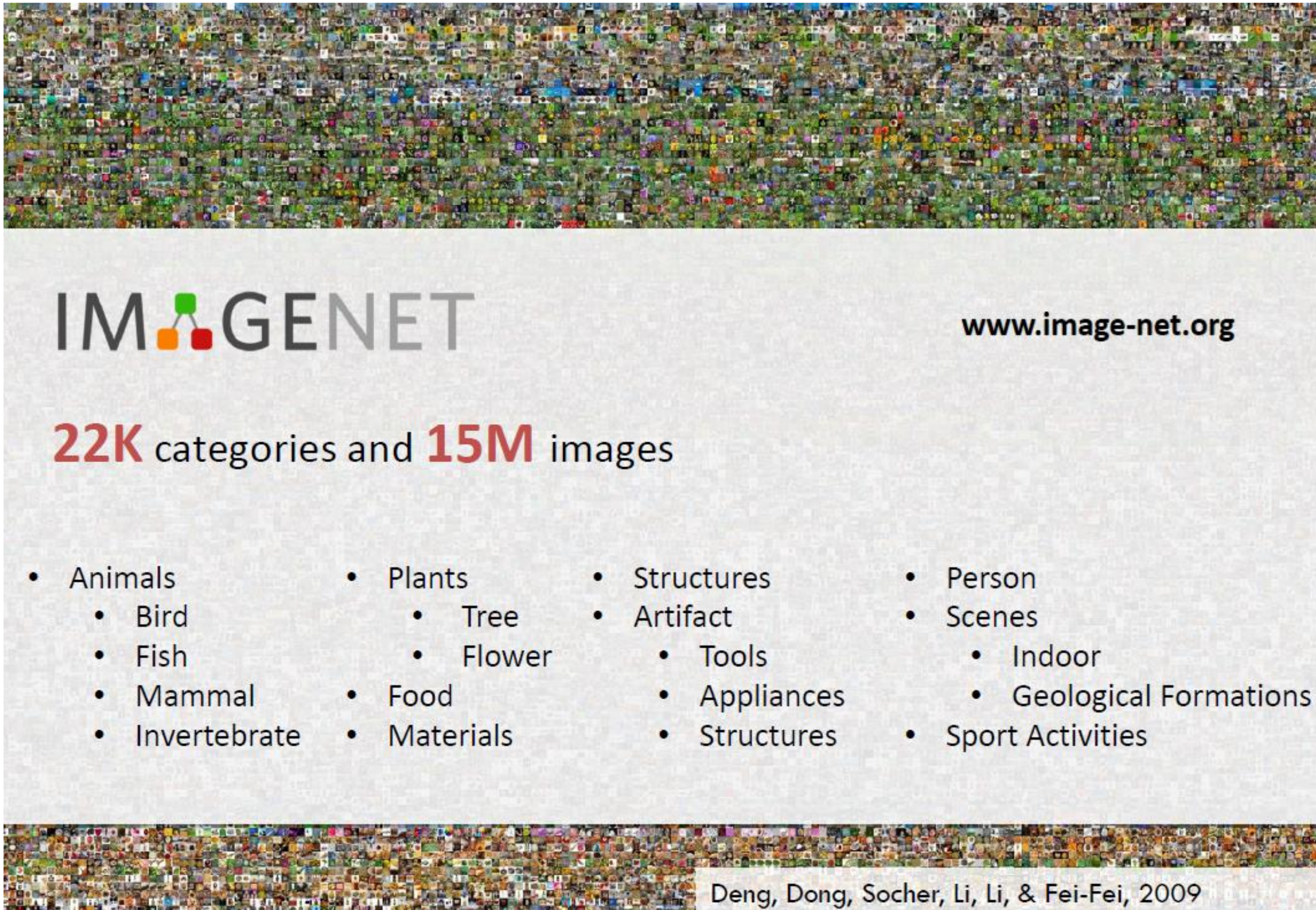
Outline

- Object recognition on ImageNet
- Conventional PR&ML approaches vs. deep learning
- Neural networks and deep learning
- Convolutional neural networks

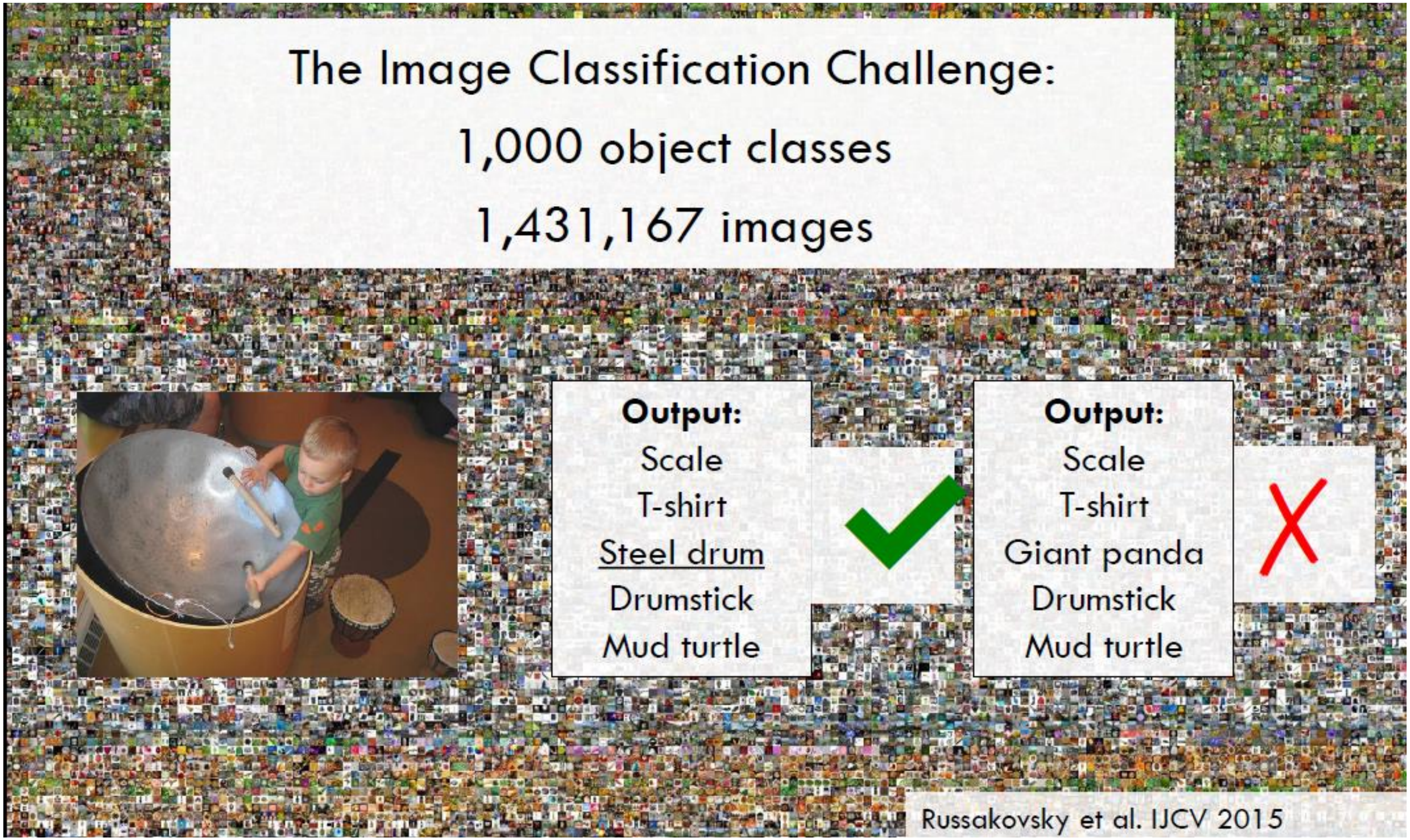
Outline

- Object recognition on ImageNet
- Conventional PR&ML approaches vs. deep learning
- Neural networks and deep learning
- Convolutional neural networks

ImageNet dataset

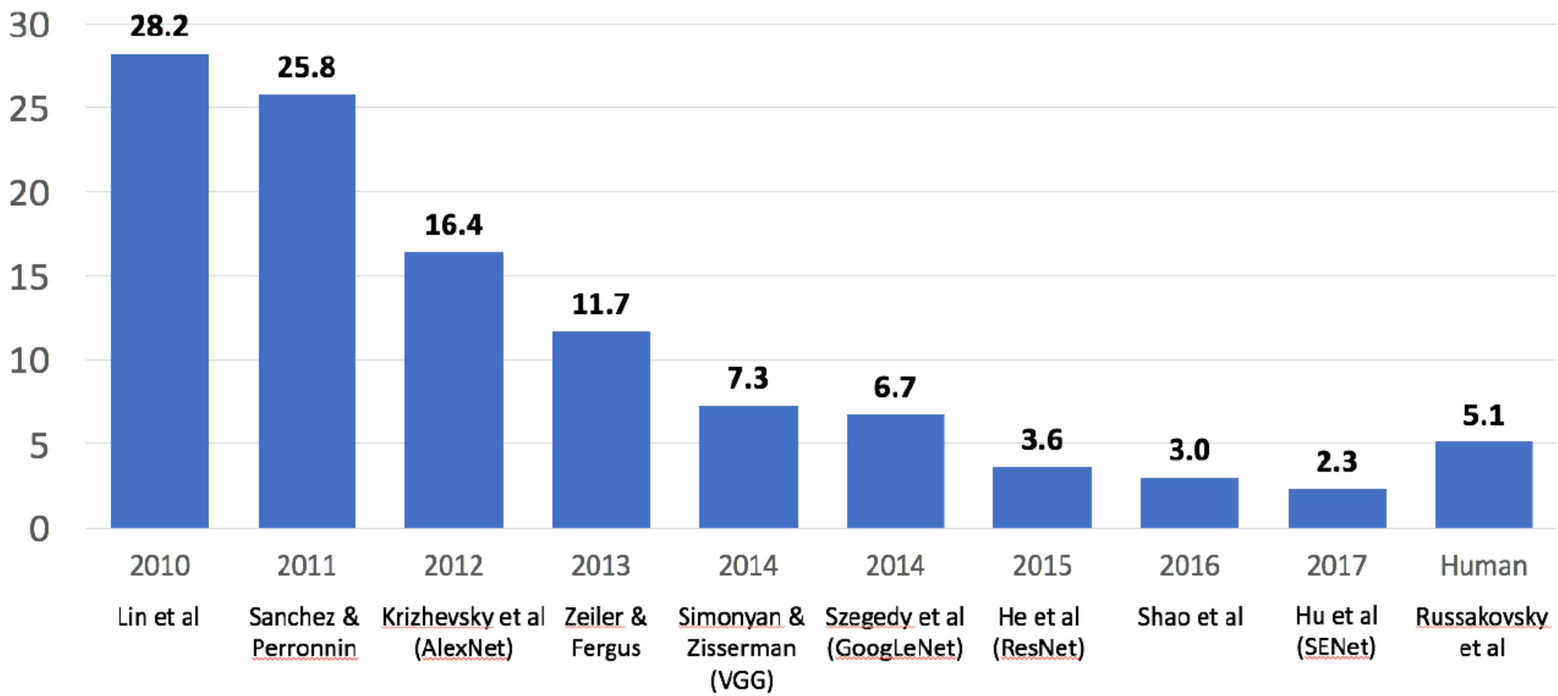


ImageNet large scale visual recognition challenge (ILSVRC)



ILSVRC winning algorithms from 2010 ~ 2017

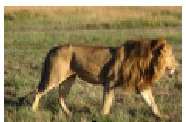
- Since 2012, convolutional neural networks (CNN) has become the most important tool for object recognition



ILSVRC winning algorithms from 2010 ~ 2015

Year 2010

NEC-UIUC



Dense descriptor grid: HOG, LBP

↓

Coding: local coordinate, super-vector

↓

Pooling, SPM

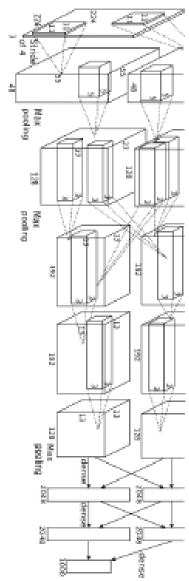
↓

Linear SVM

[Lin CVPR 2011]

Year 2012

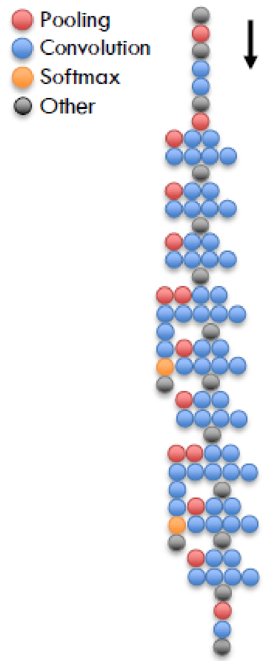
SuperVision



[Krizhevsky NIPS 2012]

Year 2014

GoogLeNet



[Szegedy arxiv 2014]

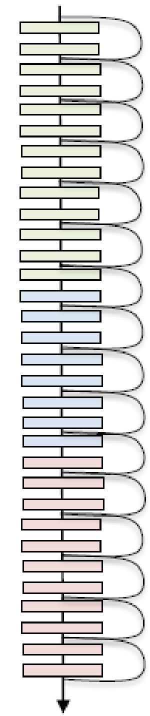
VGG

Image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
fc-4096
fc-4096
fc-1000
softmax

[Simonyan arxiv 2014]

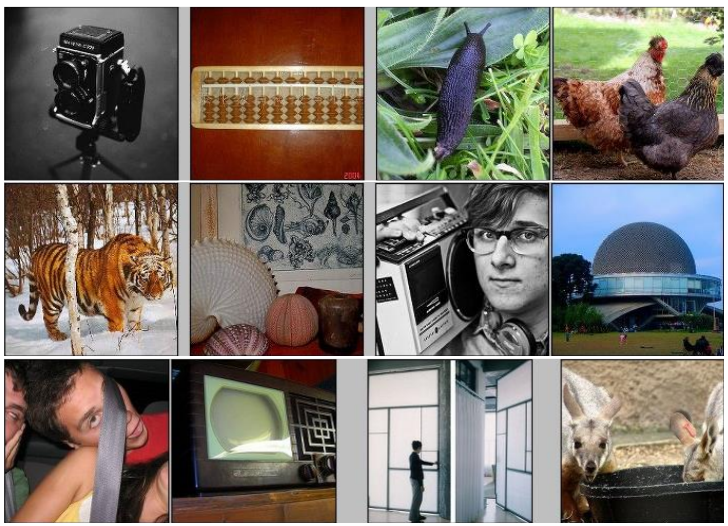
Year 2015

MSRA



[He ICCV 2015]

ImageNet challenge



- About 15M labeled high resolution images
- About 22K categories
- Collected from web and labeled by Amazon Mechanical Turk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  - 1.2 million training images of 1000 classes
  - 50K validation images
  - 150K testing images

ILSVRC winners

- The annual “Olympics” of computer vision.

- Teams from the world compete to determine who has the best computer vision models for classification, detection, segmentation, and more.

- The winner in 2012, AlexNet, achieves a top-5 error rate of 15.3%

- The next best team achieves an error of 26.2%

Outline

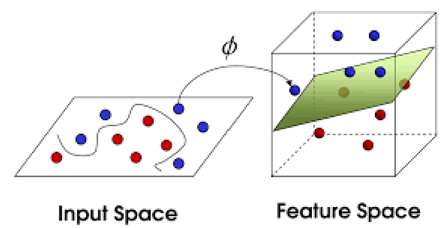
- Object recognition on ImageNet
- Conventional PR&ML approaches vs. deep learning
- Neural networks and deep learning
- Convolutional neural networks

Conventional approach to object recognition

- Training phase

image collection → feature extraction → classifier training → trained classifier

histogram of oriented gradients (HoG)



support vector machines (SVMs)

dogs? ✖

flowers? ○

trains? ✖

persons? ✖

Conventional approach to vision applications

key step

- Training phase

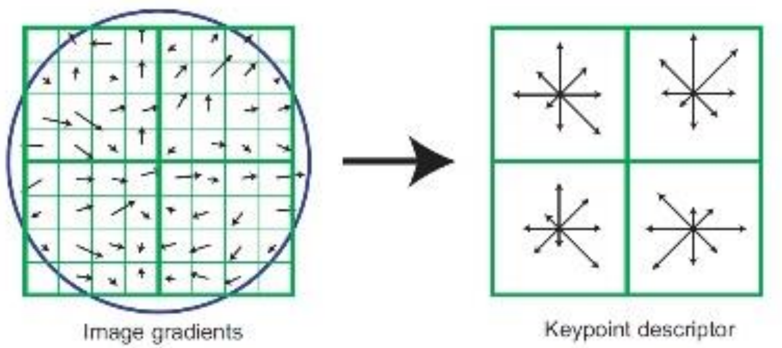
image collection → feature extraction → classifier training → trained classifier

- Testing phase

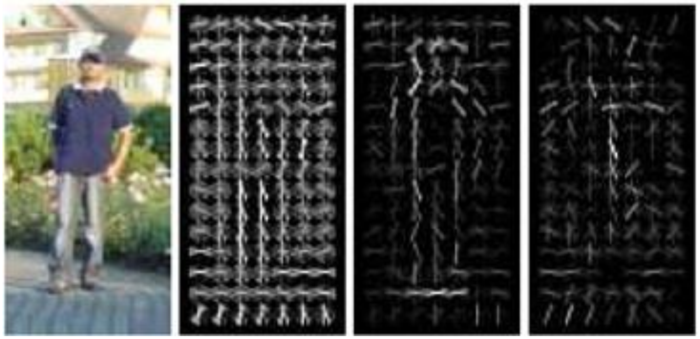
test image → feature extraction → trained classifier → prediction

Features are the keys

- Off-the-shelf visual features

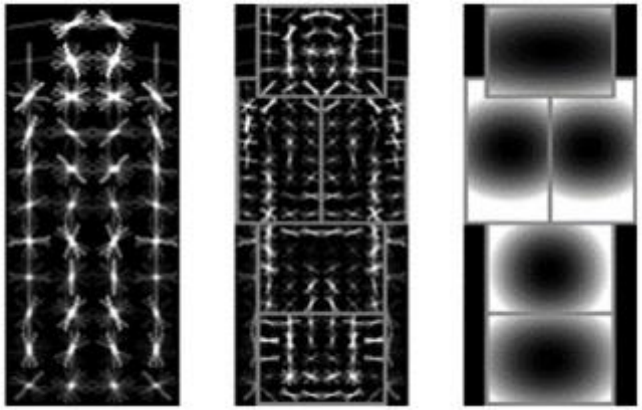


SIFT [Lowe, IJCV'04]
Citations: 43465



HoG [Dalal & Triggs, CVPR'05]
Citations: 20174

Constellation model [Fergus et al., CVPR'03]
Citations: 2551

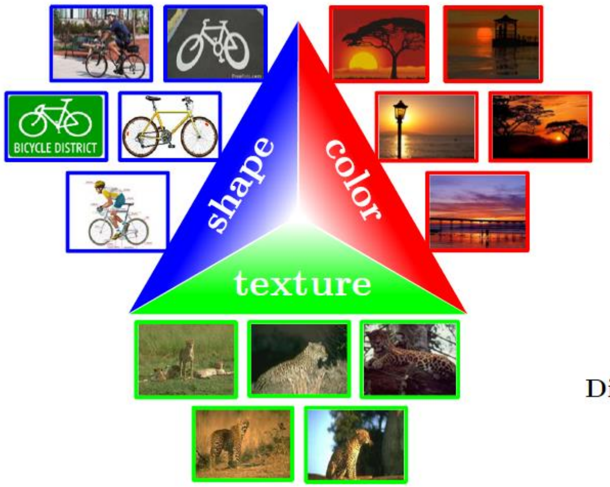


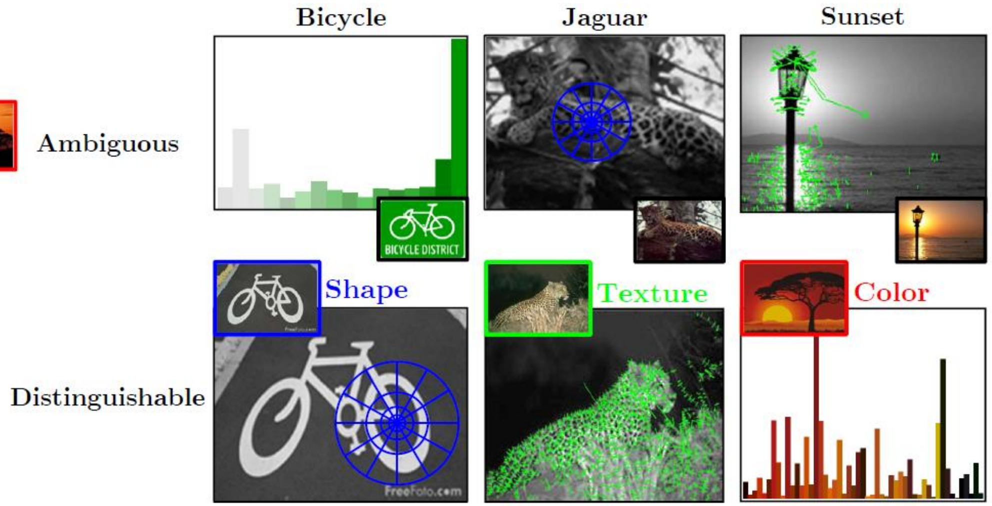
DPM [Felzenszwalb et al., PAMI'10]
Citations: 5093

国立交通大學
National Chiao Tung University

Features are the keys

- Features are the keys to recent progress in classification
- Are handcrafted features optimal?
- The optimal features for classification in general vary from task to task, even from category to category





Conventional approaches vs. Deep learning

- Conventional approaches
  - Fixed/engineered features + trainable classifier

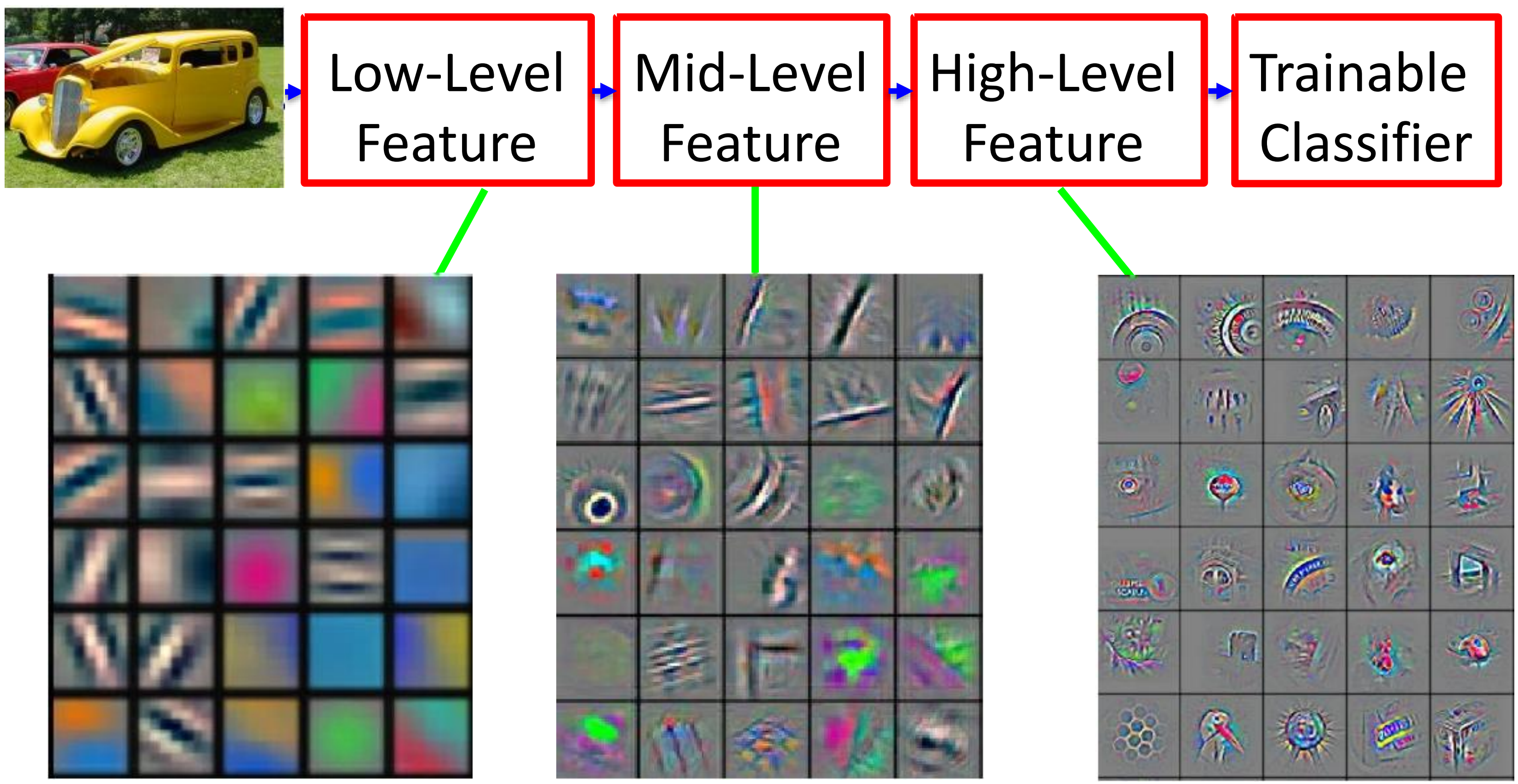
Hand-crafted Feature Extractor → "Simple" Trainable Classifier →

- Deep learning / End-to-end learning / Feature learning
  - Trainable features + trainable classifier

Trainable Feature Extractor → **Trainable** Classifier →

slide: Y LeCun & MA Ranzato

Deep learning = Learning hierarchical representations



slide: Y LeCun & MA Ranzato

國立交通大學
National Chiao Tung University

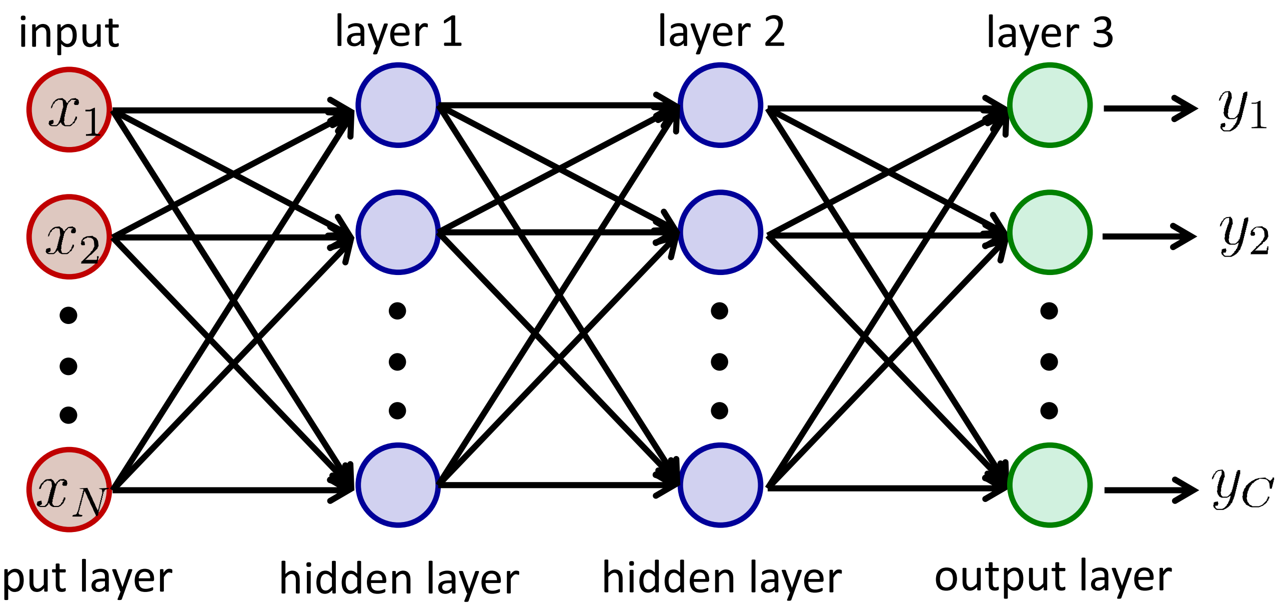
Outline

- Object recognition on ImageNet
- Conventional PR&ML approaches vs. deep learning
- Neural networks and deep learning
- Convolutional neural networks

Neural networks and neurons

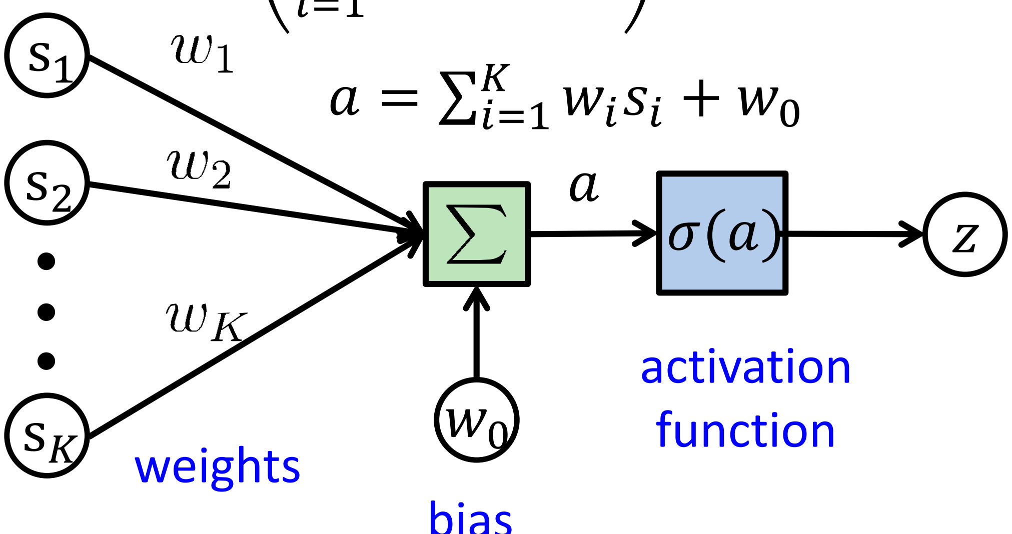
- Neural networks are presented as layers of interconnected neurons
  - Each layer of neurons takes messages from output of previous layer

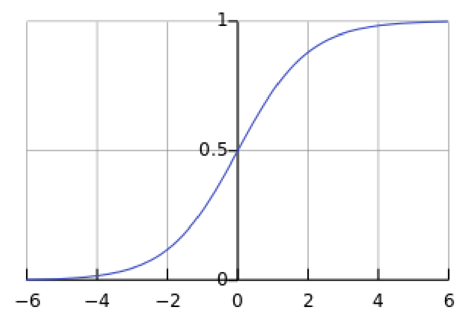
image of N pixels



A neuron

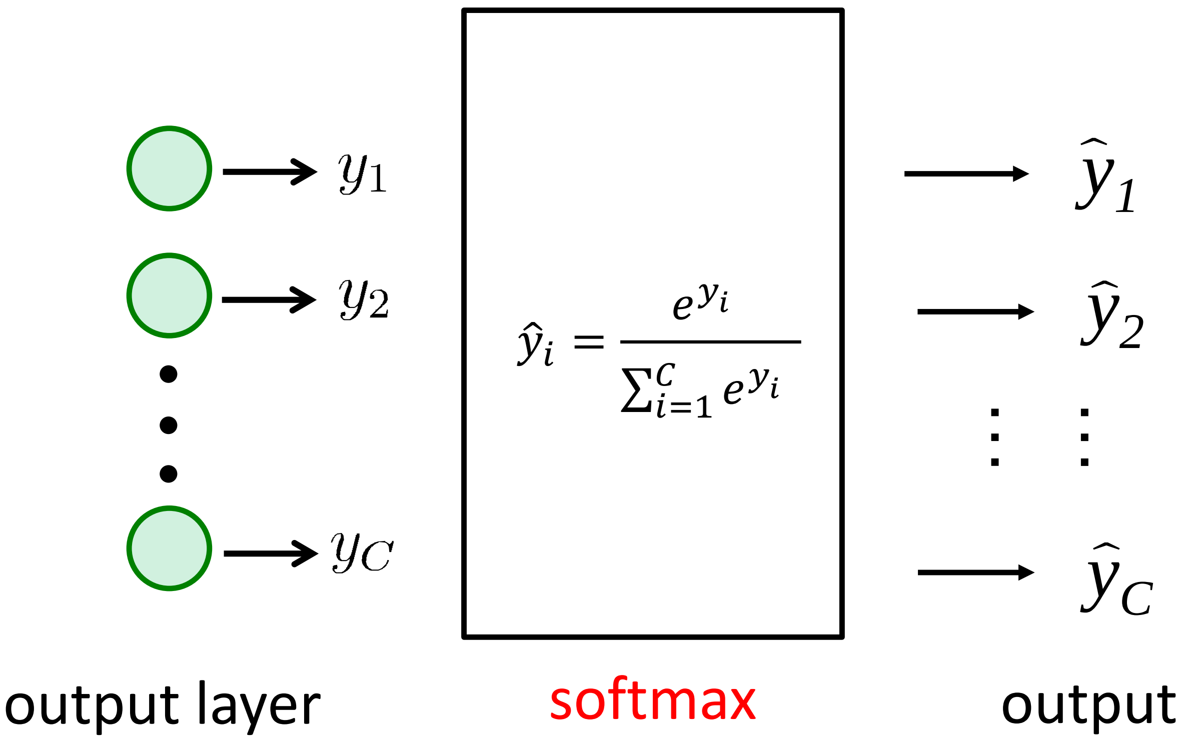
- A function $f : R^K \mapsto R$
  - Map K inputs to 1 output
  - Compute the biased weighted sum
  - Apply a non-linear mapping function (activation function)
  - $f(\mathbf{s}) = \sigma\left(\sum_{i=1}^{K} w_i s_i + w_0\right)$, where $\sigma(a) = \dfrac{1}{1+\exp(-a)}$





sigmoid function

Softmax for multi-class classification



- A probability distributions
- Larger output leads to higher probability
- All positive
- Sum to 1

國立交通大學
National Chiao Tung University

Training neural networks

- Collect a set of labeled training data $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$
- Training neural networks: Finding network parameters $\theta = \{\mathbf{w}, \mathbf{b}\}$ to minimize the loss between true training label $\mathbf{y}_i$ and the estimated label, e.g.,

$$L(\theta) = \sum_{i=1}^{N} \|\mathbf{y}_i - g_{\mathbf{w}}(\mathbf{x}_i)\|^2$$

- Minimization can be done by gradient descent if $L(\cdot)$ is differentiable with respect to θ
- Backpropagation: a widely used method for optimizing multi-layer neural networks

Gradient descent

- The simplest approach is to update optimization variable set $\mathbf{w}$ by a displacement in the negative gradient direction

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

  - This is a steepest descent algorithm
  - $\eta > 0$ is the learning rate
  - This is a batch method, as evaluation of ∇E involves the entire data set
  - A range of starting points $\{\mathbf{w}^{(0)}\}$ may be needed, in order to find a satisfactory minimum

Stochastic gradient descent

- Stochastic gradient descent (or called sequential gradient descent) has proved useful in practice when training neural networks on a large data set
- The error function needs to comprise a sum of terms, one for each data point, i.e.,

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

  - Sum-of-squares error for regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2$$

  - Cross-entropy error for classification

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\}$$

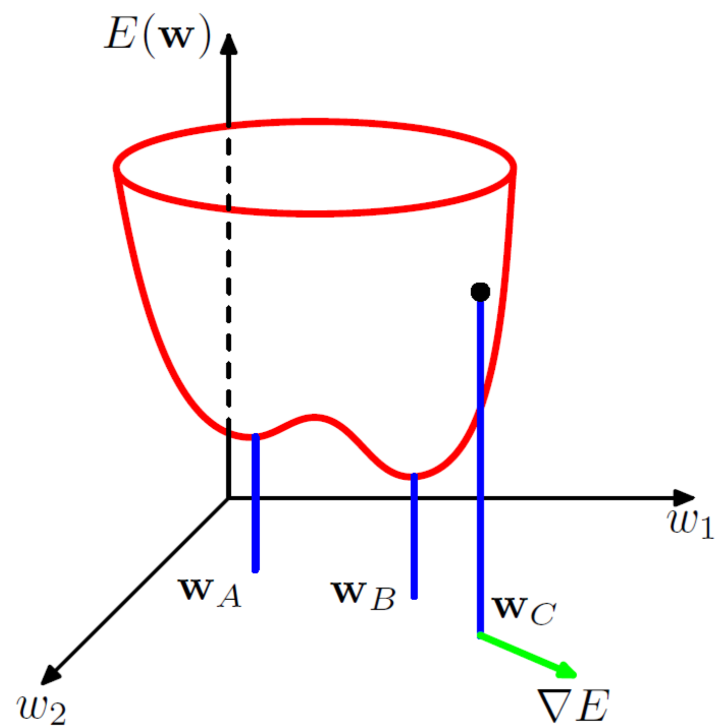
Geometric view of gradient descent

- The error function $E(\mathbf{w})$ is a surface sitting over the weight space

- $\mathbf{w}_A$ is a local minimum

- $\mathbf{w}_B$ is a global minimum

- At any point $\mathbf{w}_C$, the local gradient of the error surface is given by the vector ∇E

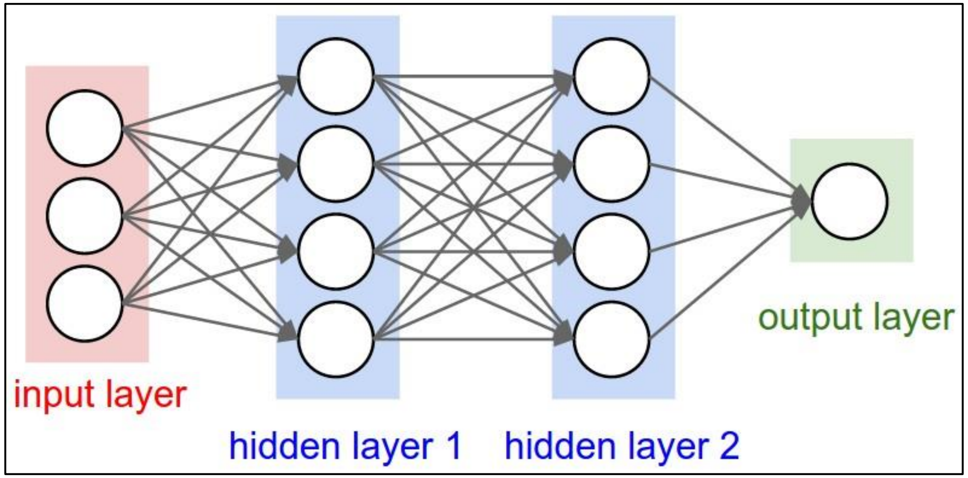


國立交通大學
National Chiao Tung University

Optimization

- Mini-batch SGD (stochastic gradient descent)
- Loop:

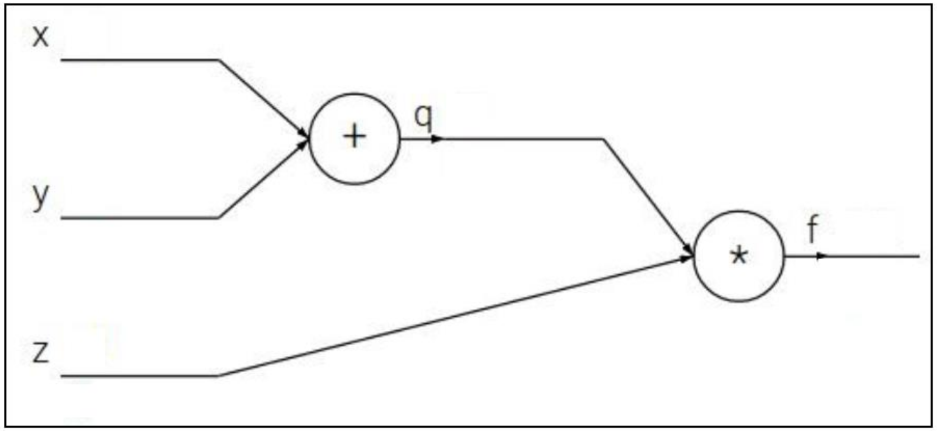
1. **Sample** a batch of data
2. **Forward** prop it through the network and get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient



Backpropagation and chain rule

- Consider a simple example

$$f(x, y, z) = (x + y)z$$



National Chiao Tung University

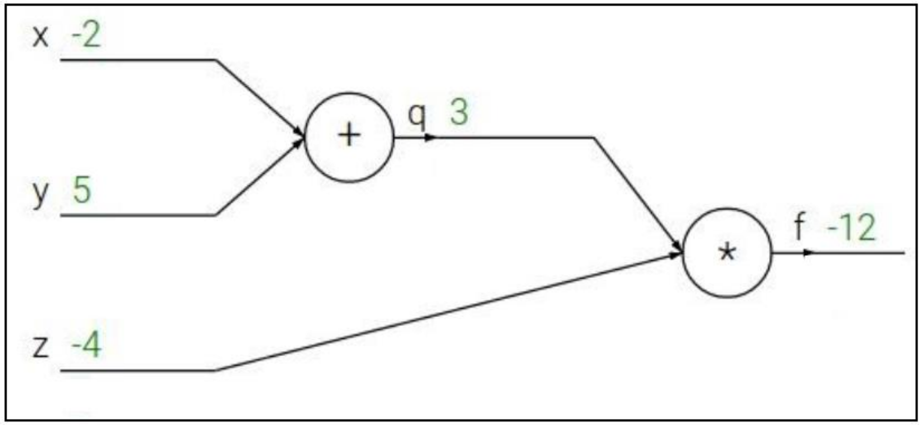
Backpropagation and chain rule

- Consider a simple example

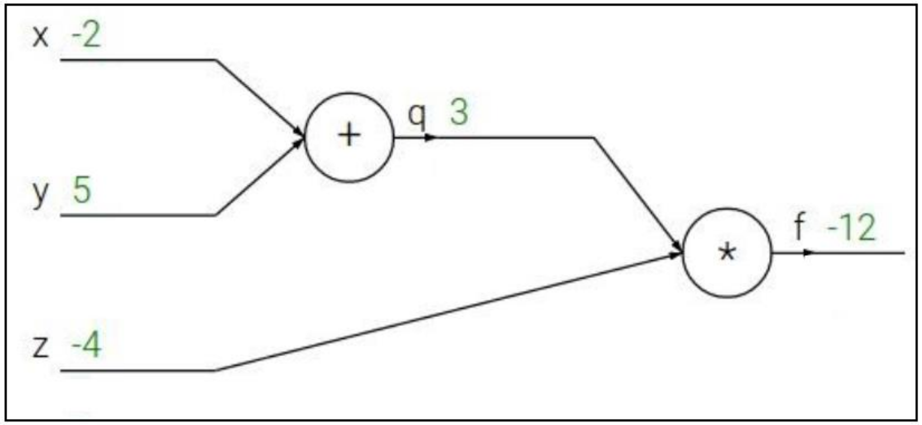
  $f(x, y, z) = (x + y)z$

- Current values

  $x = -2, y = 5, z = -4$



Backpropagation and chain rule



- Consider a simple example
  $f(x, y, z) = (x + y)z$
- Current values
  $x = -2, y = 5, z = -4$
- Find the gradient of f with respect to each variable: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

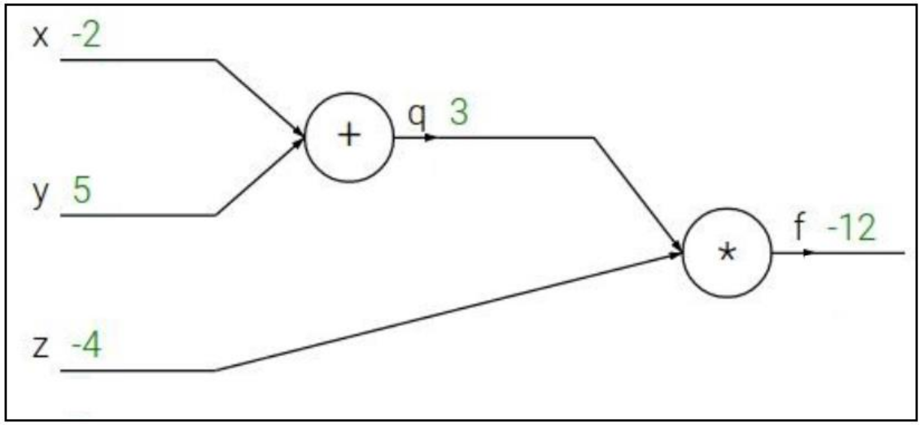
Backpropagation and chain rule

- Consider a simple example

  $f(x, y, z) = (x + y)z$

- Current values

  $x = -2, y = 5, z = -4$



- Find the gradient of f with respect to each variable: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

$$q = x + y \Rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

國立交通大學
National Chiao Tung University

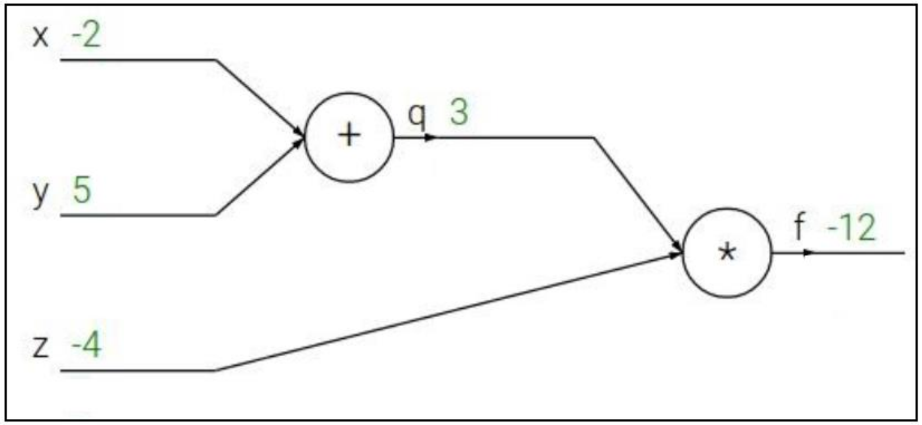
Backpropagation and chain rule

- Consider a simple example

  $f(x, y, z) = (x + y)z$

- Current values

  $x = -2, y = 5, z = -4$



- Find the gradient of f with respect to each variable: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$

$$q = x + y \Rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \Rightarrow \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

國立交通大學
National Chiao Tung University

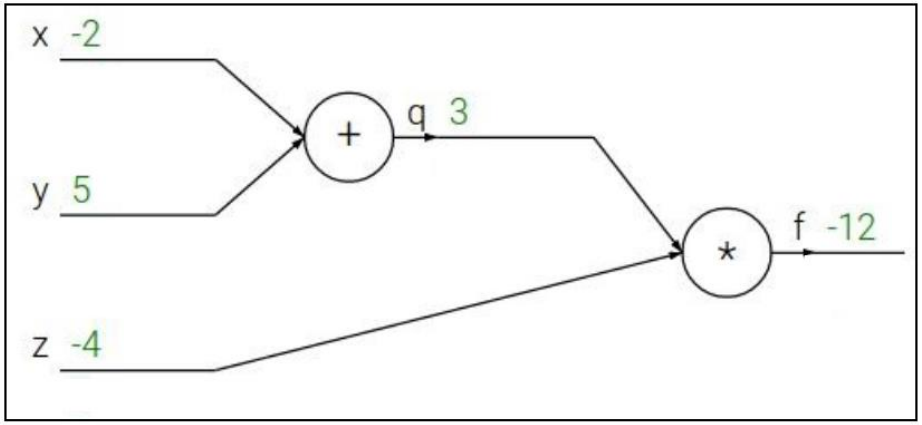
Backpropagation and chain rule

- Consider a simple example

  $f(x, y, z) = (x + y)z$

- Current values

  $x = -2, y = 5, z = -4$



- Find the gradient of f with respect to each variable: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$q = x + y \Rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \Rightarrow \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\boxed{\frac{\partial f}{\partial z} = q = 3}$$

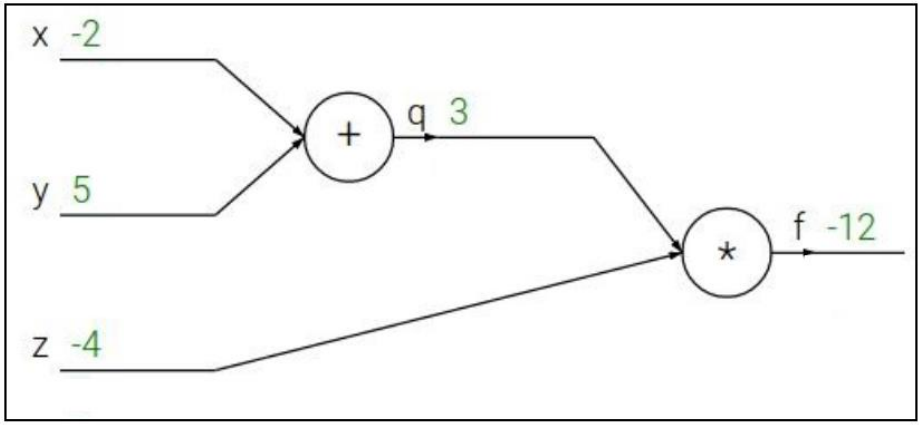
Backpropagation and chain rule

- Consider a simple example

  $$f(x, y, z) = (x + y)z$$

- Current values

  $$x = -2, y = 5, z = -4$$



- Find the gradient of f with respect to each variable: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$q = x + y \Rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \Rightarrow \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = z \times 1 = -4}$$

國立交通大學
National Chiao Tung University

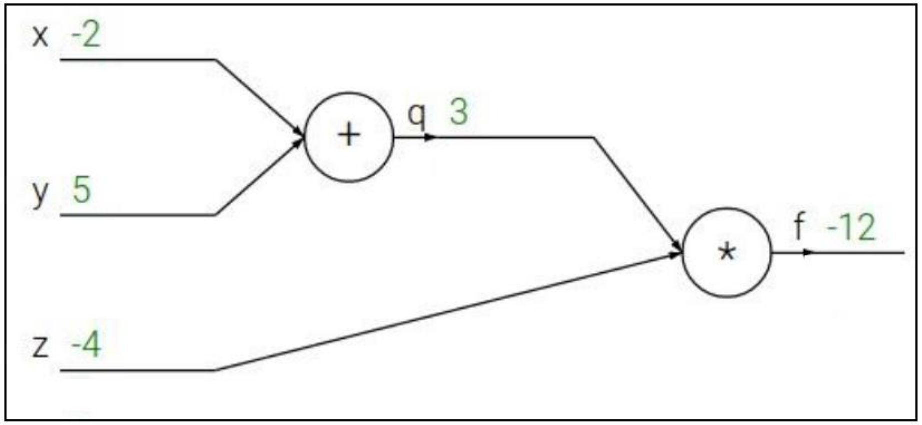
Backpropagation and chain rule

- Consider a simple example

  $$f(x,y,z)=(x+y)z$$

- Current values

  $$x=-2, y=5, z=-4$$



- Find the gradient of f with respect to each variable: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$q = x + y \Rightarrow \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \Rightarrow \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

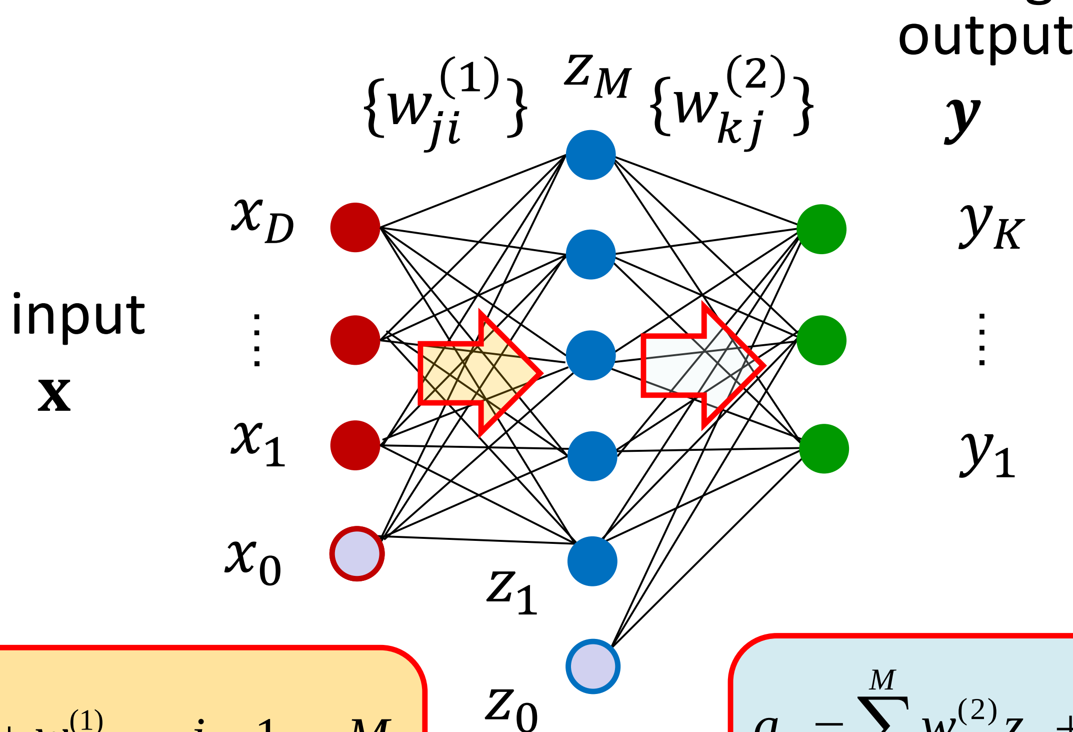
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = z \times 1 = -4}$$

Error backpropagation

- The computational cost of gradient descent mainly lies in the evaluation of gradient at each iteration
- In feed-forward neural networks, the gradient of an error function $E(\mathbf{w})$ can be efficiently evaluated via an algorithm called error backpropagation

Feed-forward neural networks

- Two-layer feed-forward neural networks for regression



$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \qquad j = 1, \ldots, M$$

$$z_j = h(a_j)$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \qquad k = 1, \ldots, K$$

$$y_k = a_k$$

Error backpropagation

- Variables/Activations dependency:

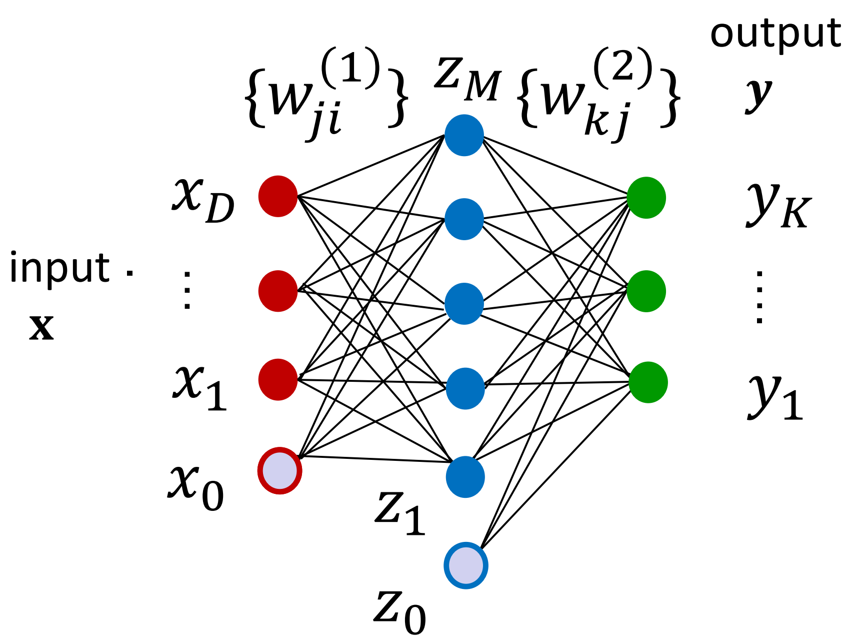
$$\{x_i\} \rightarrow \{w_{ji}^{(1)}\} \rightarrow \{a_j\} \rightarrow \{z_j\} \rightarrow \{w_{kj}^{(2)}\} \rightarrow \{a_k\} \rightarrow \{y_k\} \rightarrow E$$

- Our goal in gradient computation:

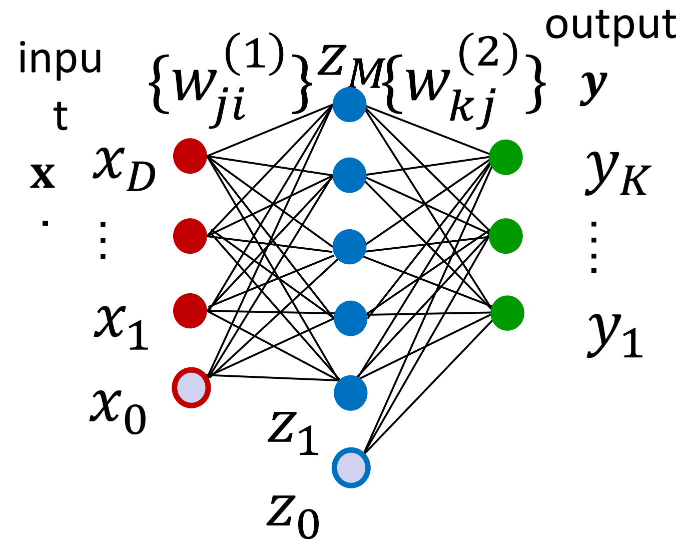
$$\frac{\partial E}{\partial w_{kj}^{(2)}} \text{ and } \frac{\partial E}{\partial w_{ji}^{(1)}}$$

- In backpropagation, we also need to compute

$$\delta_k = \frac{\partial E}{\partial a_k} \text{ and } \delta_j = \frac{\partial E}{\partial a_j}$$



Error backpropagation



- Stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

- Multi-dimensional regression

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \qquad j = 1, \ldots, M$$

$$z_j = h(a_j)$$

Hidden layer

$$\delta_j \equiv \frac{\partial E}{\partial a_j} = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

$$= h'(a_j) \sum_k w_{kj}^{(2)} \delta_k$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \qquad k = 1, \ldots, K$$

$$y_k = a_k$$

Output layer

$$\delta_k \equiv \frac{\partial E}{\partial a_k} = y_k - t_k$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2$$

Error function

國立交通大學
National Chiao Tung University

Error backpropagation

- Variables/Activations dependency:

$$\{x_i\} \rightarrow \{w_{ji}^{(1)}\} \rightarrow \{a_j\} \rightarrow \{z_j\} \rightarrow \{w_{kj}^{(2)}\} \rightarrow \{a_k\} \rightarrow \{y_k\} \rightarrow E$$

$$\delta_j = h'(a_j)\sum_k w_{kj}^{(2)}\delta_k$$

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \qquad j = 1,\dots,M$$

$$z_j = h(a_j)$$

Hidden layer

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \qquad k = 1,\dots,K$$

$$y_k = a_k$$

Output layer

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2$$

Error function

$$\delta_k = y_k - t_k$$

國立交通大學 National Chiao Tung University

Error backpropagation

- Variables/Activations dependency:

$$\{x_i\} \rightarrow \{w_{ji}^{(1)}\} \rightarrow \{a_j\} \rightarrow \{z_j\} \rightarrow \{w_{kj}^{(2)}\} \rightarrow \{a_k\} \rightarrow \{y_k\} \rightarrow E$$

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \qquad j = 1, \ldots, M$$

$$z_j = h(a_j)$$

Hidden layer

$$\delta_j = h'(a_j) \sum_k w_{kj}^{(2)} \delta_k$$

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \qquad k = 1, \ldots, K$$

$$y_k = a_k$$

Output layer

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

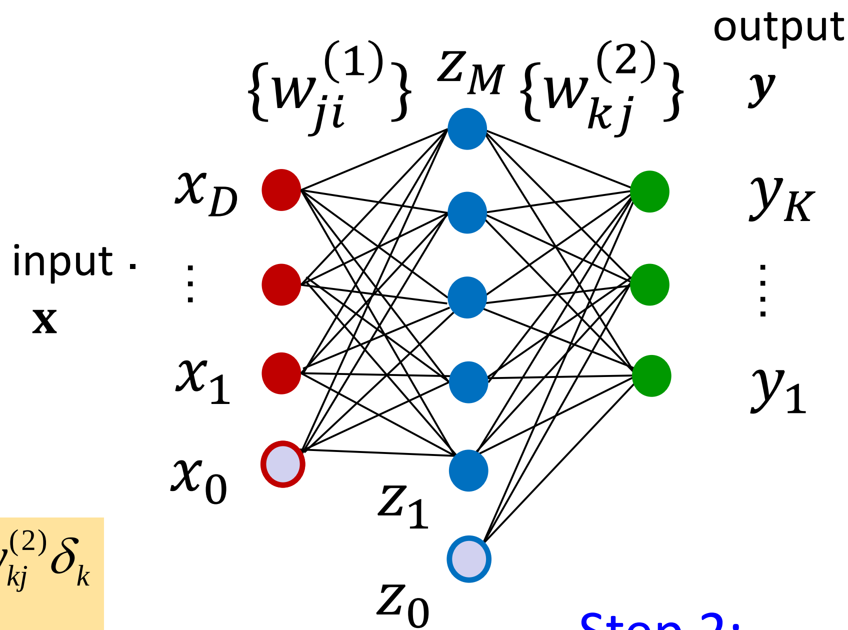
$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2$$

Error function

$$\delta_k = y_k - t_k$$

國立交通大學
National Chiao Tung University

A review of error backpropagation



Step 1:

$$\delta_k = y_k - t_k$$

Step 2:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

Step 3:

$$\delta_j = h'(a_j)\sum_k w_{kj}^{(2)}\delta_k$$

Step 4:

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

Error backpropagation for other tasks

- Step 1: $\delta_k \equiv \dfrac{\partial E}{\partial a_k} = \dfrac{\partial E}{\partial y_k}\dfrac{\partial y_k}{\partial a_k}$

$$E(\mathbf{w}) = \begin{cases} \dfrac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2 & \text{regression} \\ -\{t \ln y(\mathbf{x},\mathbf{w}) + (1-t)\ln(1-y(\mathbf{x},\mathbf{w}))\} & \text{binary classification} \\ -\sum_{k=1}^{K} t_k \ln y_k(\mathbf{x}, \mathbf{w}) & \text{multi-calss classification} \end{cases}$$

$$y_k = a_k \quad \text{regression}$$

$$y = \frac{1}{1+e^{-a}} \text{binary classification}$$

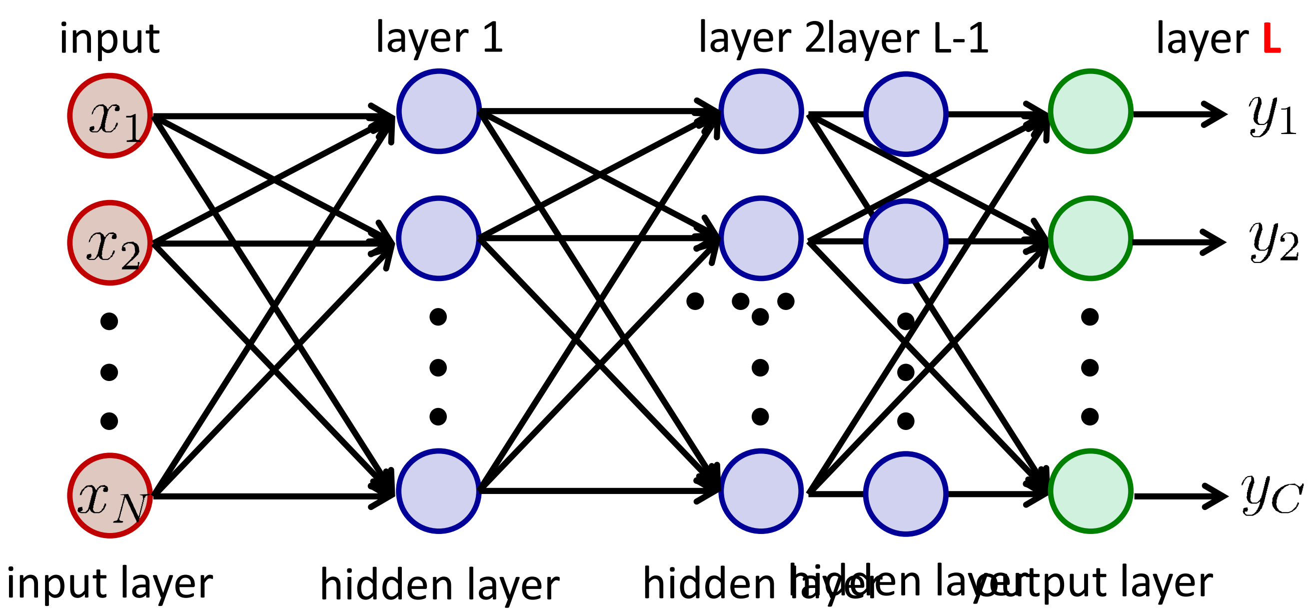
$$y_k = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{multi}-\text{class classification}$$

- Steps 2 ~ 4 remain unchanged

國立交通大學 National Chiao Tung University

What is deep neural networks (DNN)

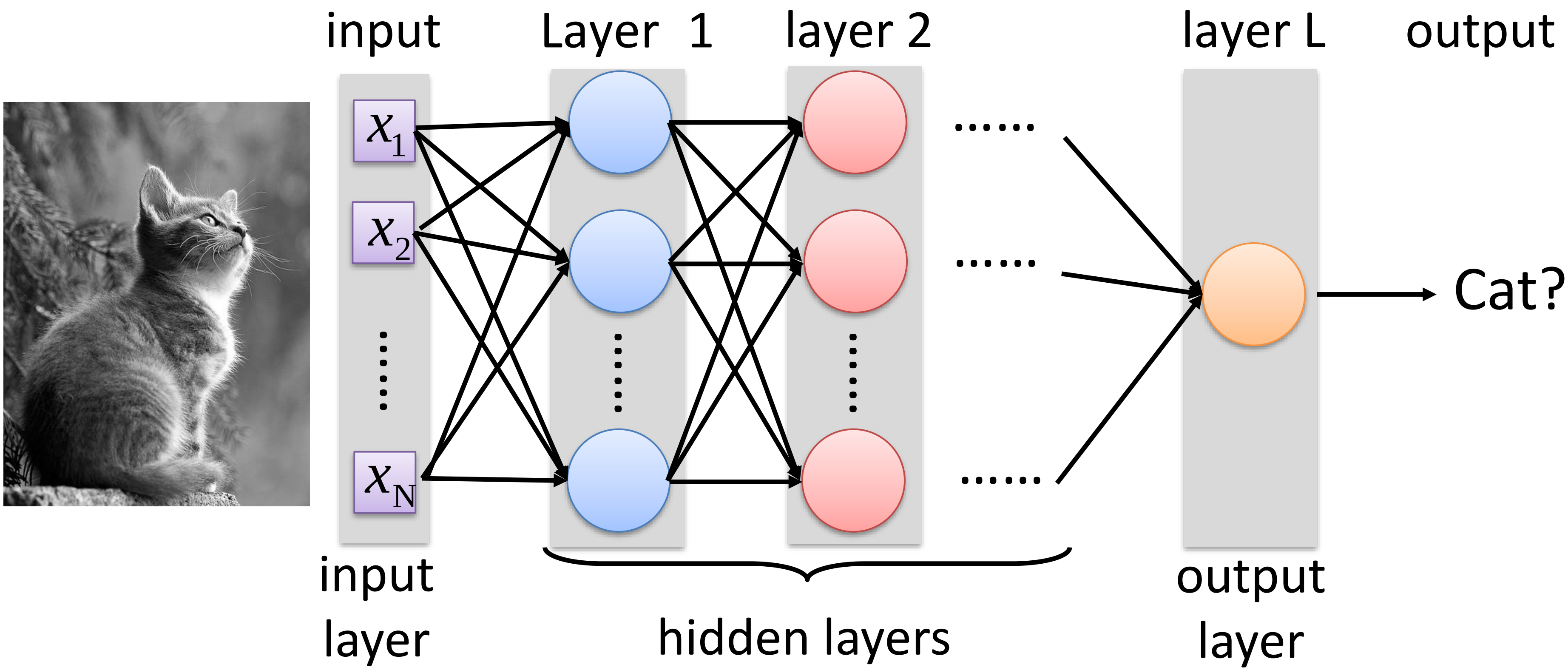
- DNN is neural networks with many hidden layers



Outline

- Object recognition on ImageNet
- Conventional PR&ML approaches vs. deep learning
- Neural networks and deep learning
- Convolutional neural networks

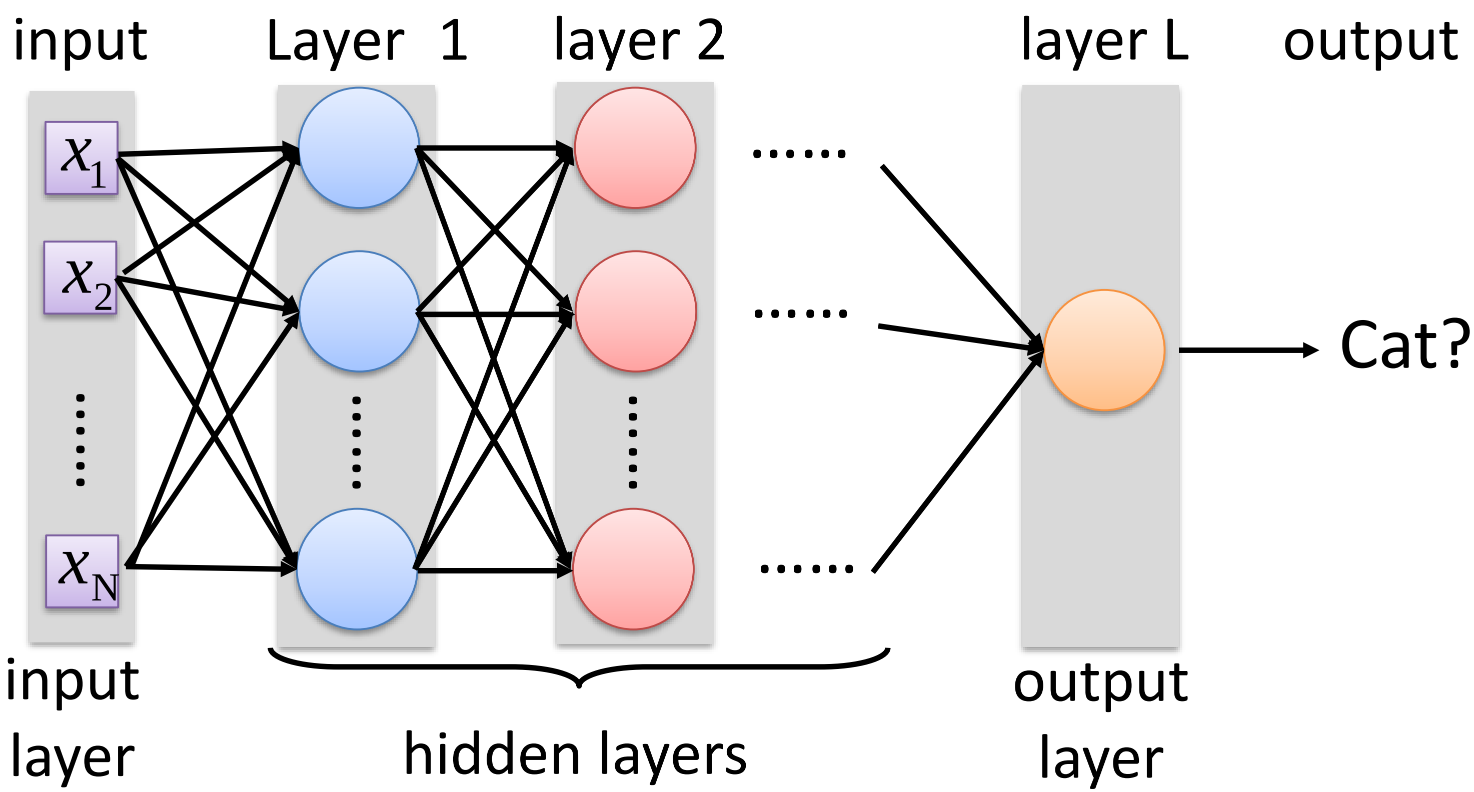
Deep learning for object recognition



National Chiao Tung University

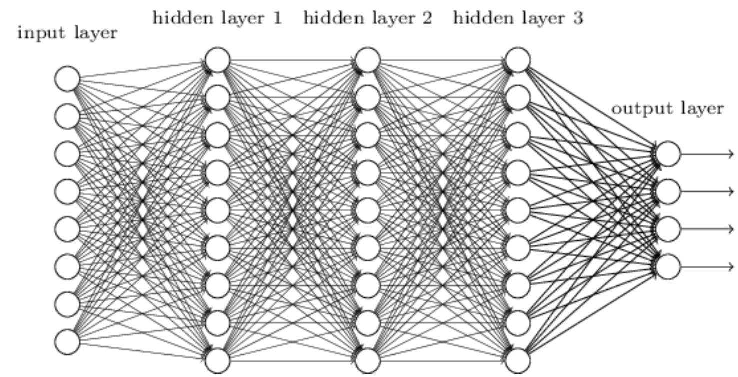
Deep learning for object recognition

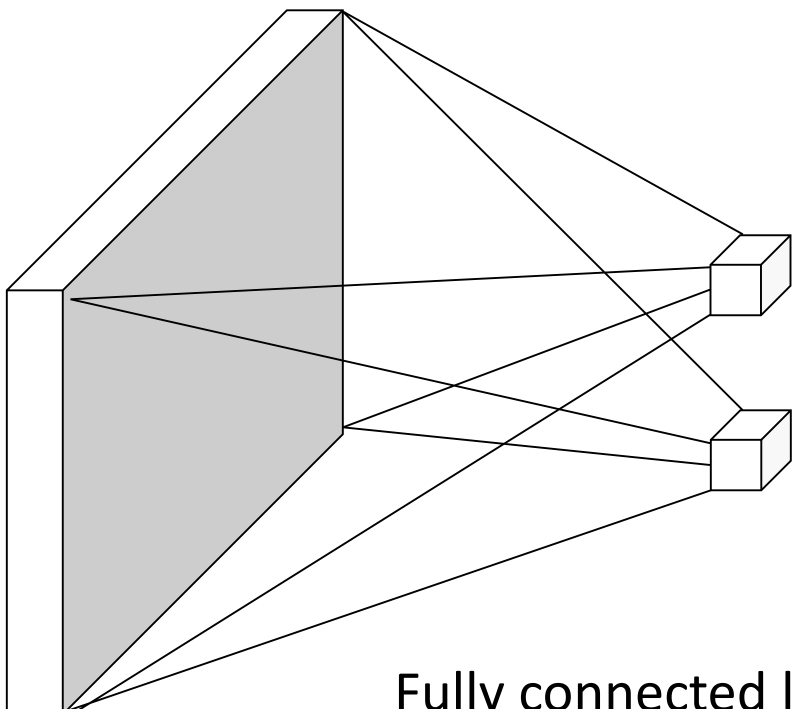
243	239	240	225	206	185	188
242	239	218	110	67	31	34
243	242	123	58	94	82	132
235	217	115	212	243	236	247
233	208	131	222	219	226	196
232	217	131	116	77	150	69
232	232	182	186	184	179	159
232	236	201	154	216	133	129
235	238	230	128	172	138	65
237	236	247	143	59	78	10
234	237	245	193	55	33	115
248	245	161	128	149	109	138



Deep learning for object recognition

- Too many links when applying fully connected layers to images





image

Fully connected layer

For a 1000x1000 image, there are 1M links for a hidden node.

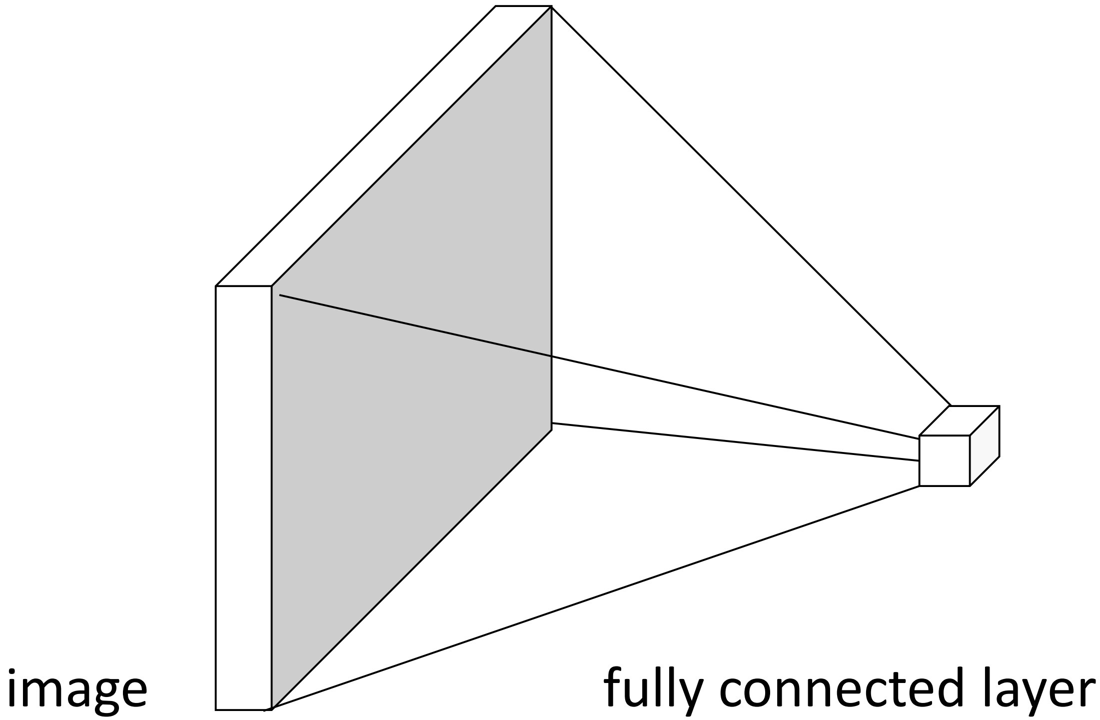
If there are 1M neurons, there are 10^{12} parameters.

Convolutional neural networks (CNN)

- CNN: a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing
- Why local connectivity?
  - Spatial correlation is local (locality of spatial dependencies)
  - Reduce # of parameters

Local connectivity

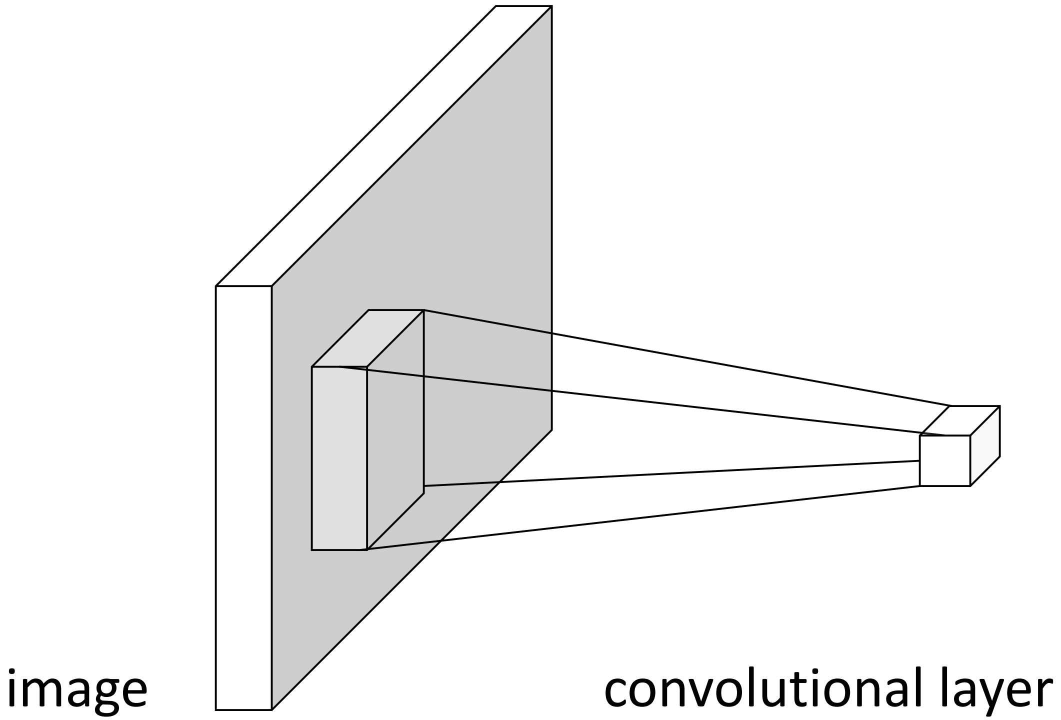
- Each neuron in the convolutional layer is connected a local region, instead of the whole input image



國立交通大學
National Chiao Tung University

Local connectivity

- Each neuron in the convolutional layer is connected a local region, instead of the whole input image

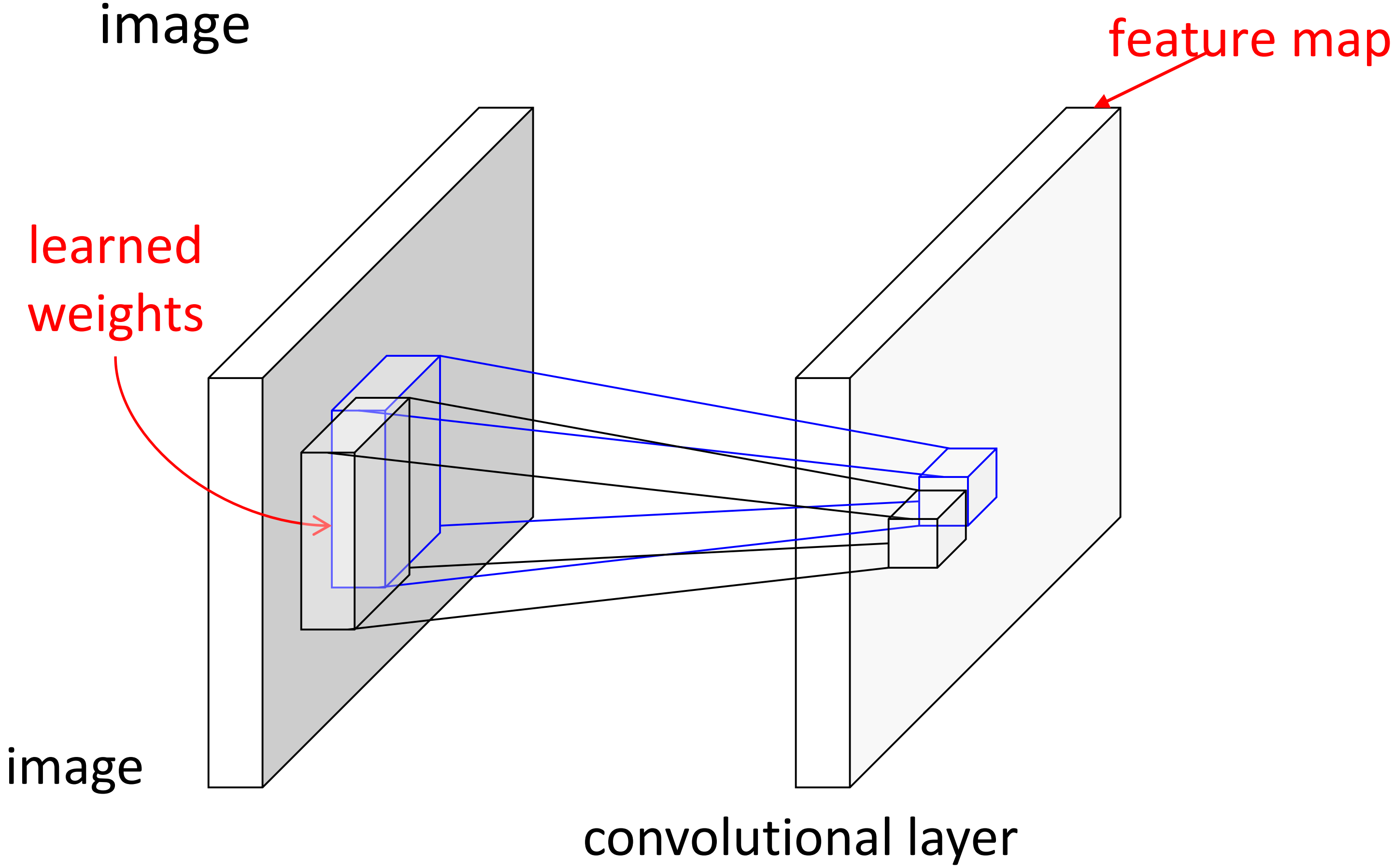


Convolutional neural networks (CNN)

- CNN: a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing
- Why local connectivity?
  - Spatial correlation is local (locality of spatial dependencies)
  - Reduce # of parameters
- Why weight sharing?
  - Same statistics is at different locations (stationarity of statistics)
  - Reduce # of parameters

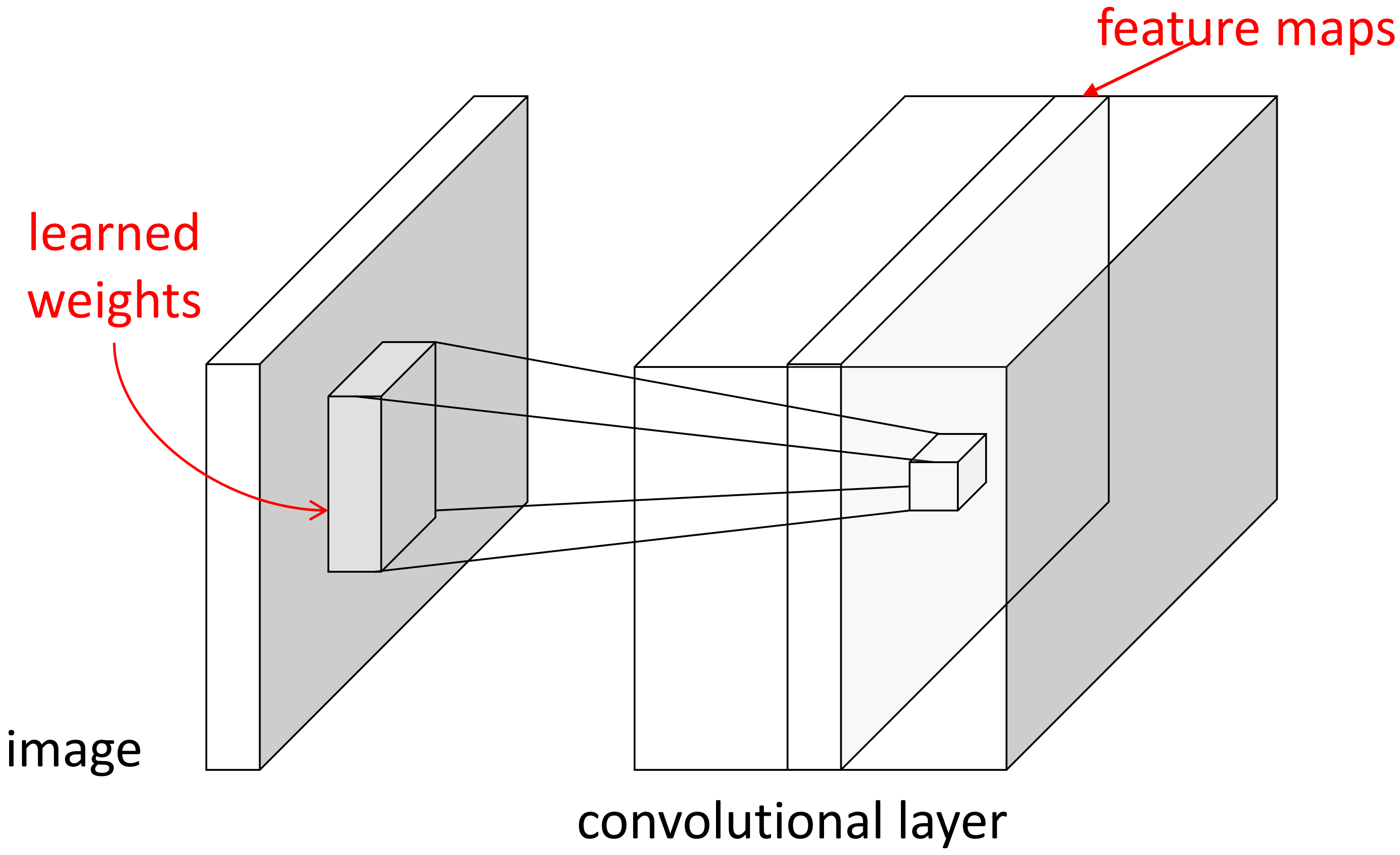
Weight sharing

- We compute the same statistics at different locations of an image

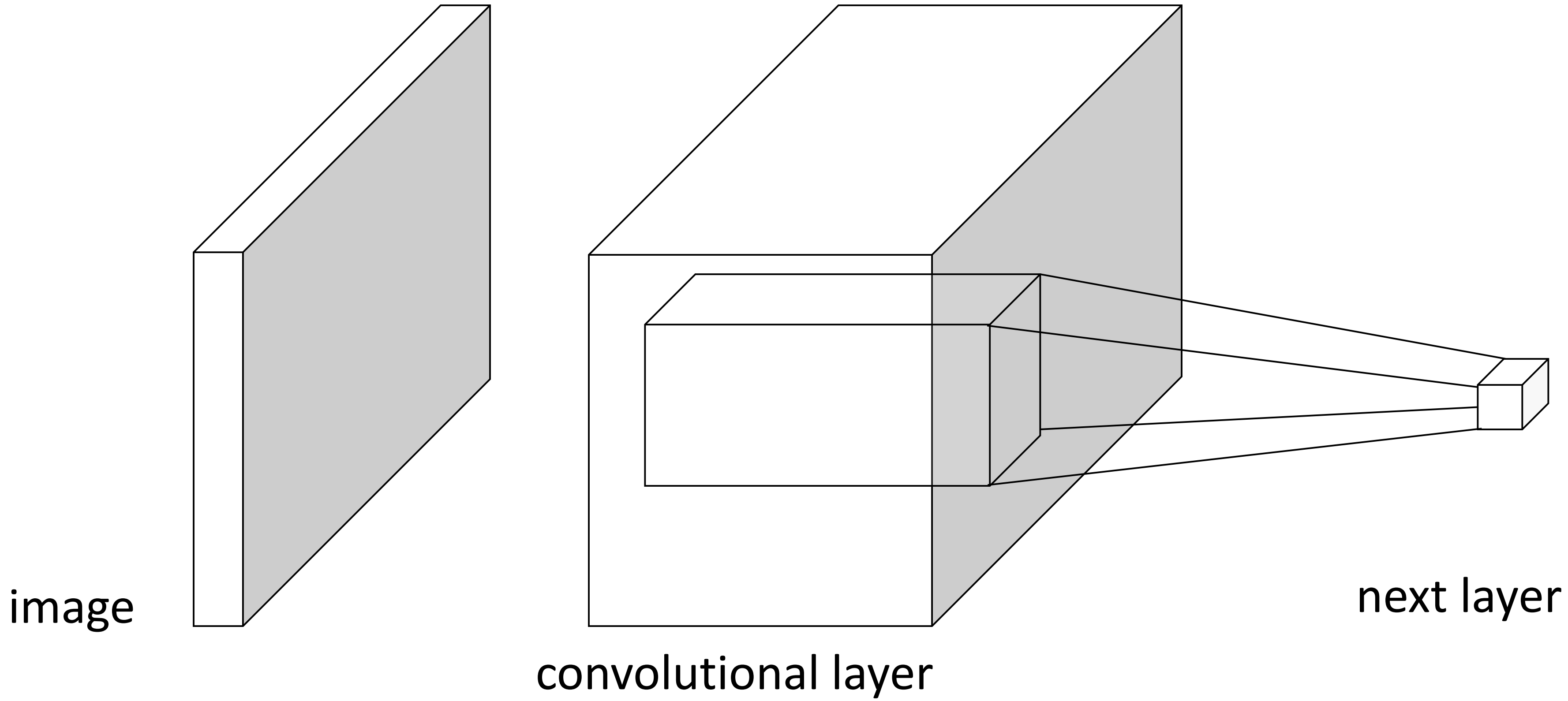


国立交通大學
National Chiao Tung University

Multiple feature maps by learning multiple filters



Stacking convolutional layers

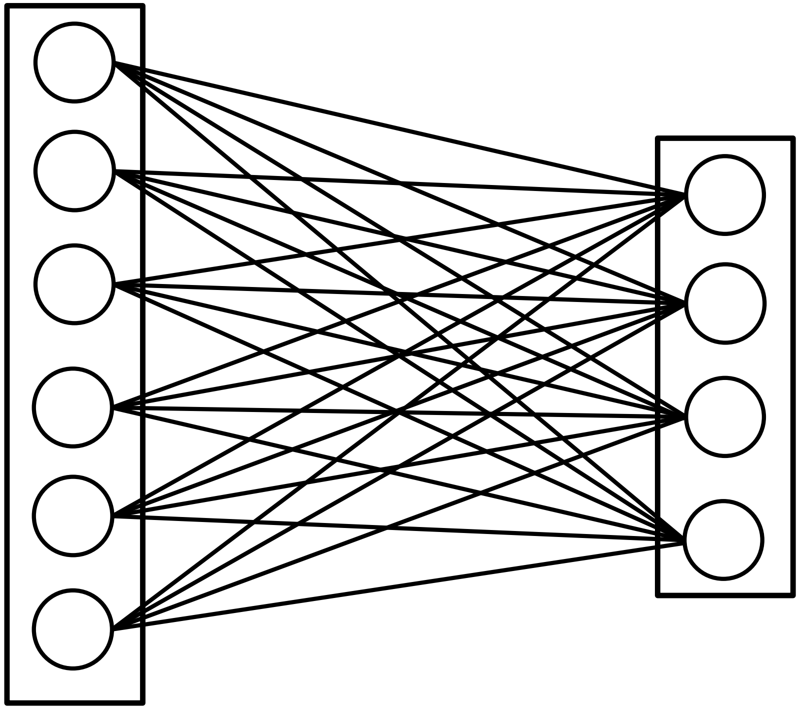


Building blocks of CNN

- Convolutional layer (CONV)
- Pooling layer (POOL)
- Fully connected layer (FC)

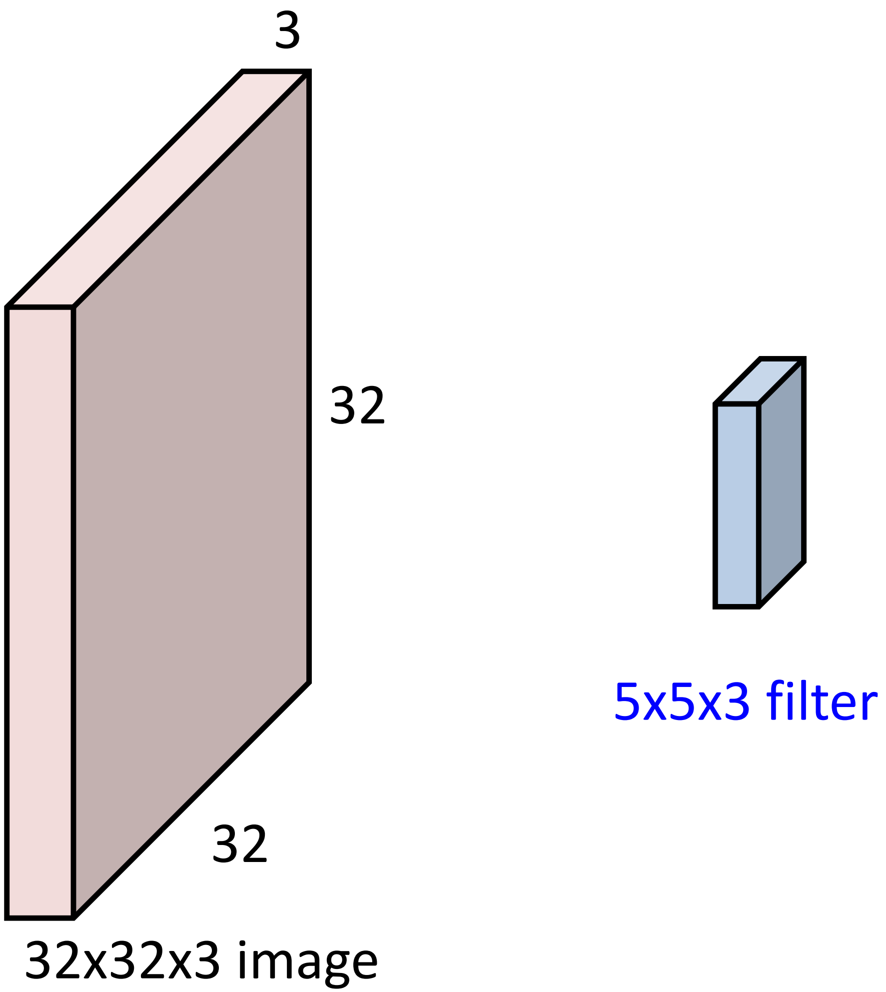
Fully connected layer

- A neuron is fully connected to all neurons in the previous layer
- Perform biased weighted sum followed by a non-linear mapping
- Deliver the output to all neurons in the next layer

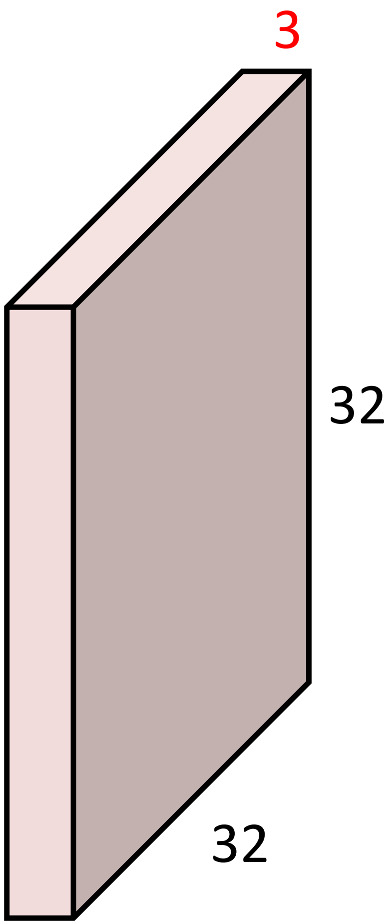


国立交通大學
National Chiao Tung University

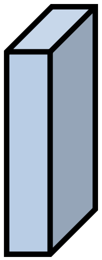
Convolutional layer



Convolutional layer



32x32x3 image

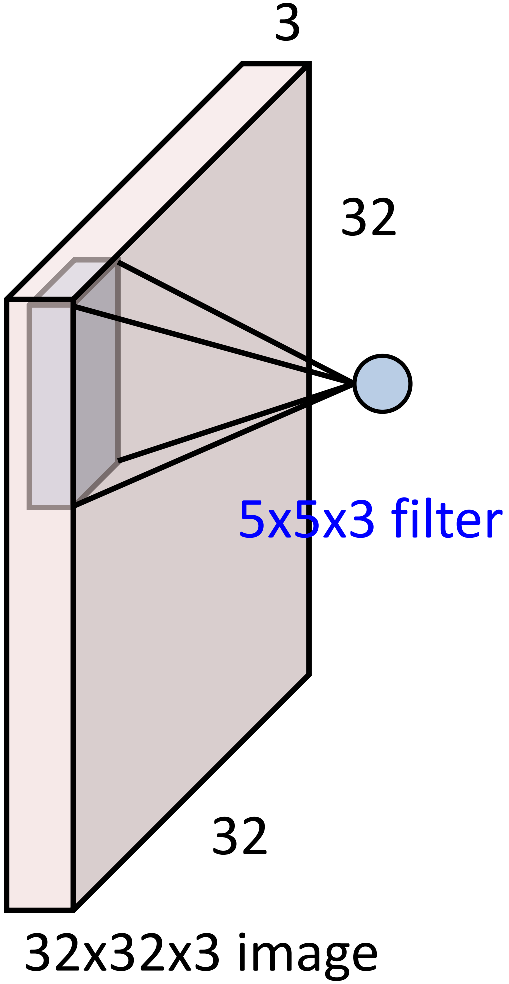


5x5x3 filter

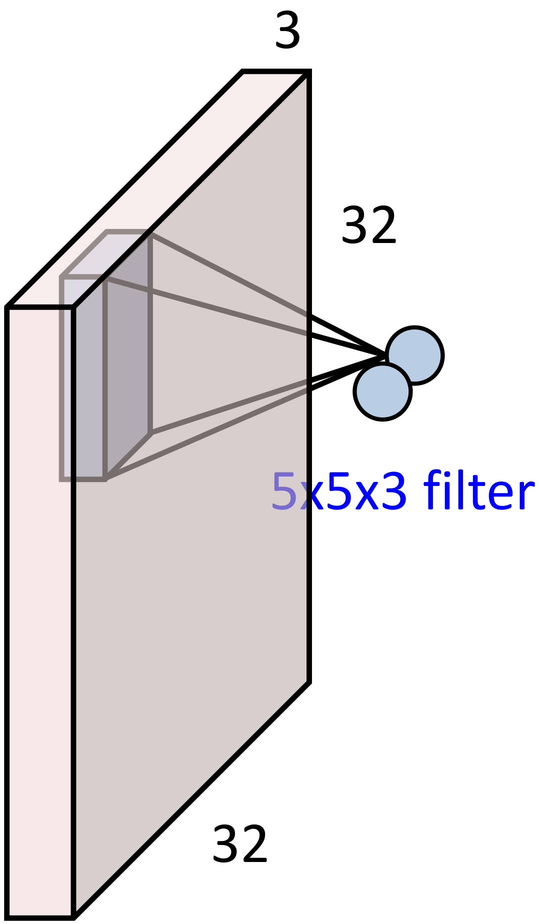
- The dimension of a filter is 3

- The third dimension is the same as the number of channels of the input image

Convolutional layer: filtering + non-linear mapping

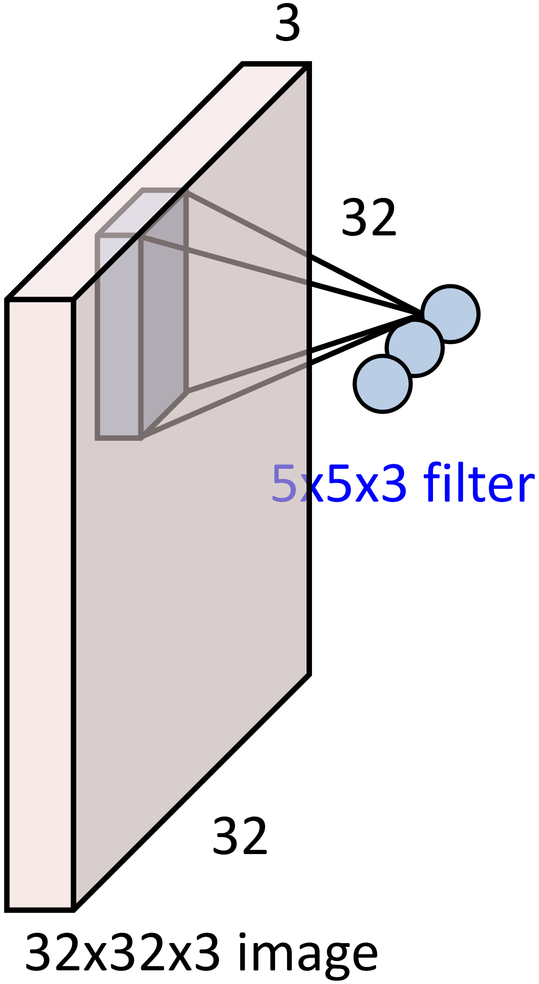


Convolutional layer: filtering + non-linear mapping

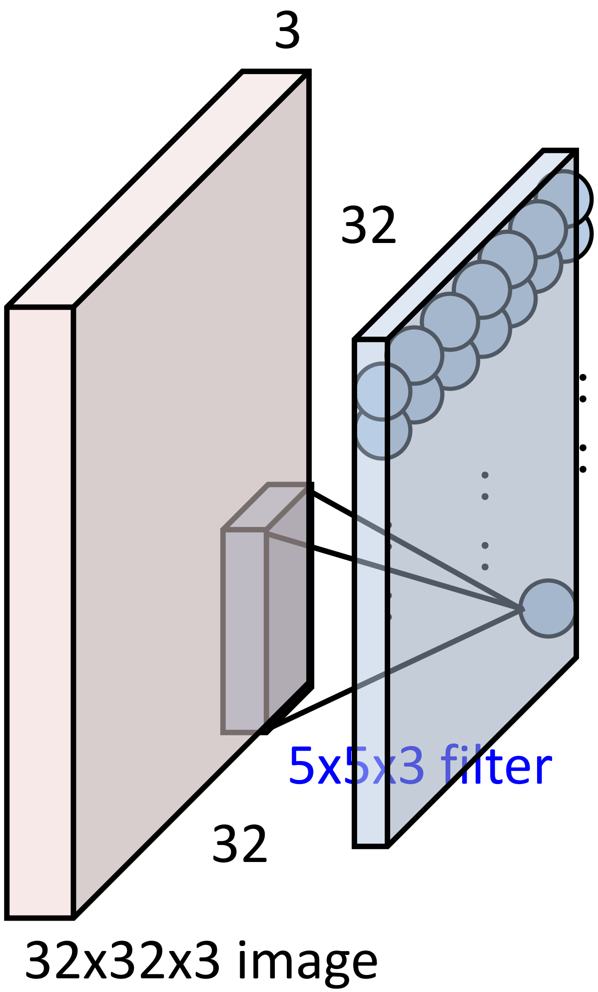


國立交通大學
National Chiao Tung University

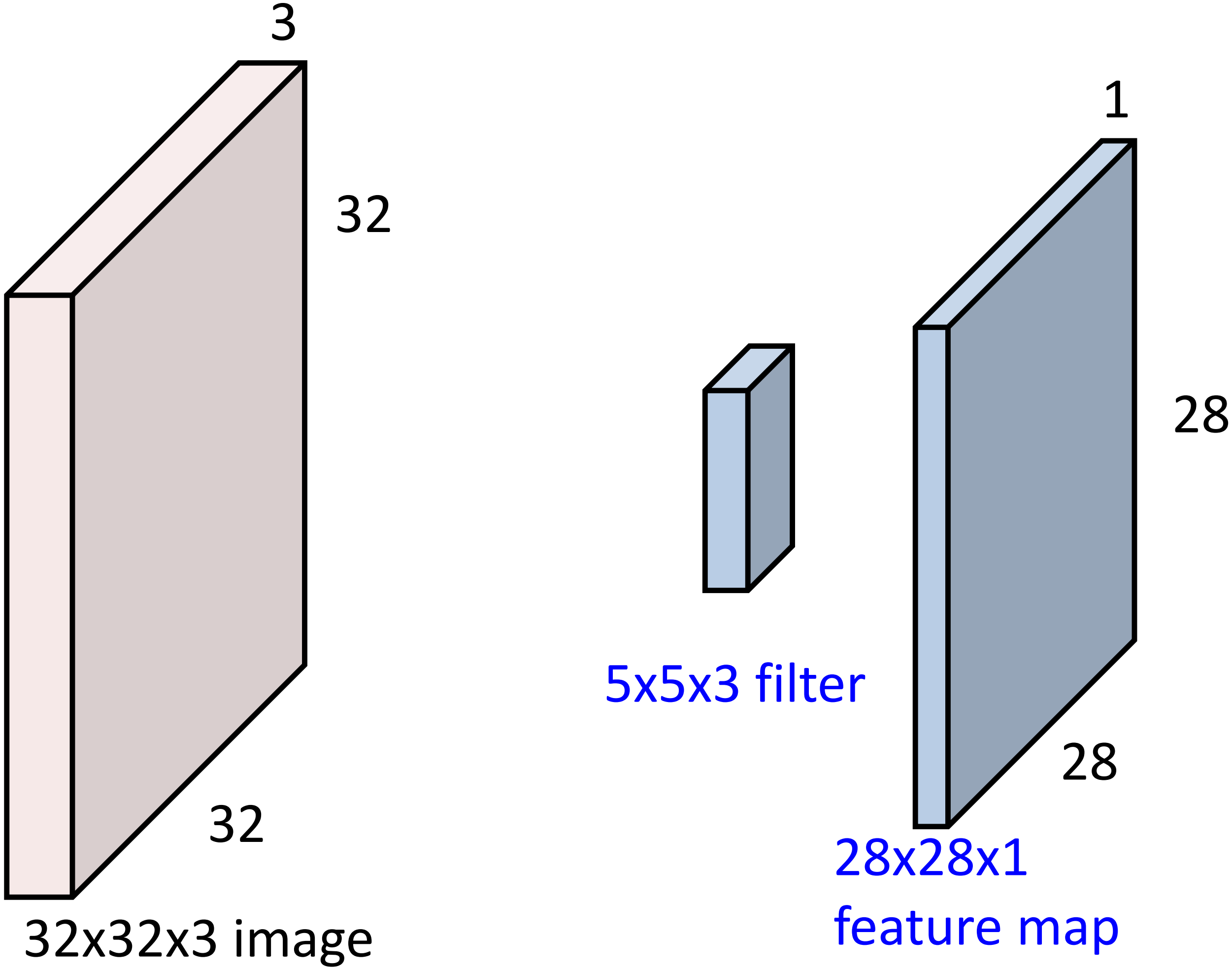
Convolutional layer: filtering + non-linear mapping



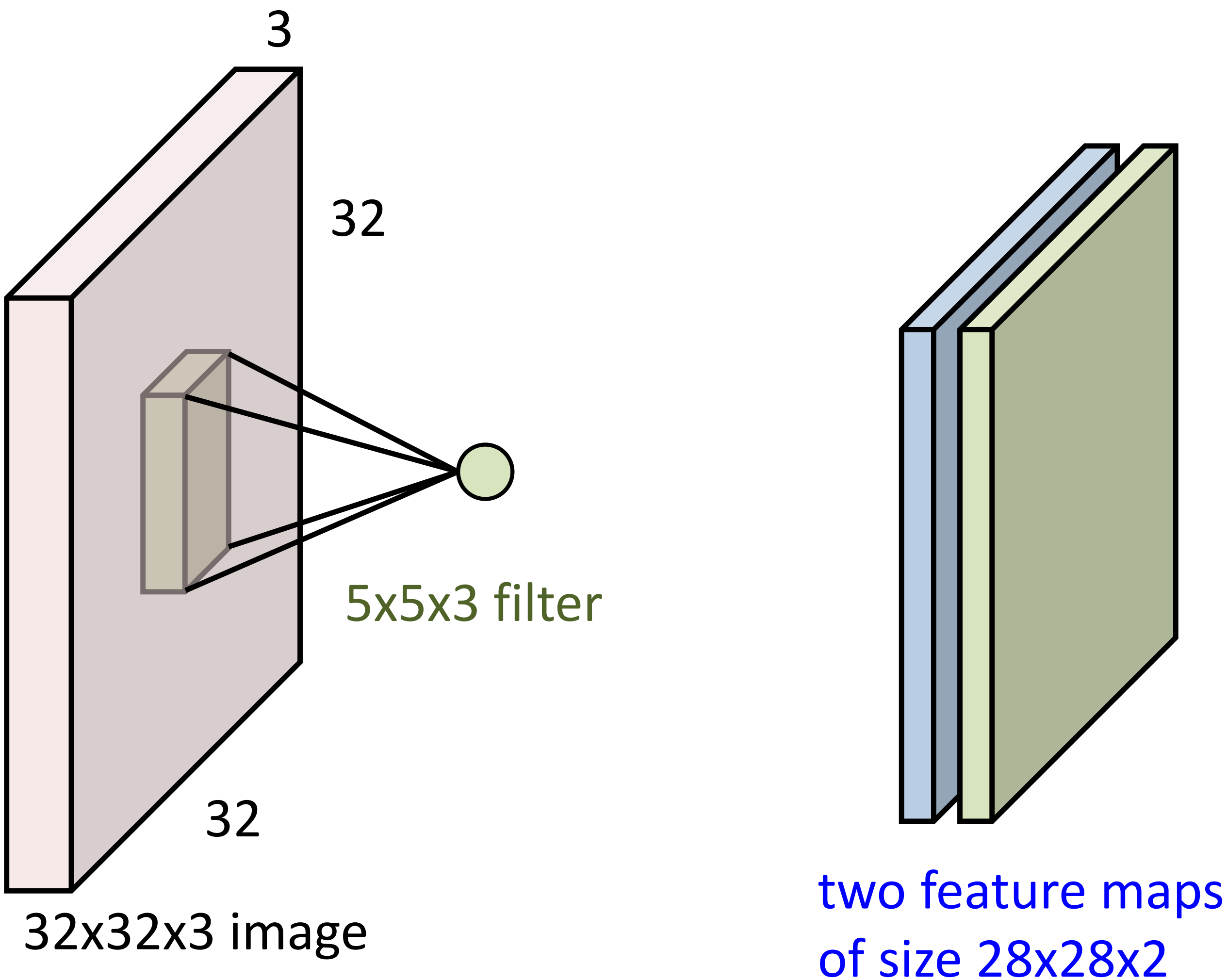
Convolutional layer: filtering + non-linear mapping



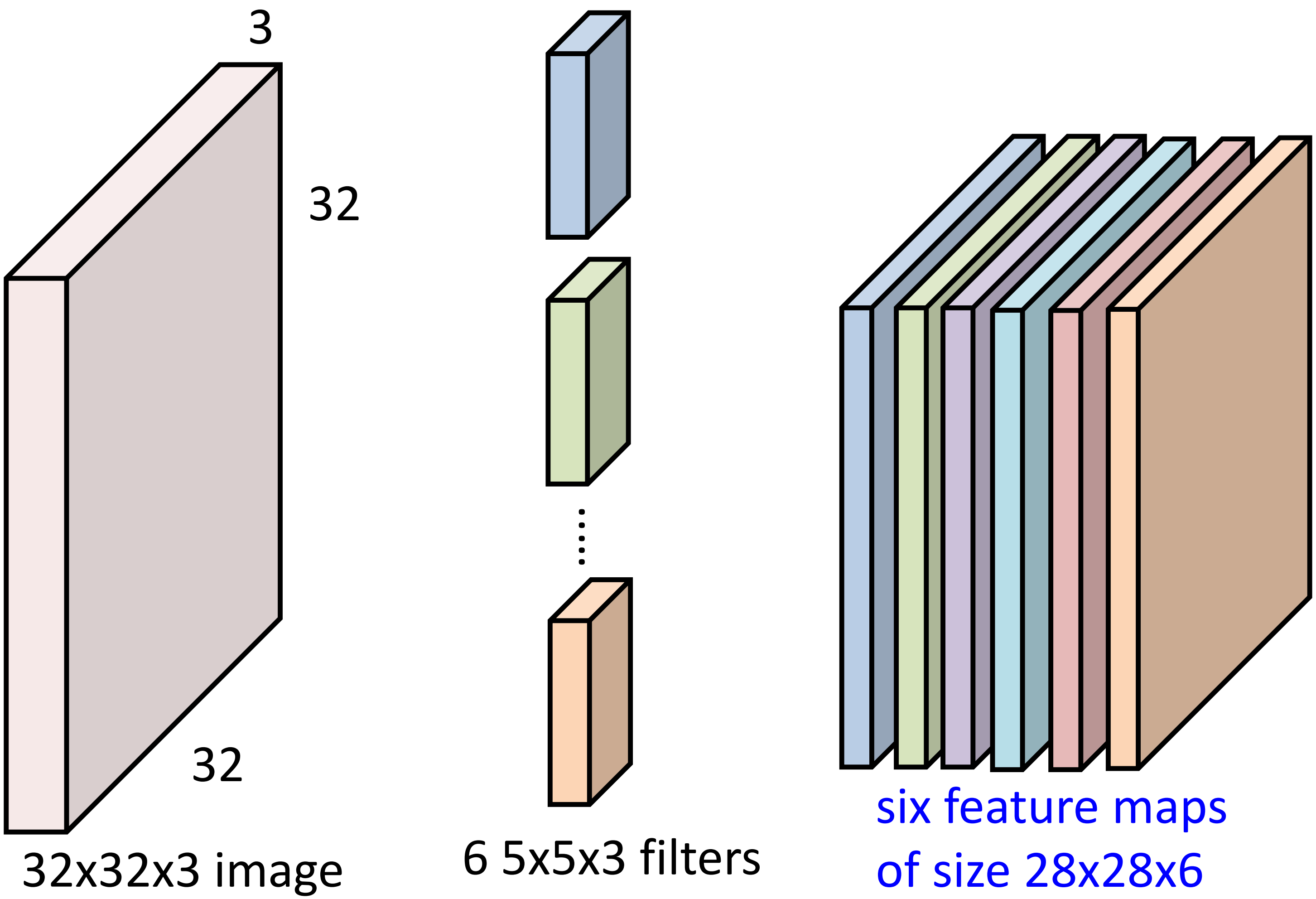
Convolutional layer: A filter yields a feature map



Convolutional layer: The second filter

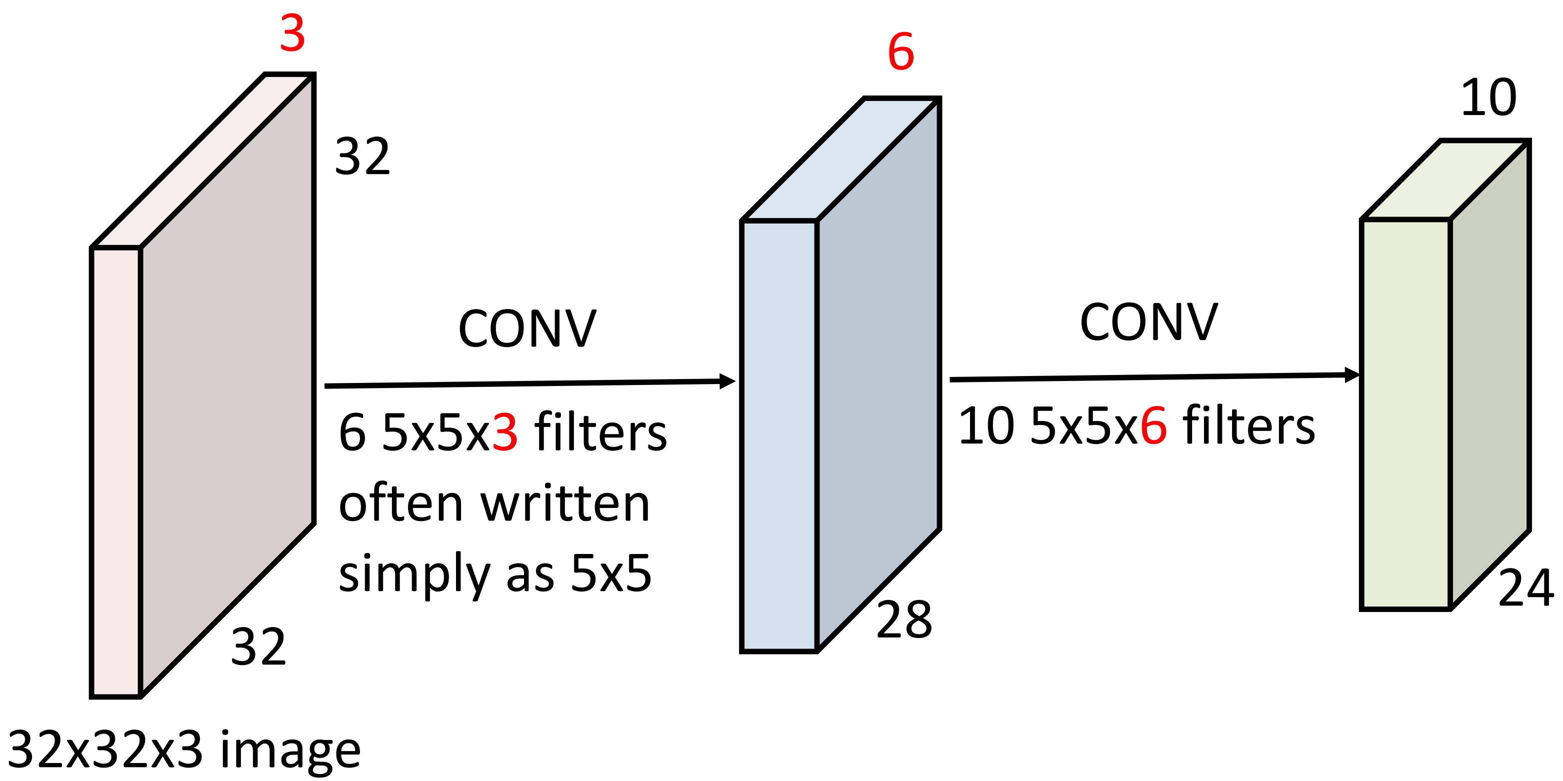


Convolutional layer: N filters yield N feature maps



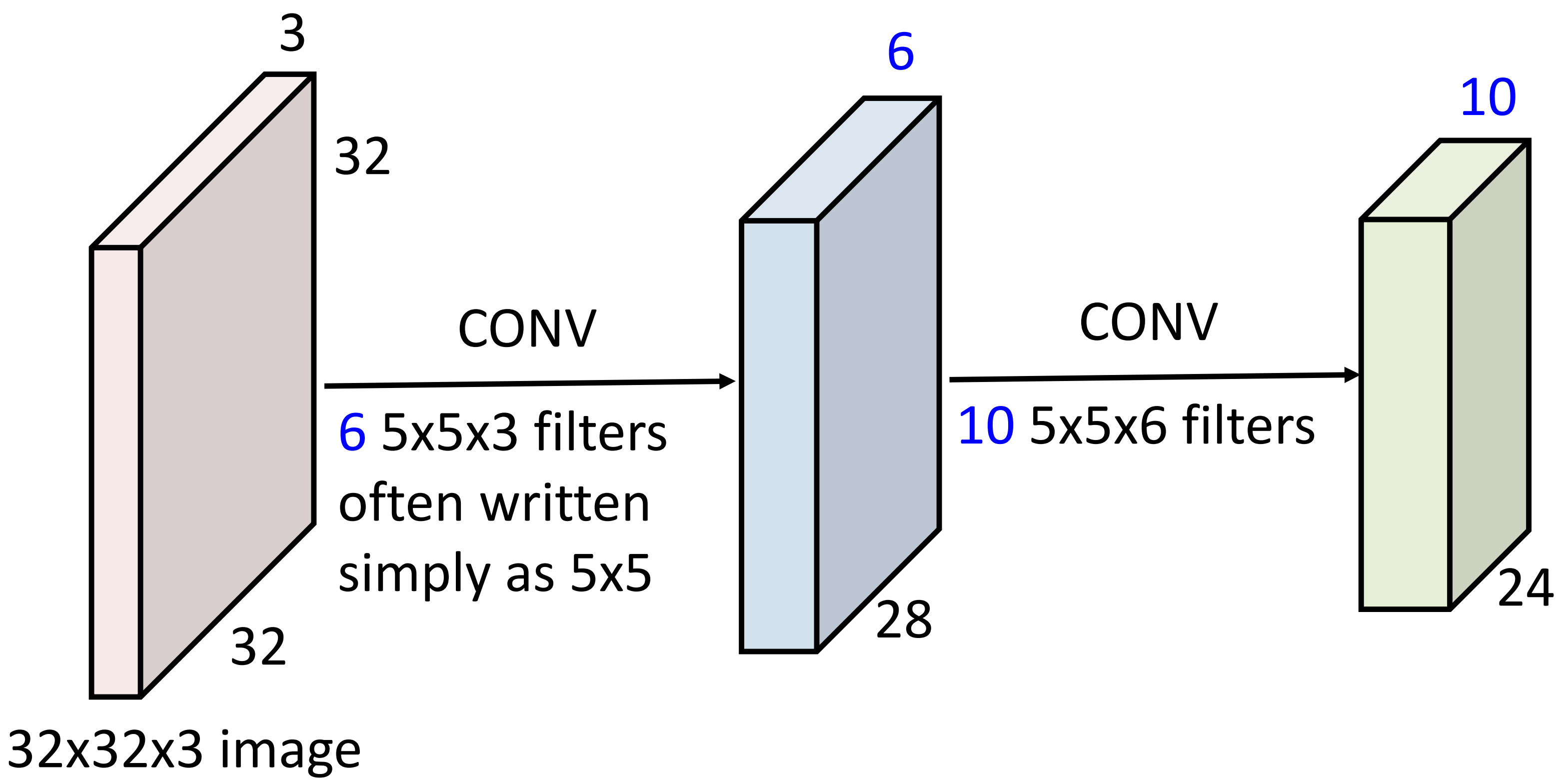
Convolutional layers

- A convolution (CONV) layer performs convolution followed by an activation function
- CNN often consists of several CONV layers.



Convolutional layers

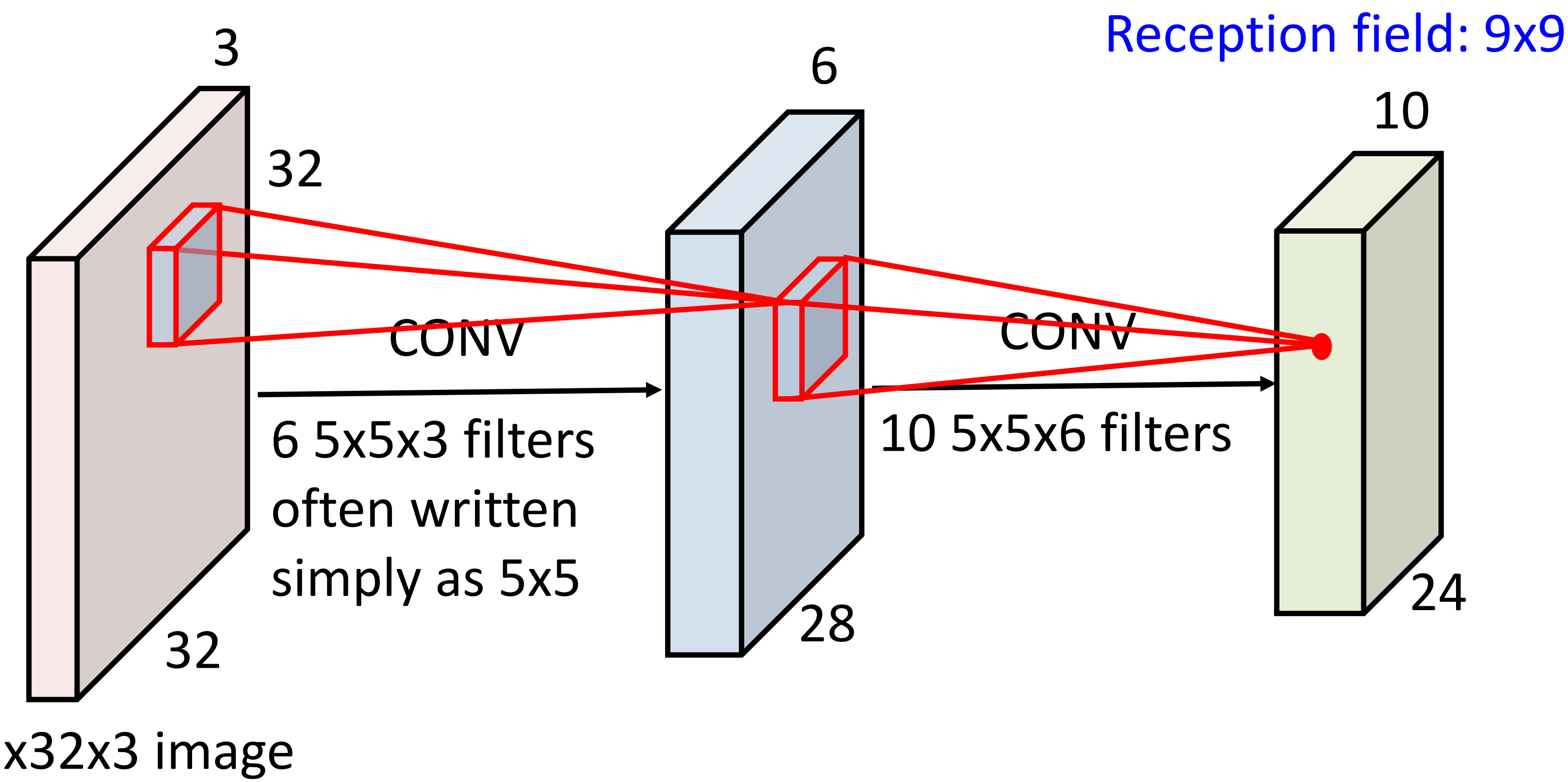
- A convolution (CONV) layer performs convolution followed by an activation function
- CNN often consists of several CONV layers.



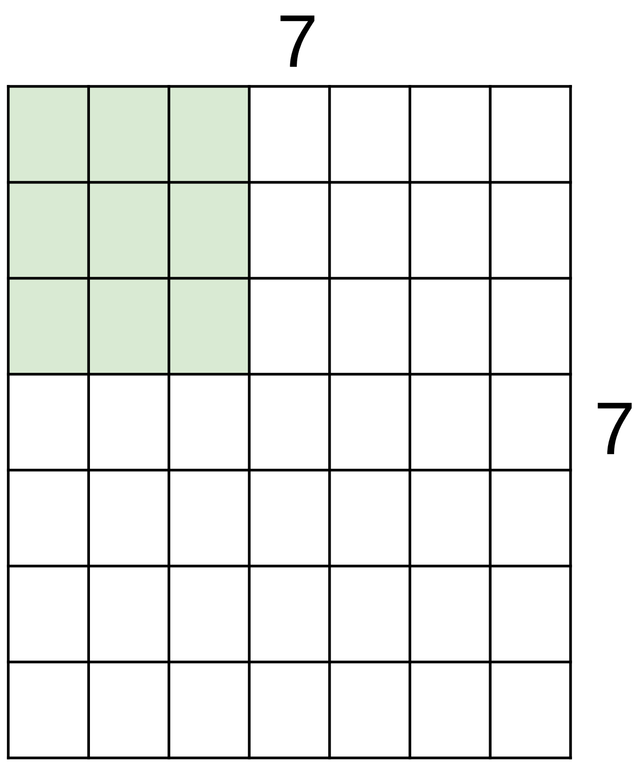
國立交通大學 National Chiao Tung University

Convolutional layer: Receptive field

- How large is the area influencing a pixel of the activation map on the input image

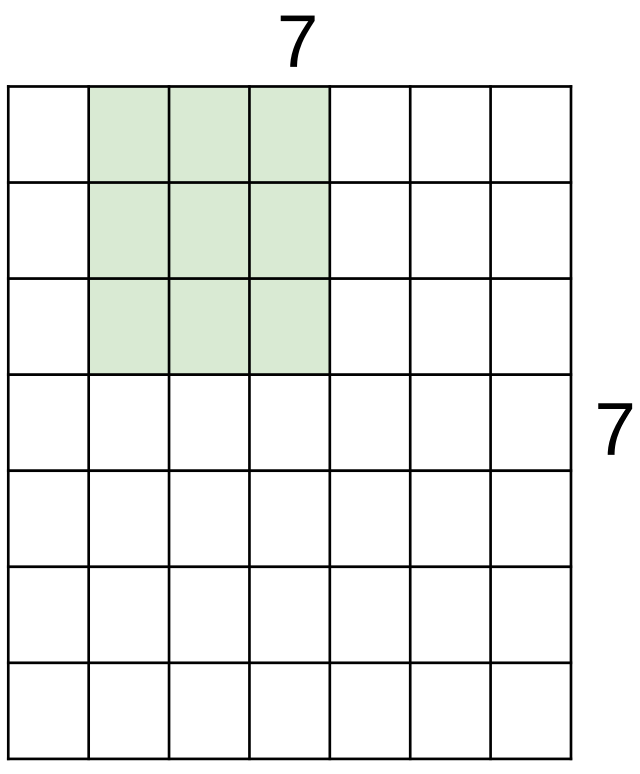


Convolutional layer: Size of a feature map



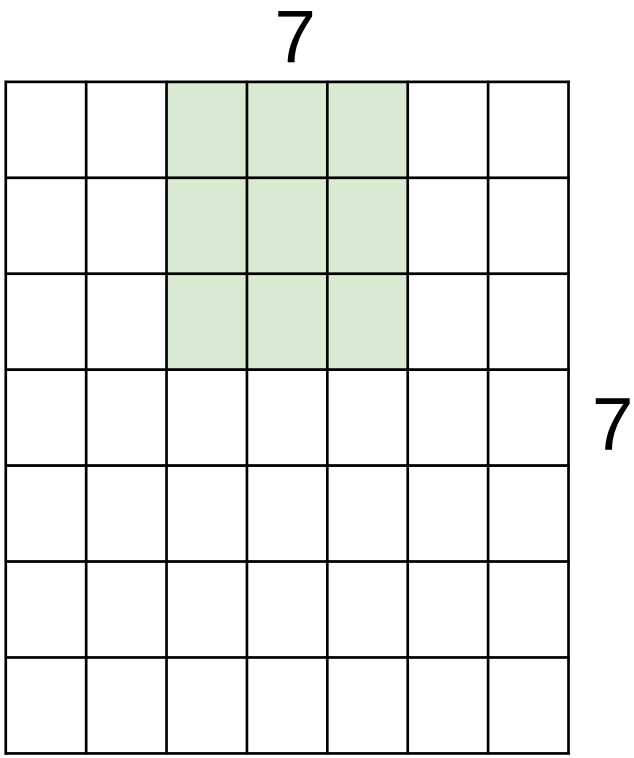
- Assume the input is 7x7 and the filter is 3x3, what is the size of the output?

Convolutional layer: Size of a feature map



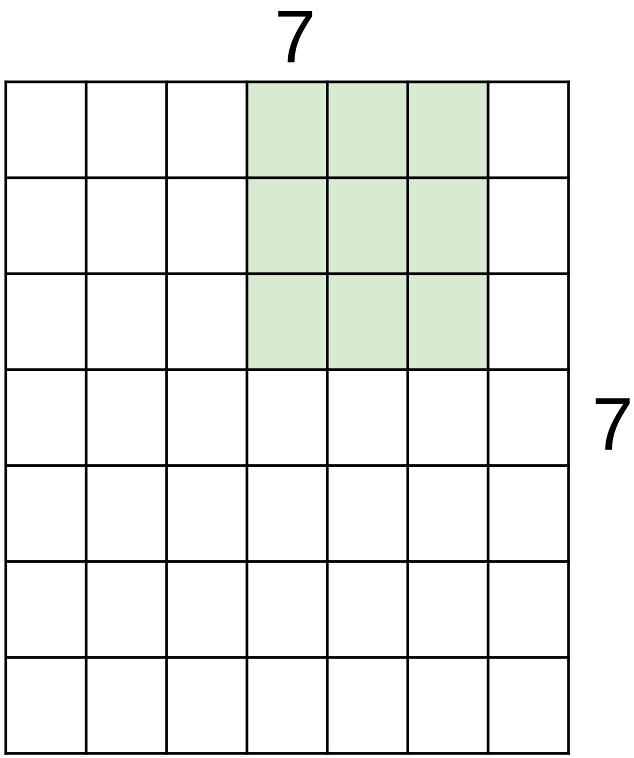
- Assume the input is 7x7 and the filter is 3x3, what is the size of the output?

Convolutional layer: Size of a feature map



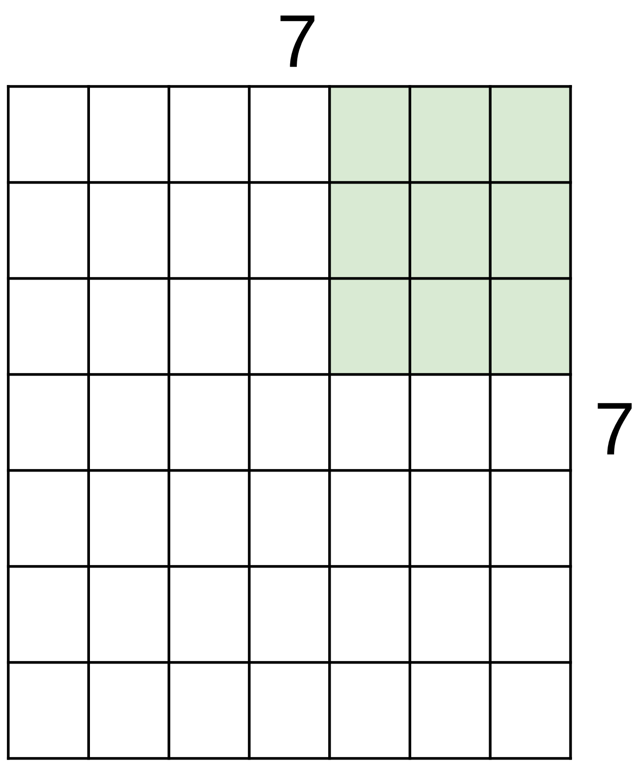
- Assume the input is 7x7 and the filter is 3x3, what is the size of the output?

Convolutional layer: Size of a feature map



- Assume the input is 7x7 and the filter is 3x3, what is the size of the output?

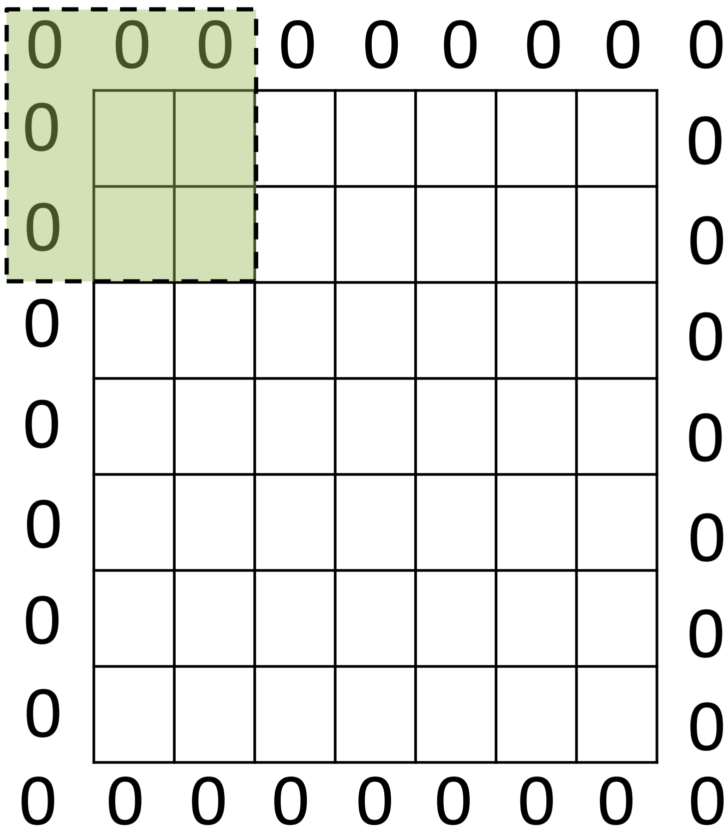
Convolutional layer: Size of a feature map



- Assume the input is 7x7 and the filter is 3x3, what is the size of the output?

5x5

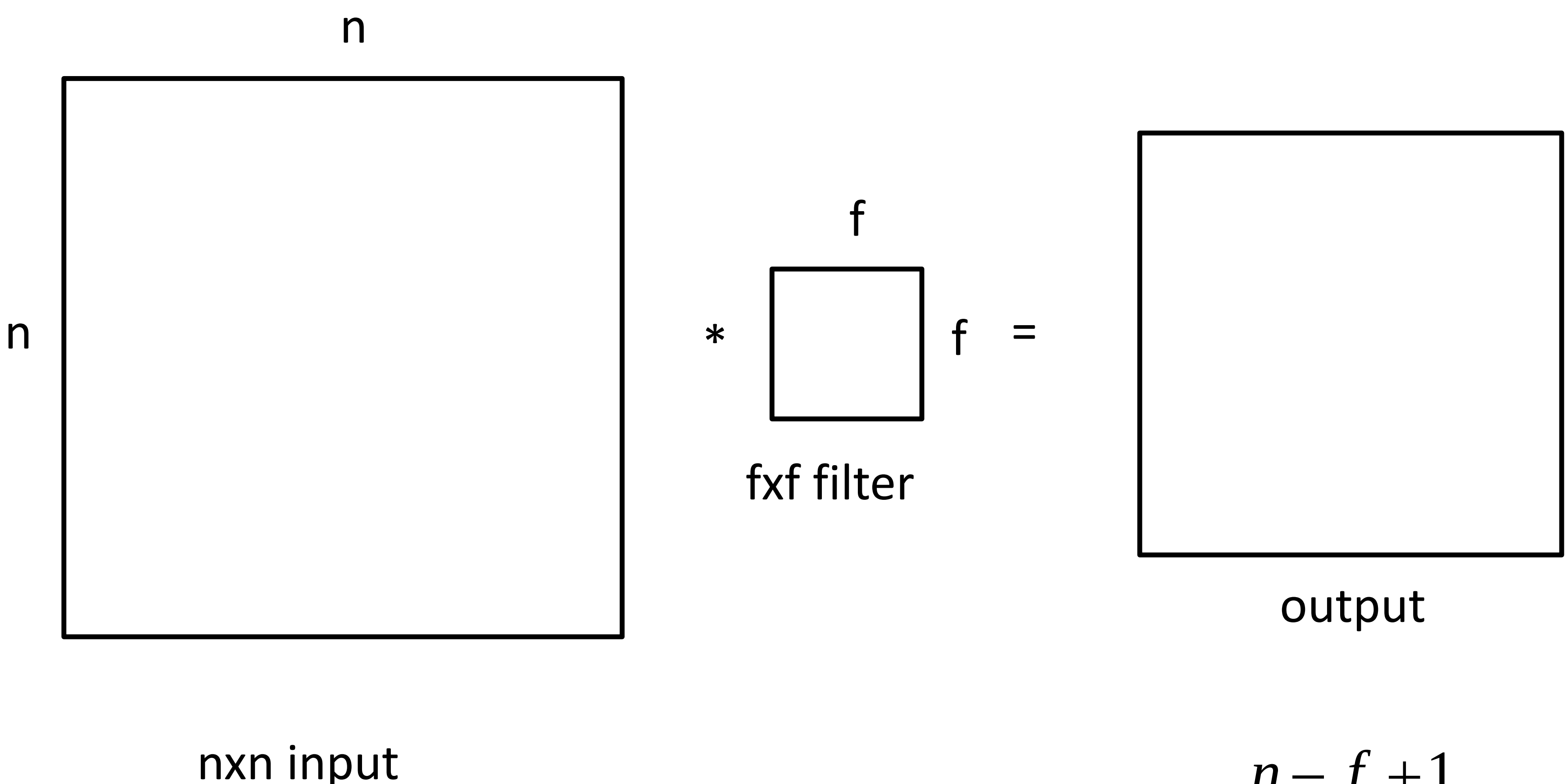
Padding



- The map will be smaller and smaller and the information on the border is lost.
- What if we want to keep the size the same?

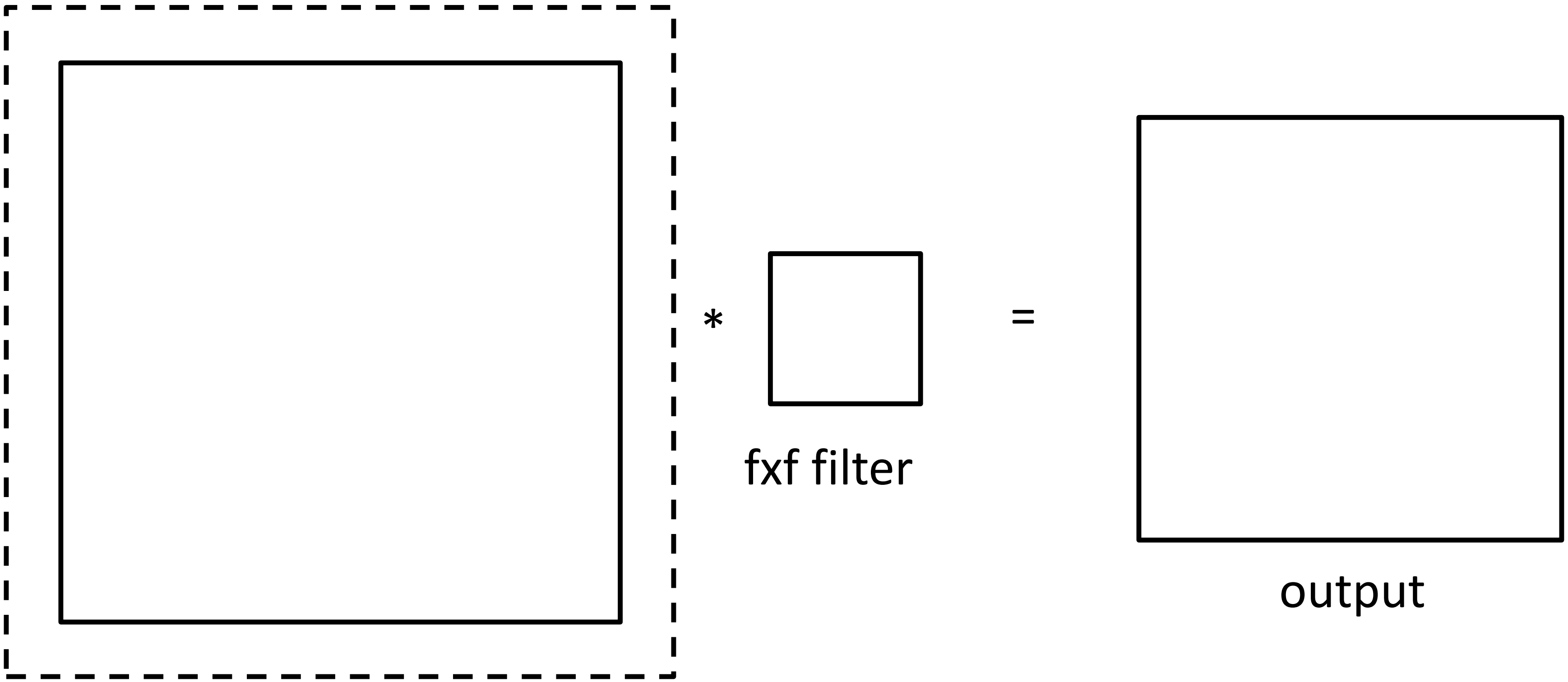
zero-padding

Size of the map



$$n - f + 1$$

Size of the map



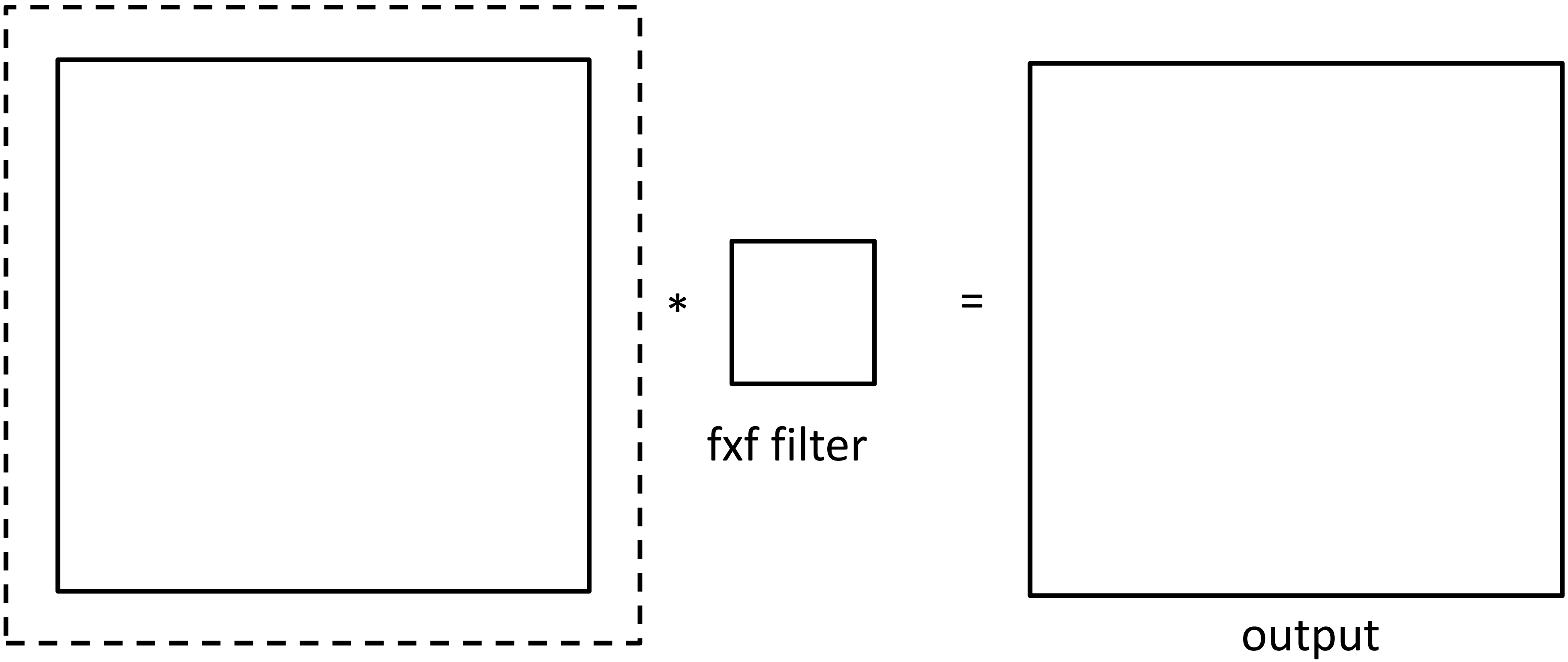
nxn input

padding p

$$n + 2p - f + 1$$

National Chiao Tung University

Size of the map



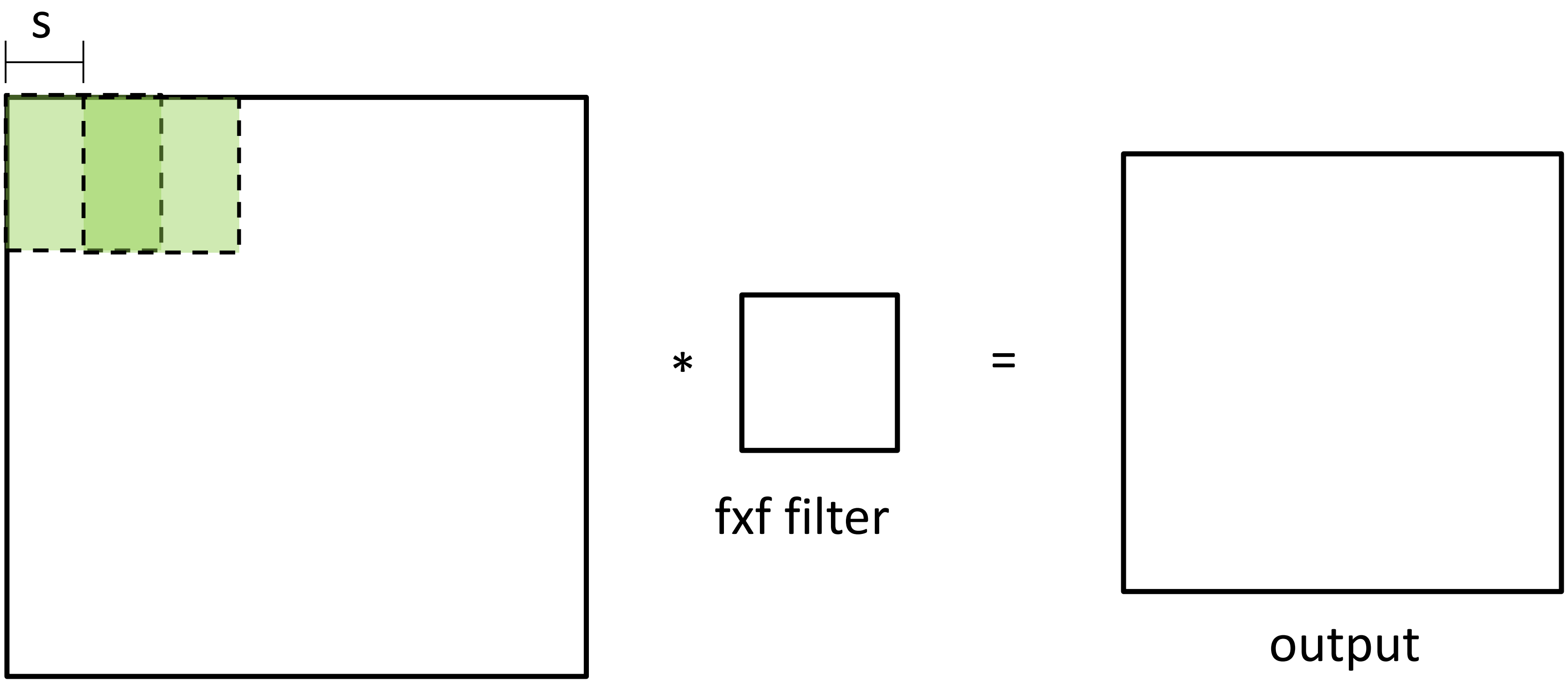
nxn input

padding p

Keep the same size: $n+2p-f+1=n$

$$p=\frac{f-1}{2}$$

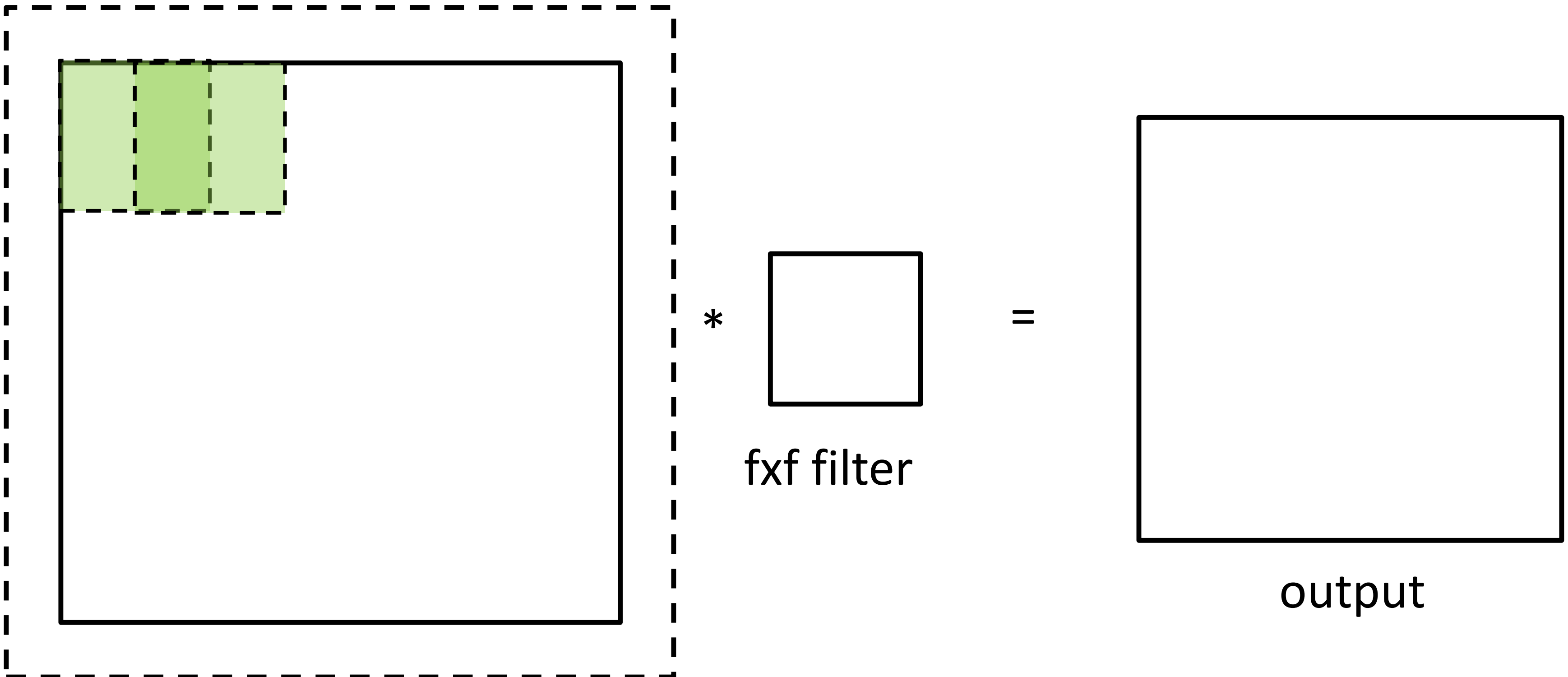
Size of the map



nxn input

stride s

Size of the map



nxn input

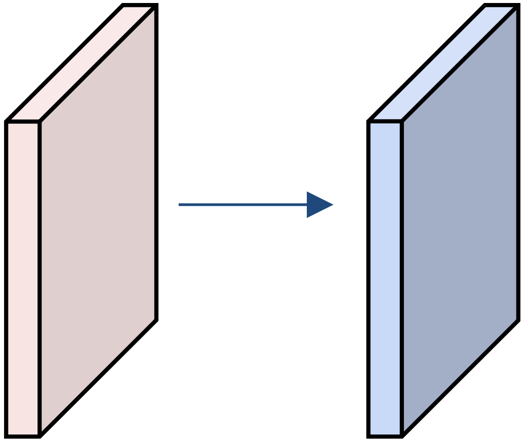
padding p, stride s

$$\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$$

An example

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?



An example

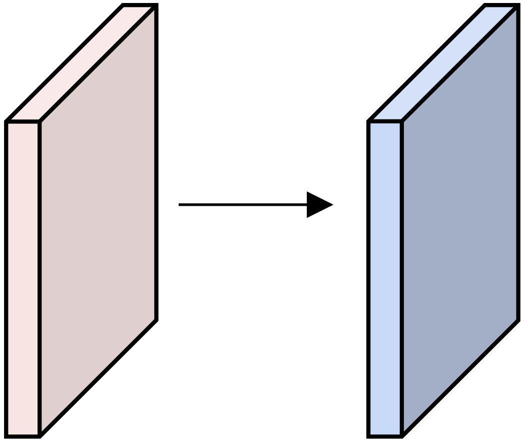
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10



An example

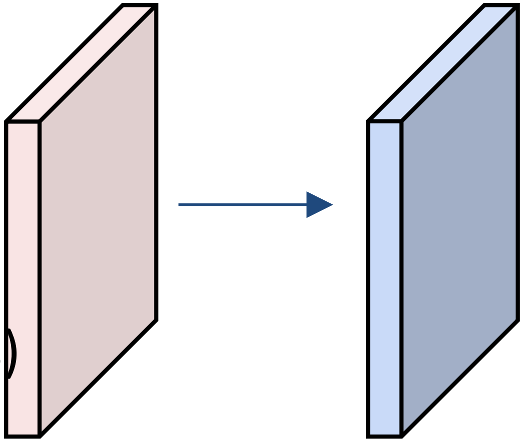
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

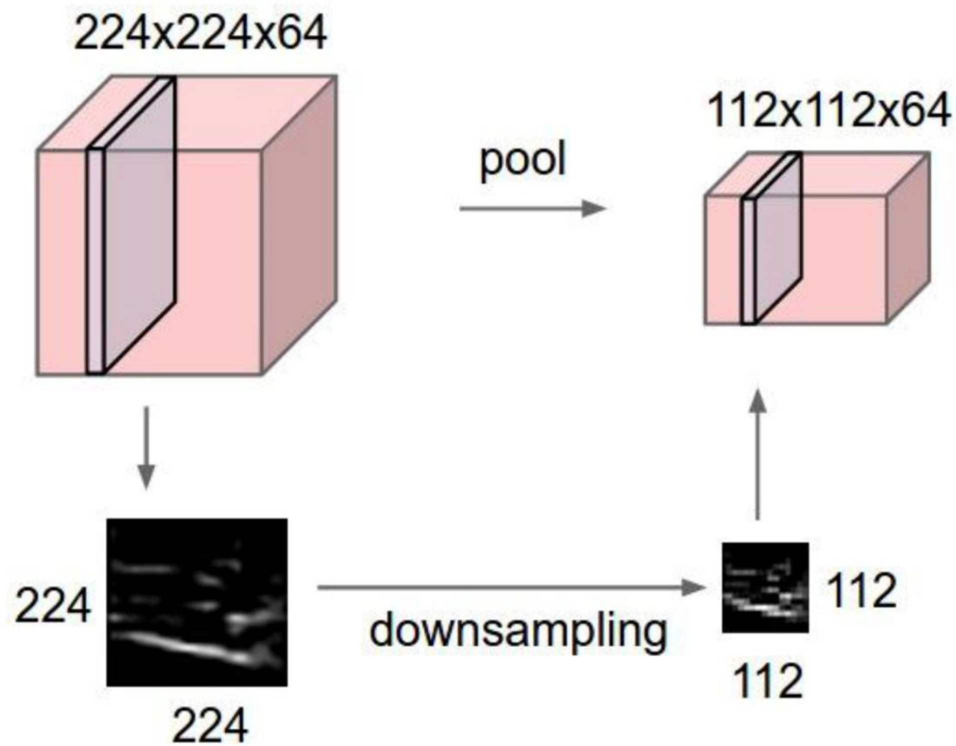
each filter has 5*5*3 + 1 = 76 params (+1 for bias)

=> 76*10 = **760**



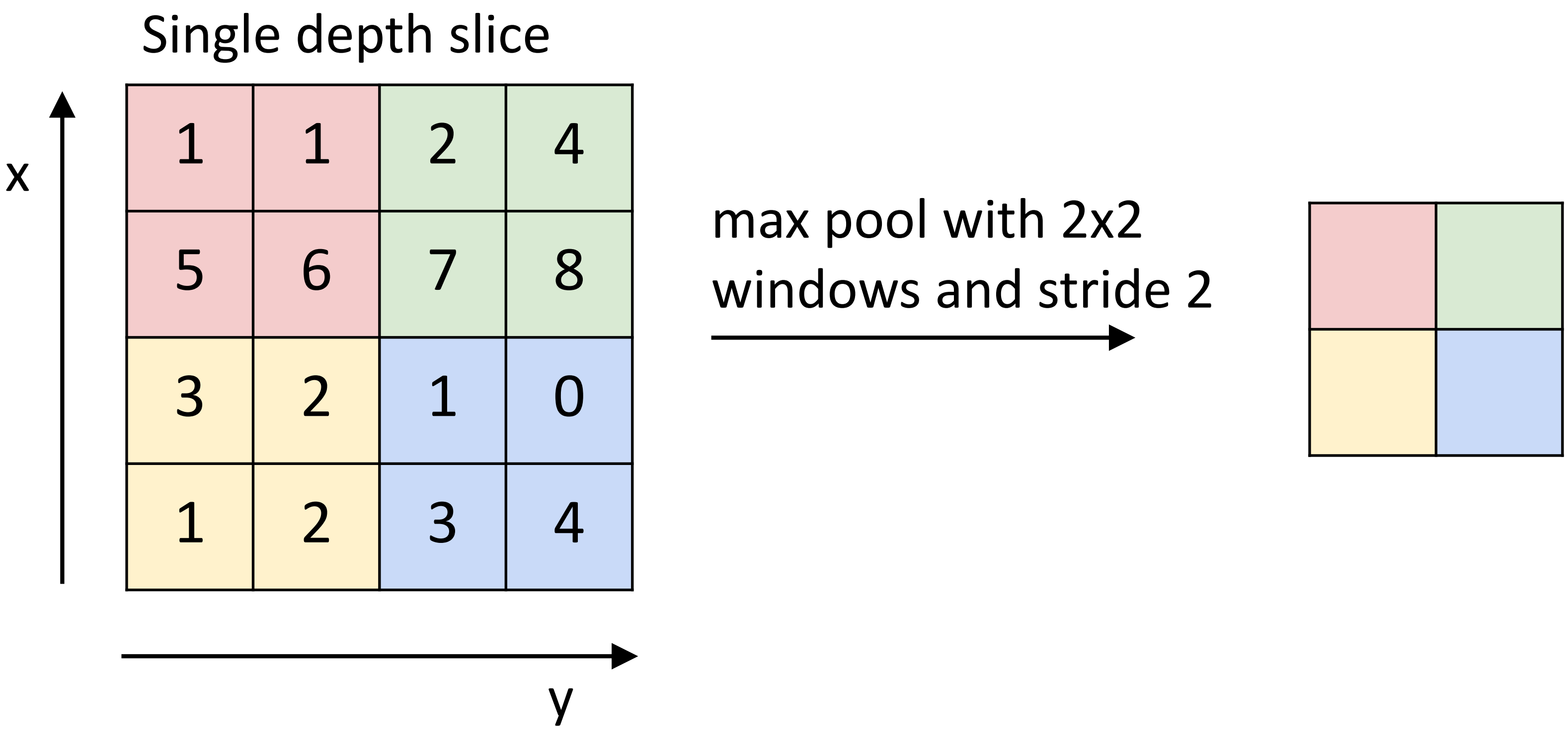
Pooling layer

- Makes the representations smaller and more manageable
- Operates over each feature map independently



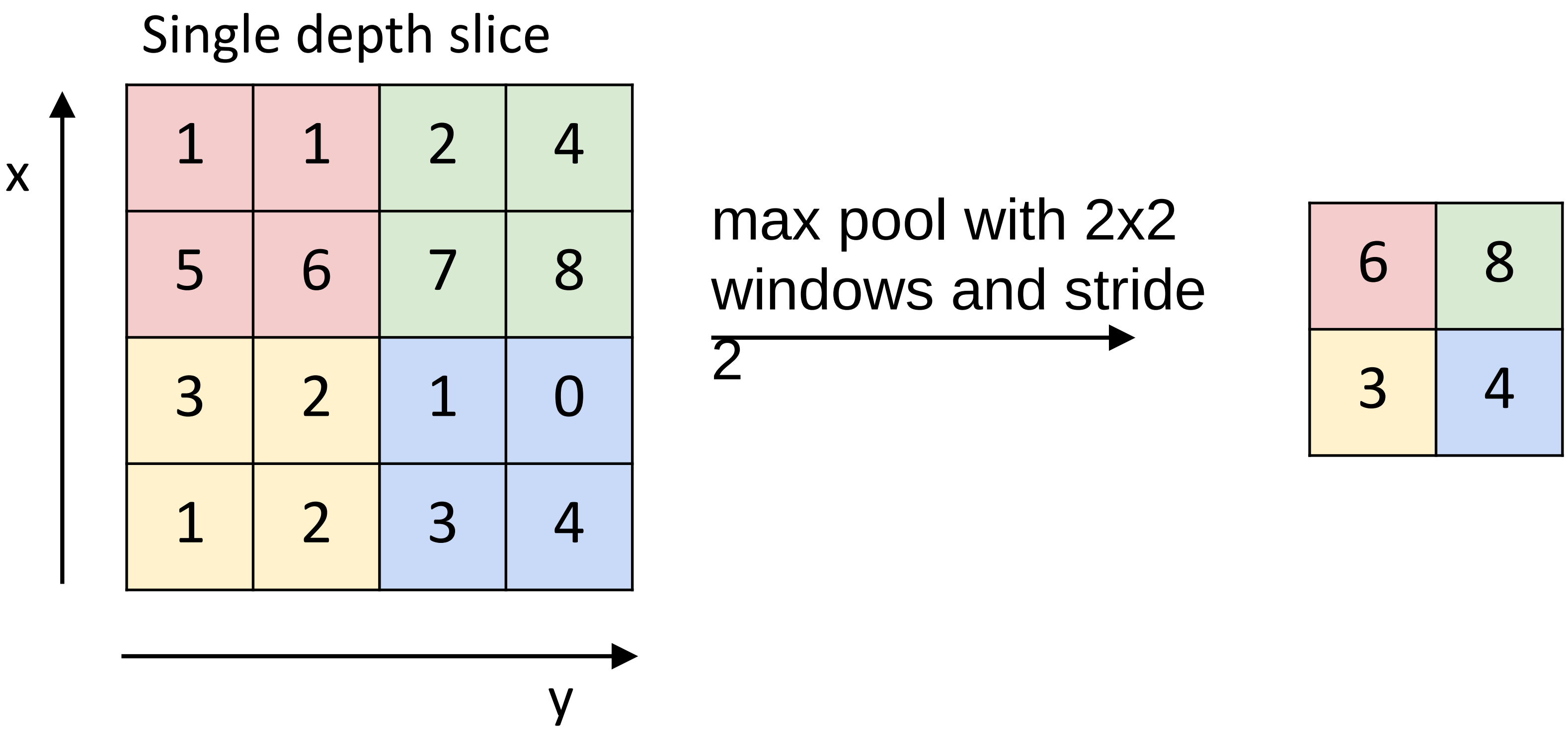
Max pooling

- There is no parameter in the layer

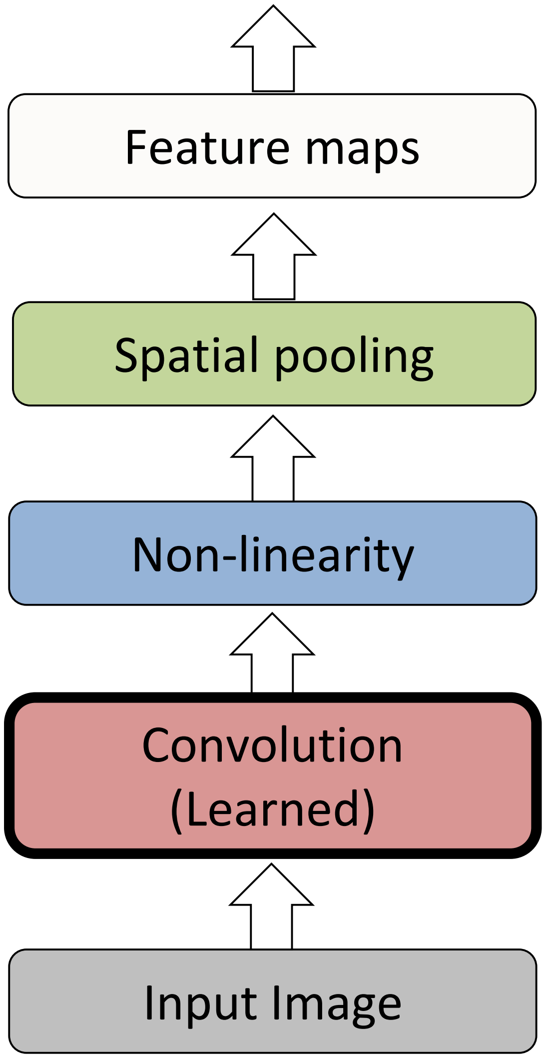


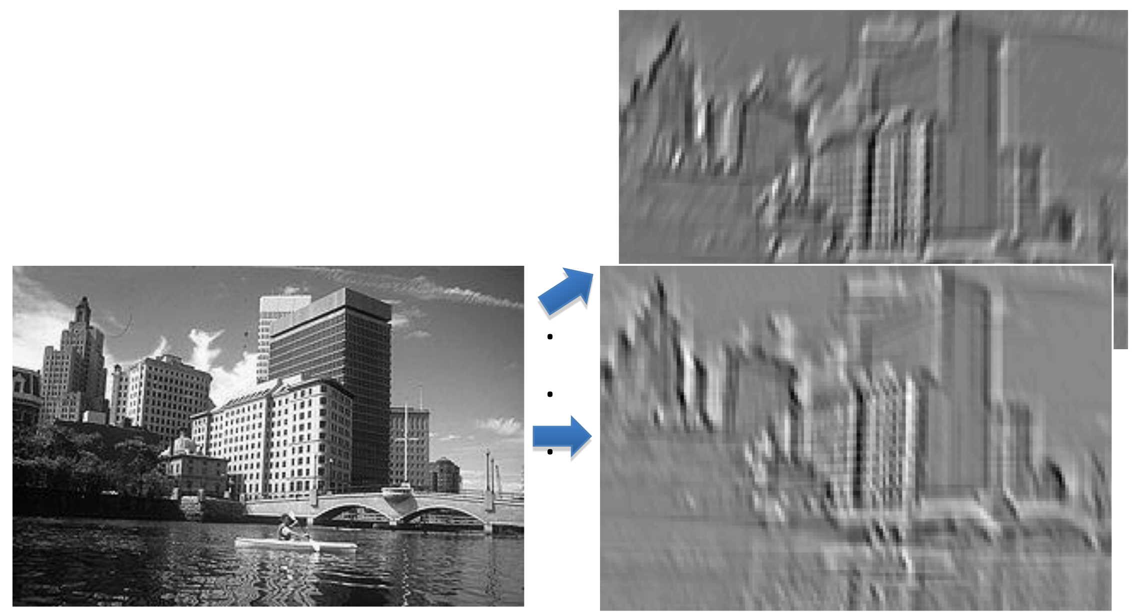
Max pooling

- There is no parameter in the layer



Operations of CNN



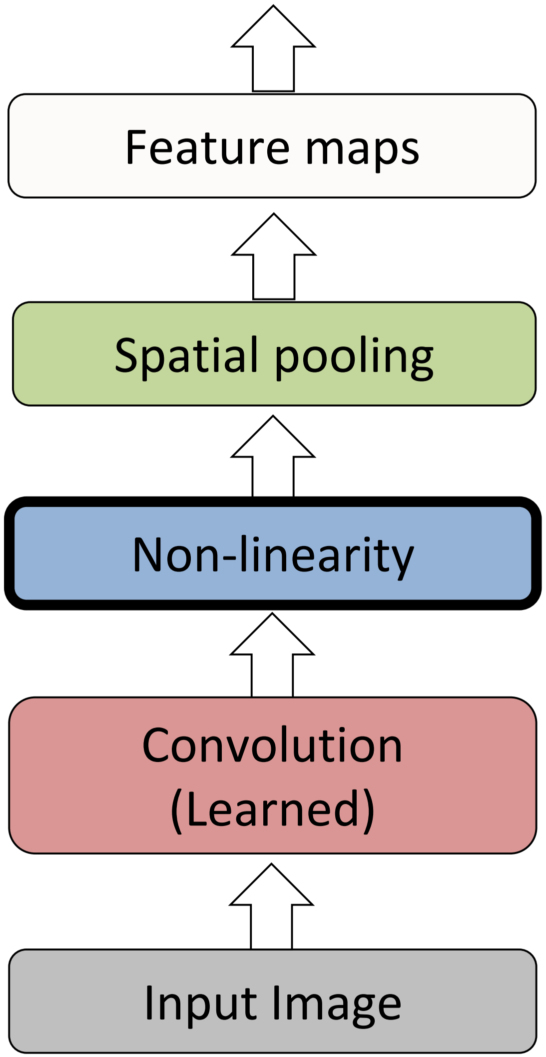


Input

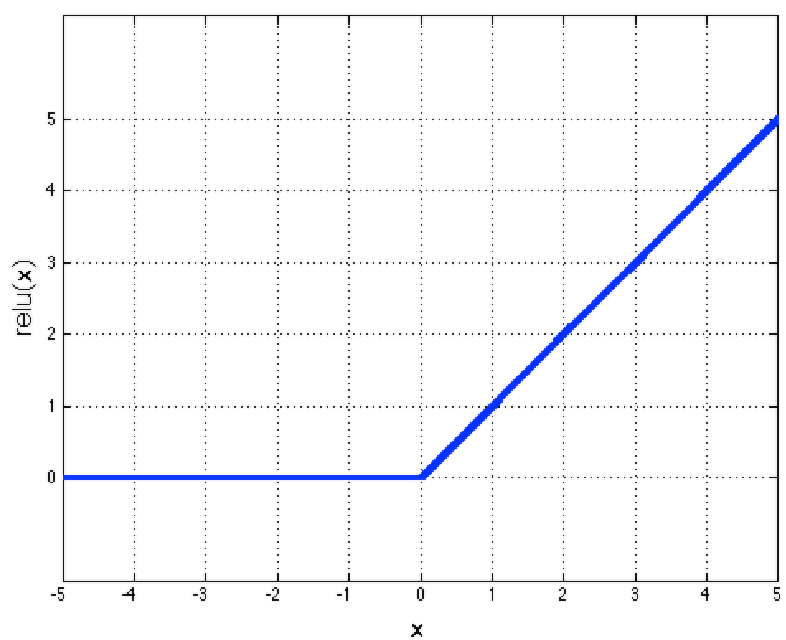
Feature Map

Source: R. Fergus, Y. LeCun

Operations of CNN

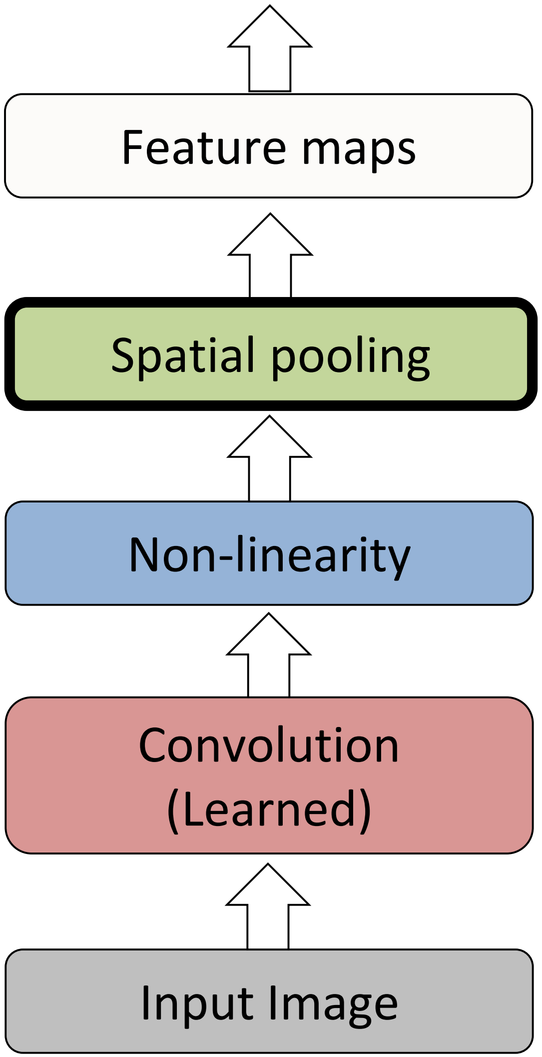


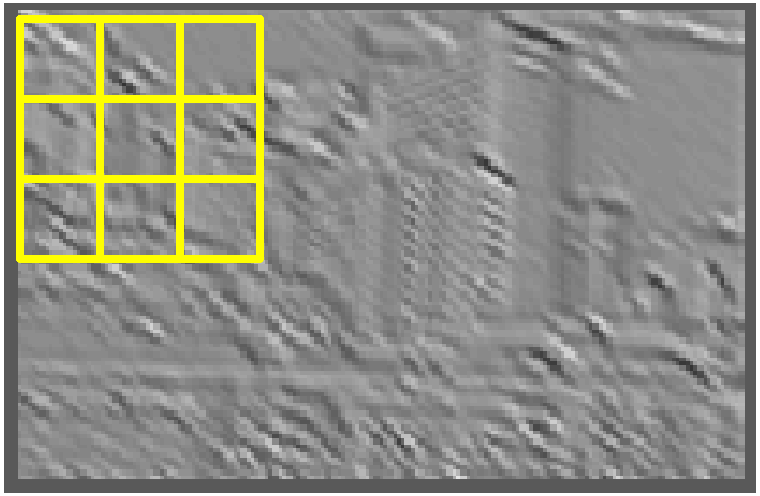
Rectified Linear Unit (ReLU)



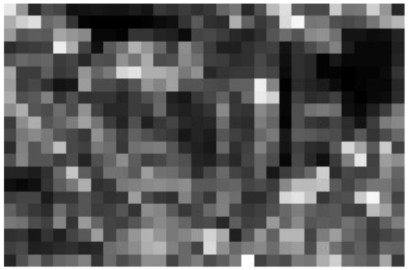
Source: R. Fergus, Y. LeCun

Operations of CNN





Max



Source: R. Fergus, Y. LeCun

CNN vs. DNN

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

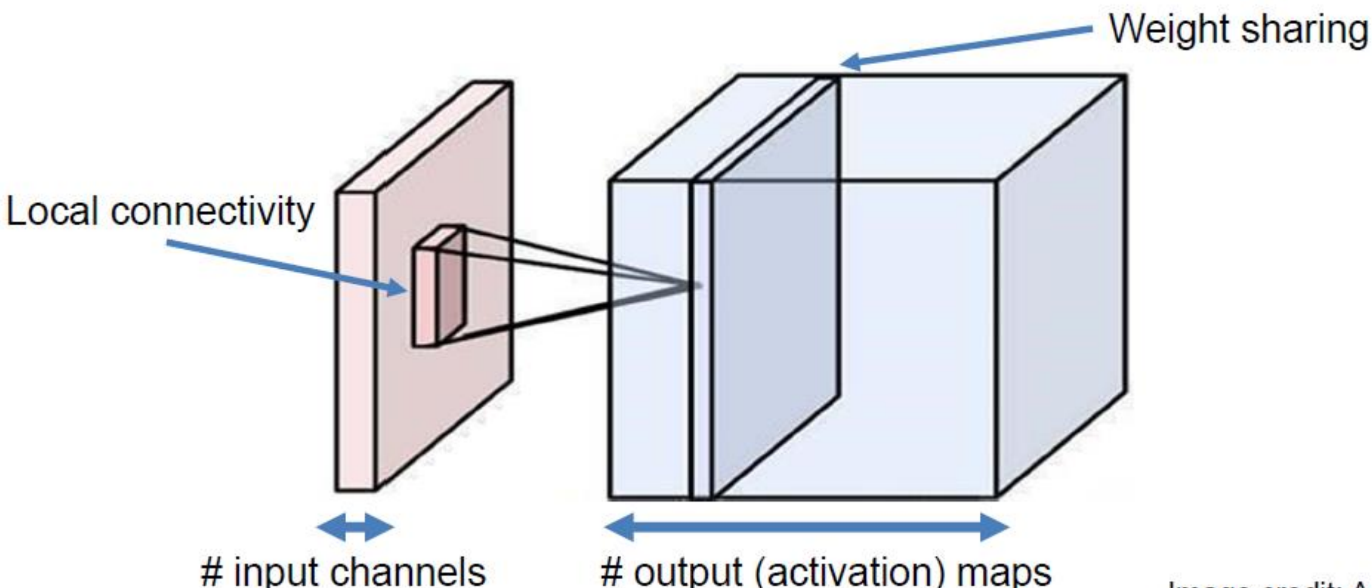
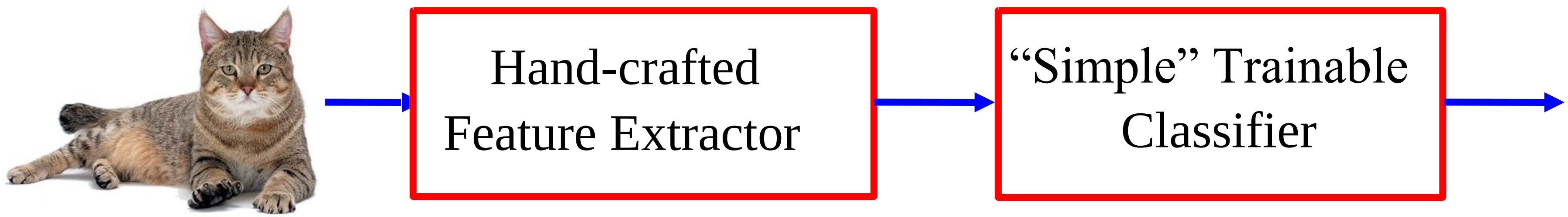


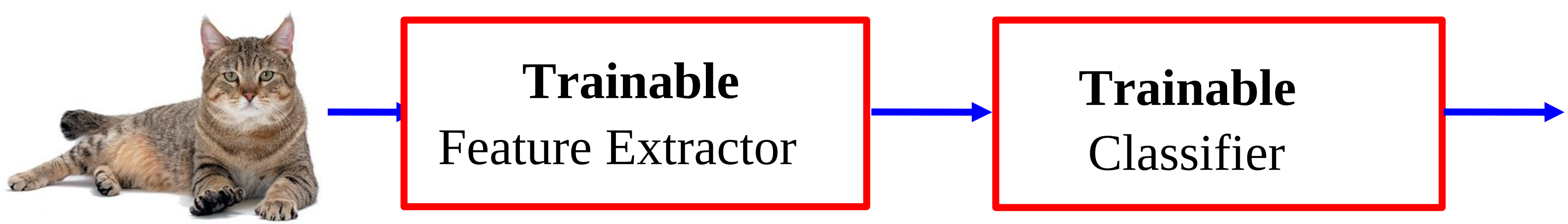
Image credit: A. Karpathy

Conventional ML method vs. deep learning

- Classical ML method

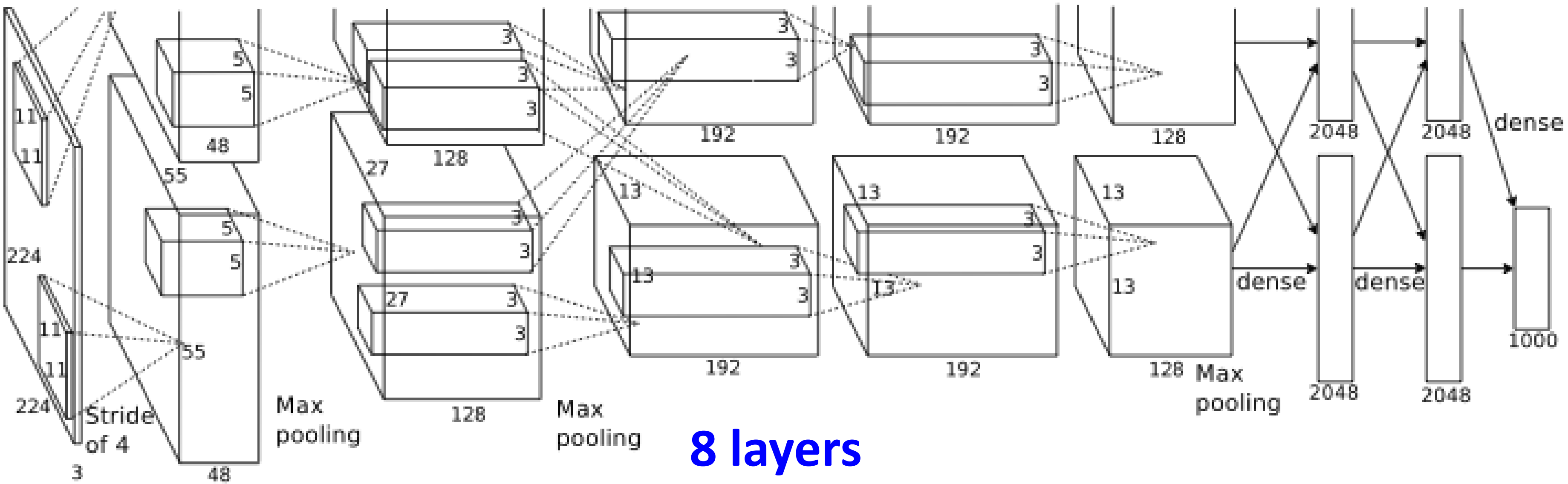


- Deep learning



End-to-End Learning

Modern CNN: AlexNet



8 layers

Input: 224*224*3=**150K**

Neurons: 290400+186624+64896+64896+43264+4096+4096+1000=**650K**

Weights: 11*11*3*48*2(35K)+5*5*48*128*2(307K)+128*3*3*192*4(884K)+ 192*3*3*192*2(663K)+192*3*3*128*2(442K)+6*6*128*2048*4(38M)+4096*4096(17M)+4096*1000(4M)=**60M**

- **More data (1.2M)**
- **Trained on two GPUs for a week**

國立交通大學 National Chiao Tung University

ImageNet ISLVRC 2012-2014: Object Recognition

Best non-convnet in 2012: 26.2%

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

Team	Method	Error (top-5)
DeepImage - Baidu	Data augmentation + multi GPU	5.33%
PReLU-nets - MSRA	Parametric ReLU + smart initialization	4.94%
BN-Inception ensemble - Google	Reducing internal covariate shift	4.82%

國立交通大學
National Chiao Tung University

Thank You for Your Attention!

Yen-Yu Lin (林彥宇)

Email: lin@cs.nctu.edu.tw

URL: https://www.cs.nctu.edu.tw/members/detail/lin