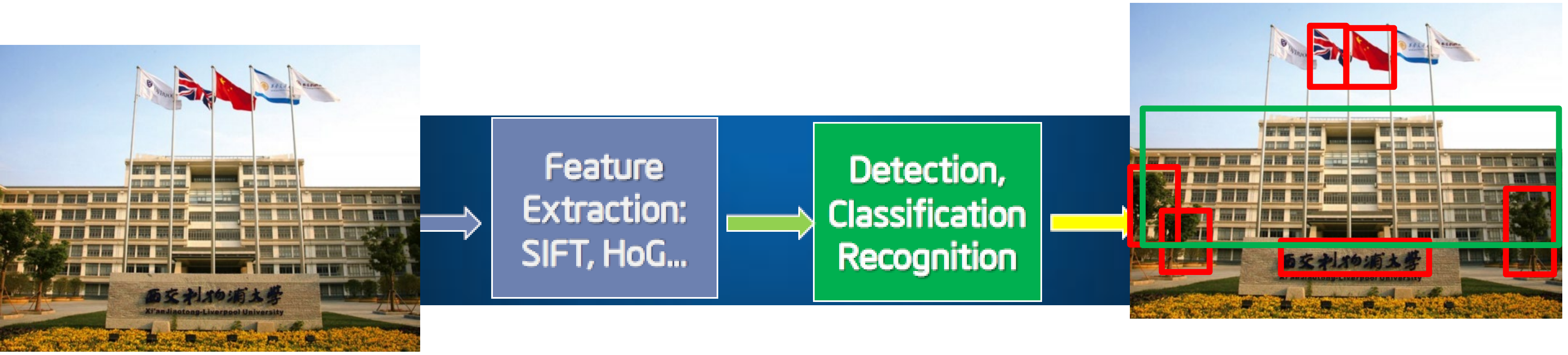
INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORK

INT301 Bio-computation, Week 5, 2025

Classical Computer Vision Pipeline

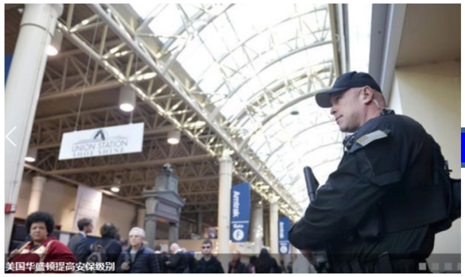
1.Select / develop features (e.g., HoG, SIFT, …)
2.Add machine learning on top of this for recognition and train classifier


Feature Extraction: SIFT, HoG…
Detection, Classification Recognition
西交利物浦大学
Xi'an Jiaotong-Liverpool University

Classical CV feature definition is domain-specific, hand engineered, and time-consuming

Features for machine learning

image



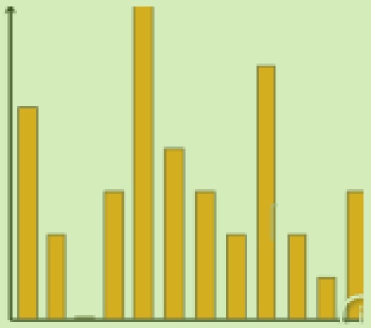
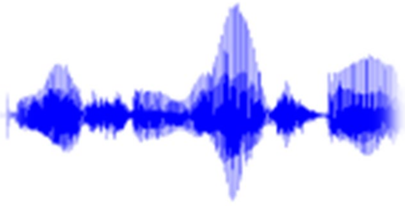
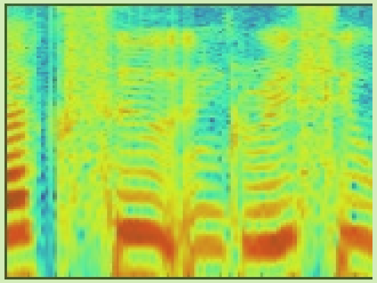


Image feature

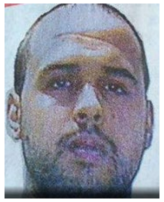
Object detection
/classification

audio



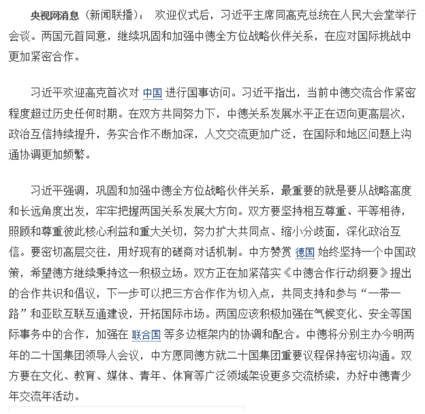


Audio feature



Speaker ID

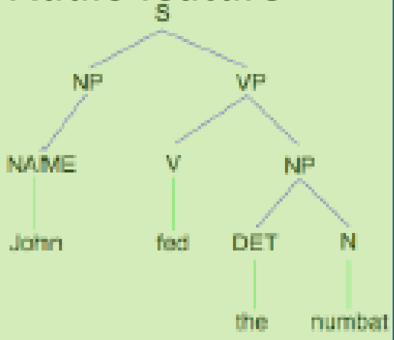
text



央视网消息（新闻联播）： 欢迎仪式后，习近平主席同高克总统在人民大会堂举行会谈。两国元首同意，继续巩固和加强中德全方位战略伙伴关系，在应对国际挑战中更加紧密合作。

习近平欢迎高克首次对 中国 进行国事访问。习近平指出，当前中德交流合作紧密程度超过历史任何时期。在双方共同努力下，中德关系发展水平正在迈向更高层次，政治互信持续提升，务实合作不断加深，人文交流更加广泛，在国际和地区问题上沟通协调更加频繁。

习近平强调，巩固和加强中德全方位战略伙伴关系，最重要的就是要从战略高度和长远角度出发，牢牢把握两国关系发展大方向。双方要坚持相互尊重、平等相待，照顾和尊重彼此核心利益和重大关切，努力扩大共同点、缩小分歧面，深化政治互信。要密切高层交往，用好现有的磋商对话机制。中方赞赏 德国 始终坚持一个中国政策，希望德方继续秉持这一积极立场。双方正在加紧落实《中德合作行动纲要》提出的合作共识和倡议，下一步可以把三方合作作为切入点，共同支持和参与“一带一路”和亚欧互联互通建设，开拓国际市场。两国应该积极加强在气候变化、安全等国际事务中的合作，加强在 联合国 等多边框架内的协调和配合。中德将分别主办今明两年的二十国集团领导人会议，中方愿同德方就二十国集团重要议程保持密切沟通。双方要在文化、教育、媒体、青年、体育等广泛领域架设更多交流桥梁，办好中德青少年交流年活动。



Text feature

Translation/
/categorization
/Web search

Key Ideas of Deep Learning

- Deal with non-linear system
- Learn feature from data (or big data)
- Build feature hierarchies (function composition)
- End-to-end learning

Learning Feature Representations

The idea:

- Most perception (input processing) in the brain may be due to **one learning algorithm**.
- Build learning algorithms that mimic the brain.
- Most of human intelligence may be due to **one learning algorithm**.

one learning algorithm $\Rightarrow$ end-to-end system!

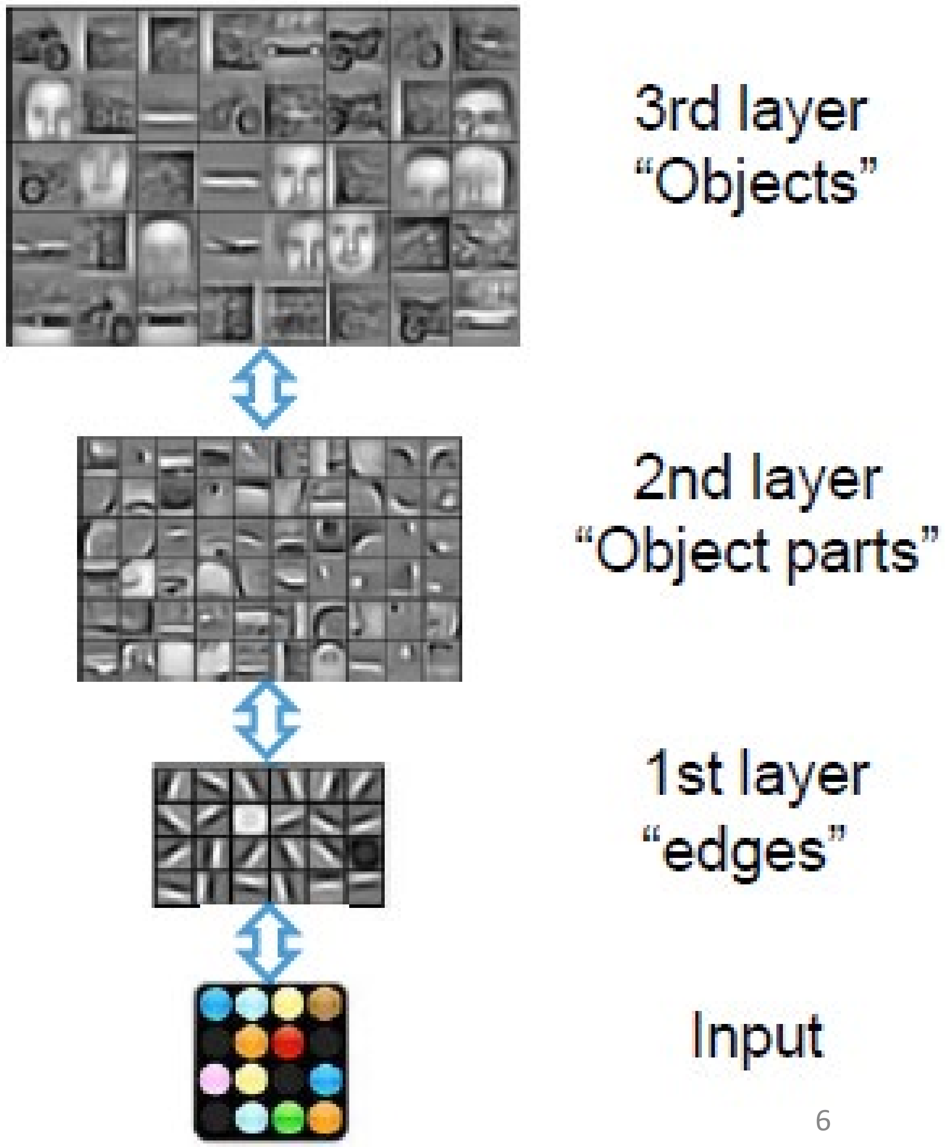
Learning Feature Hierarchy

Deep Learning

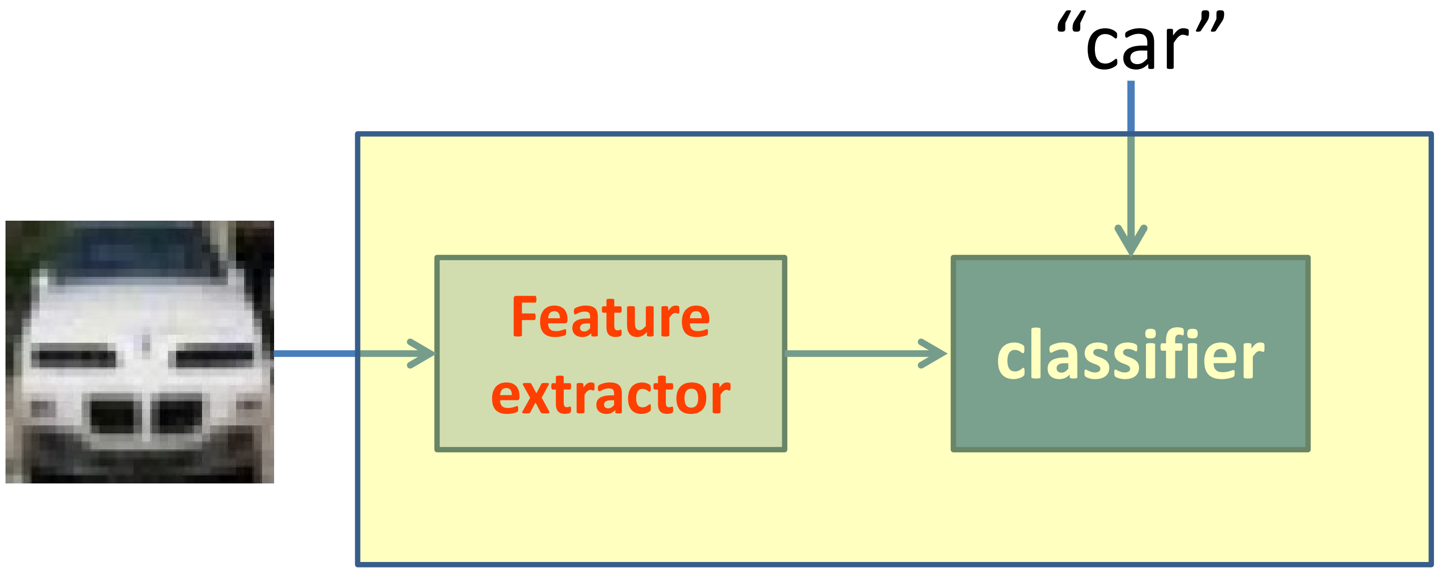
– Deep architectures can be representationally efficient.

– Natural progression from low level to high level structures.

– Can share the lower-level representations for multiple tasks.



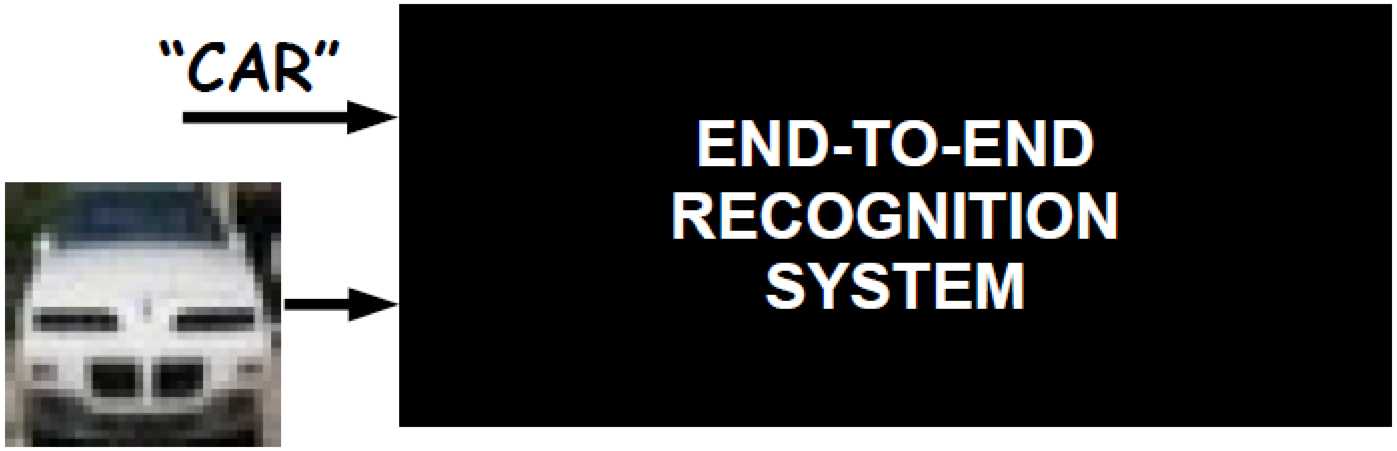
End-to-end Object Recognition



How to use data to optimize features for the given task?

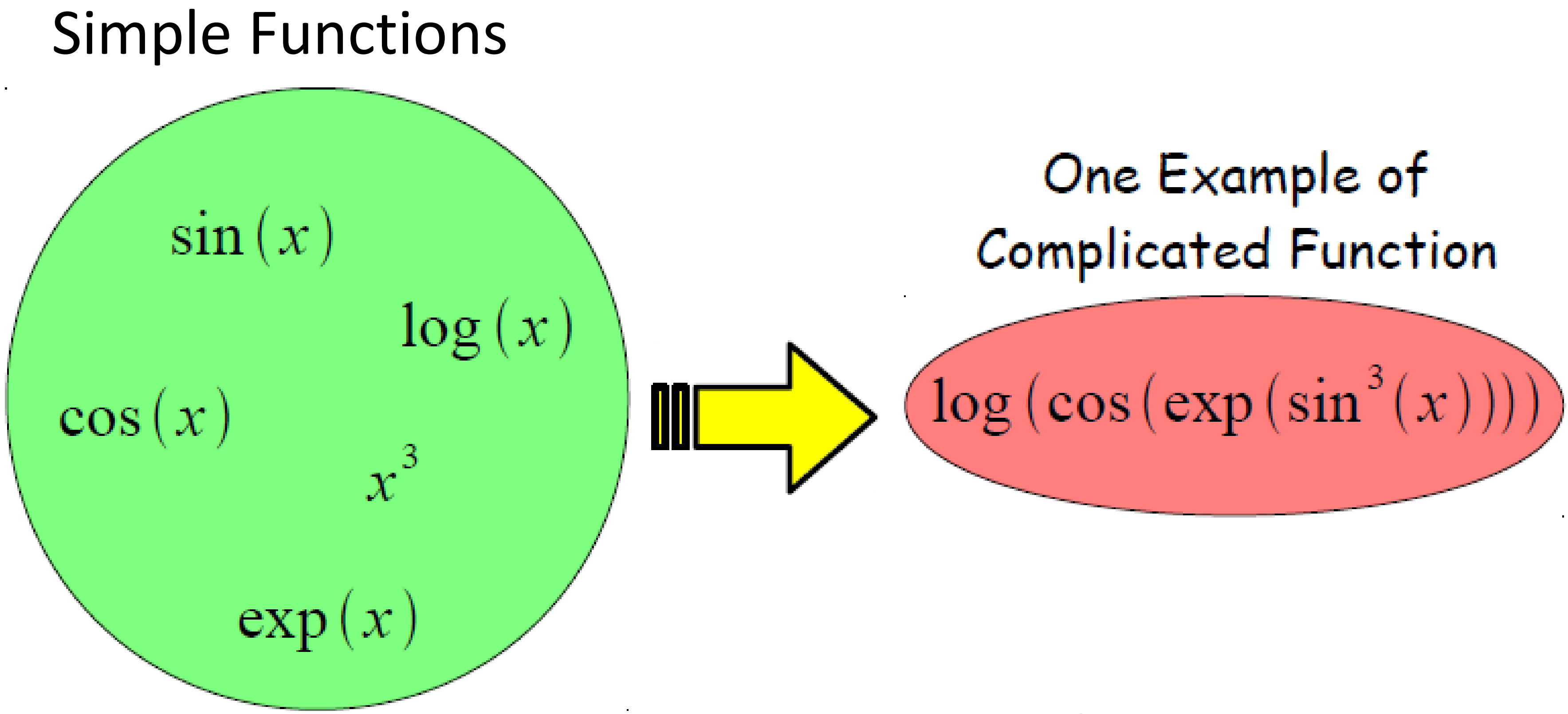
- Everything becomes adaptive.
- No distinction between feature extractor and classifier.
- Big non-linear system trained from raw pixels to labels

End-to-end Object Recognition



Q: How can we build such a highly non-linear system?
A: By combining simple building blocks, we can make more and more complex systems.

Building A Complicated Function

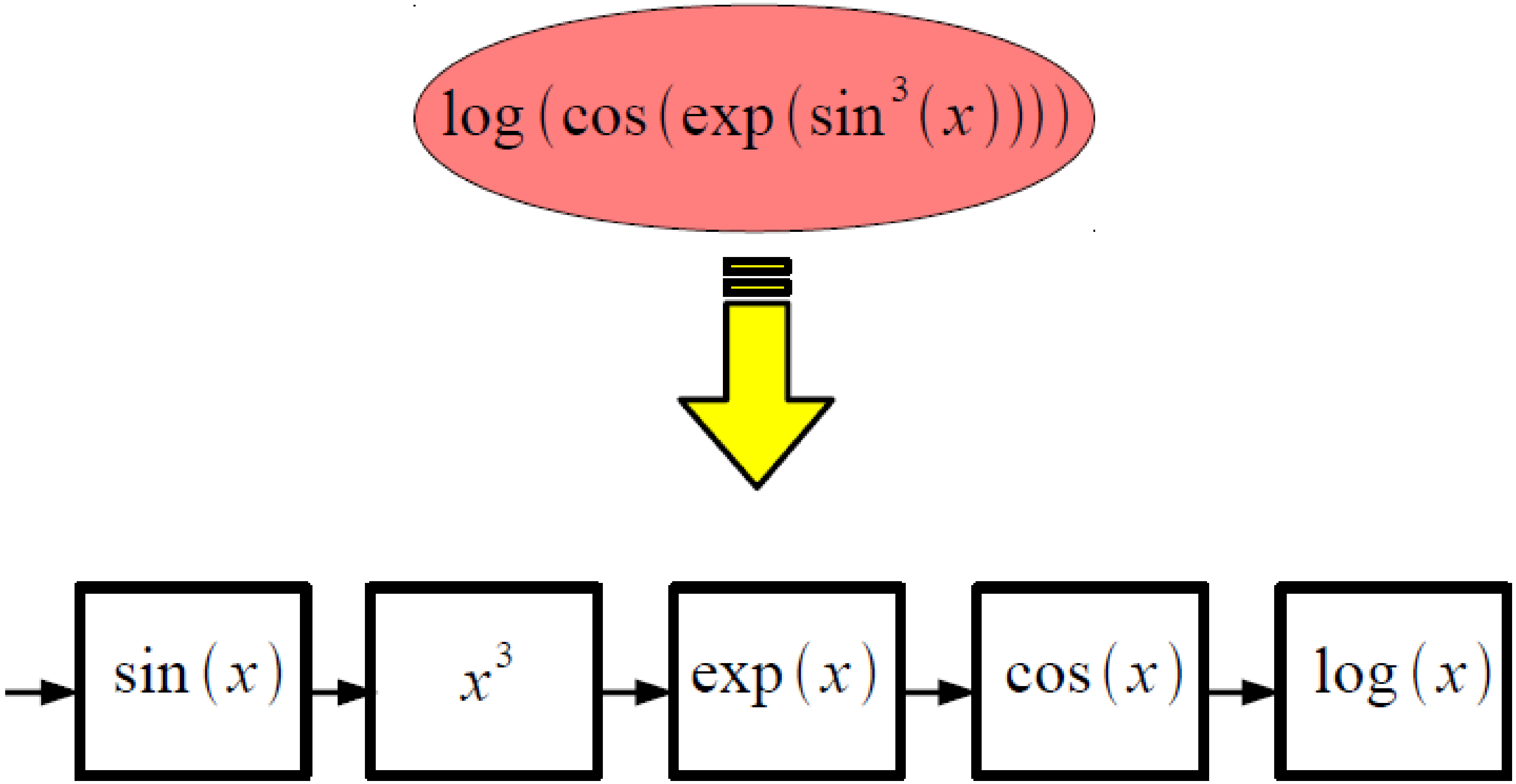


Simple Functions
$\sin(x)$
$\log(x)$
$\cos(x)$
x^3
$\exp(x)$
One Example of Complicated Function
$\log(\cos(\exp(\sin^3(x))))$

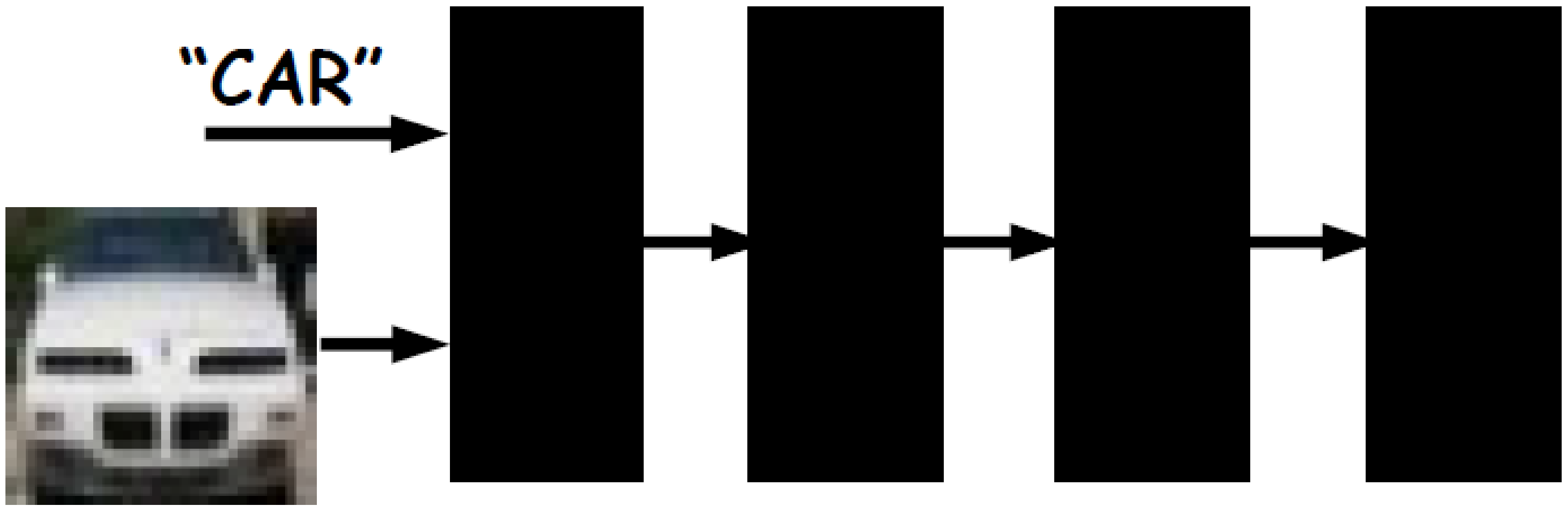
- Function composition is at the core of deep learning methods.
- Each "simple function" will have parameters subject to training.

Building A Complicated Function

Complicated Function

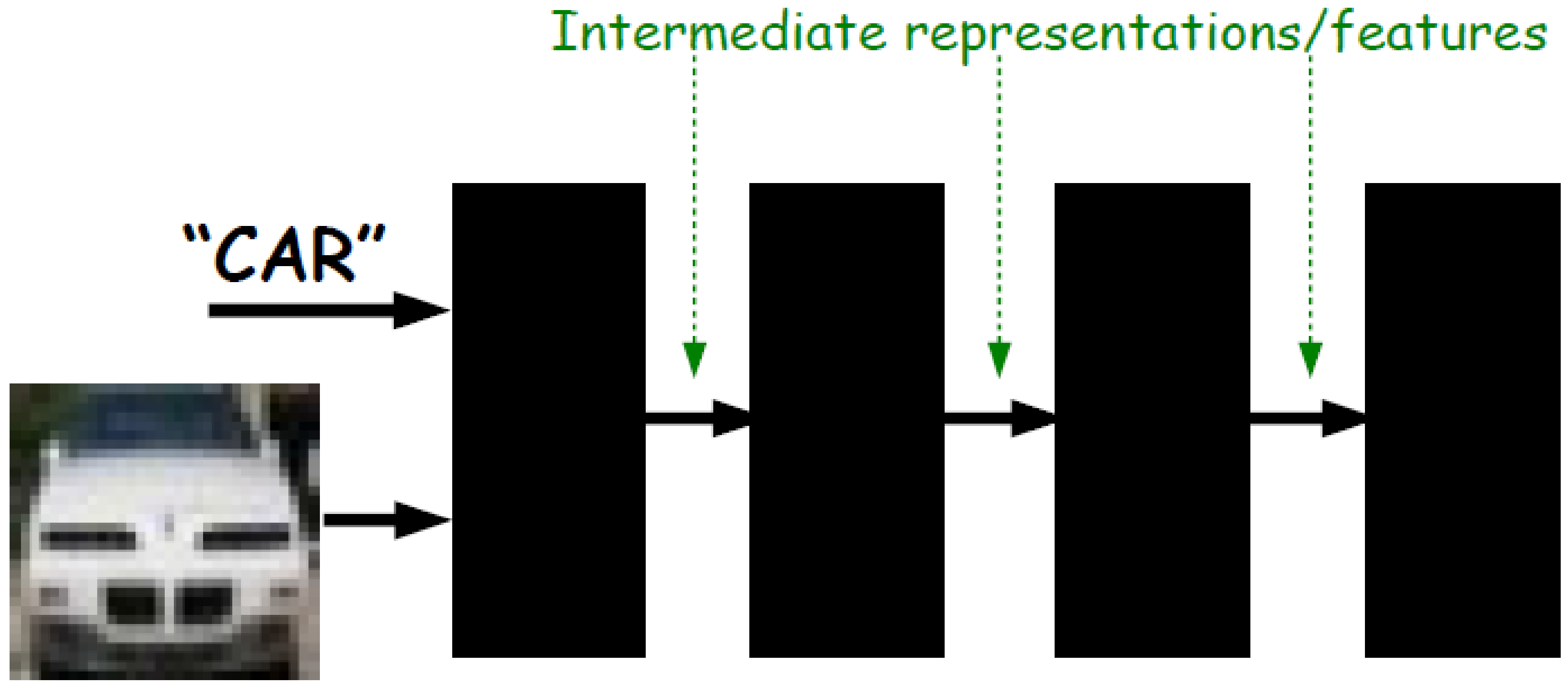


Intuition Behind Deep Neural Nets



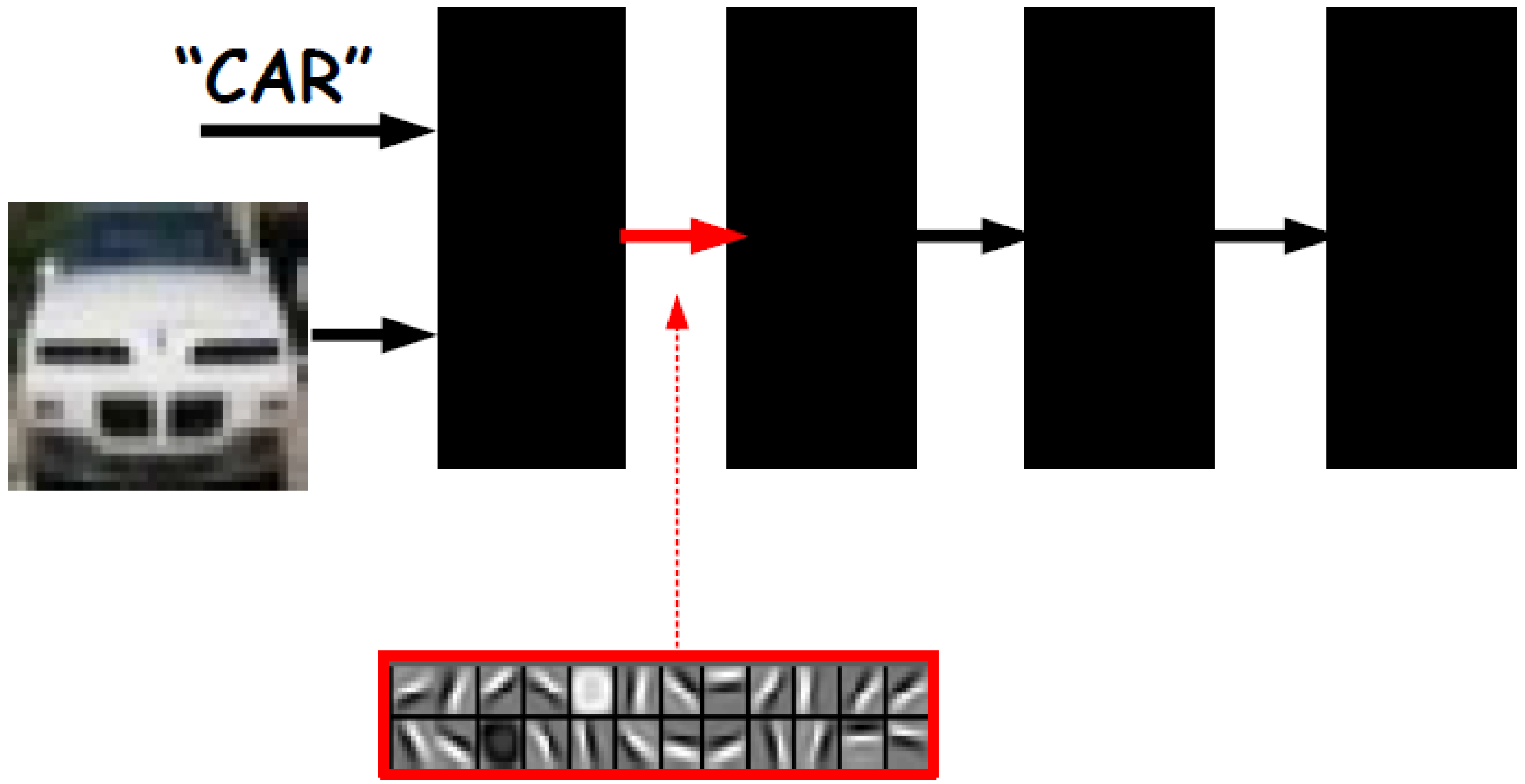
NOTE:
Each black box can have trainable parameters.
Their composition makes a highly non-linear system.

Intuition Behind Deep Neural Nets

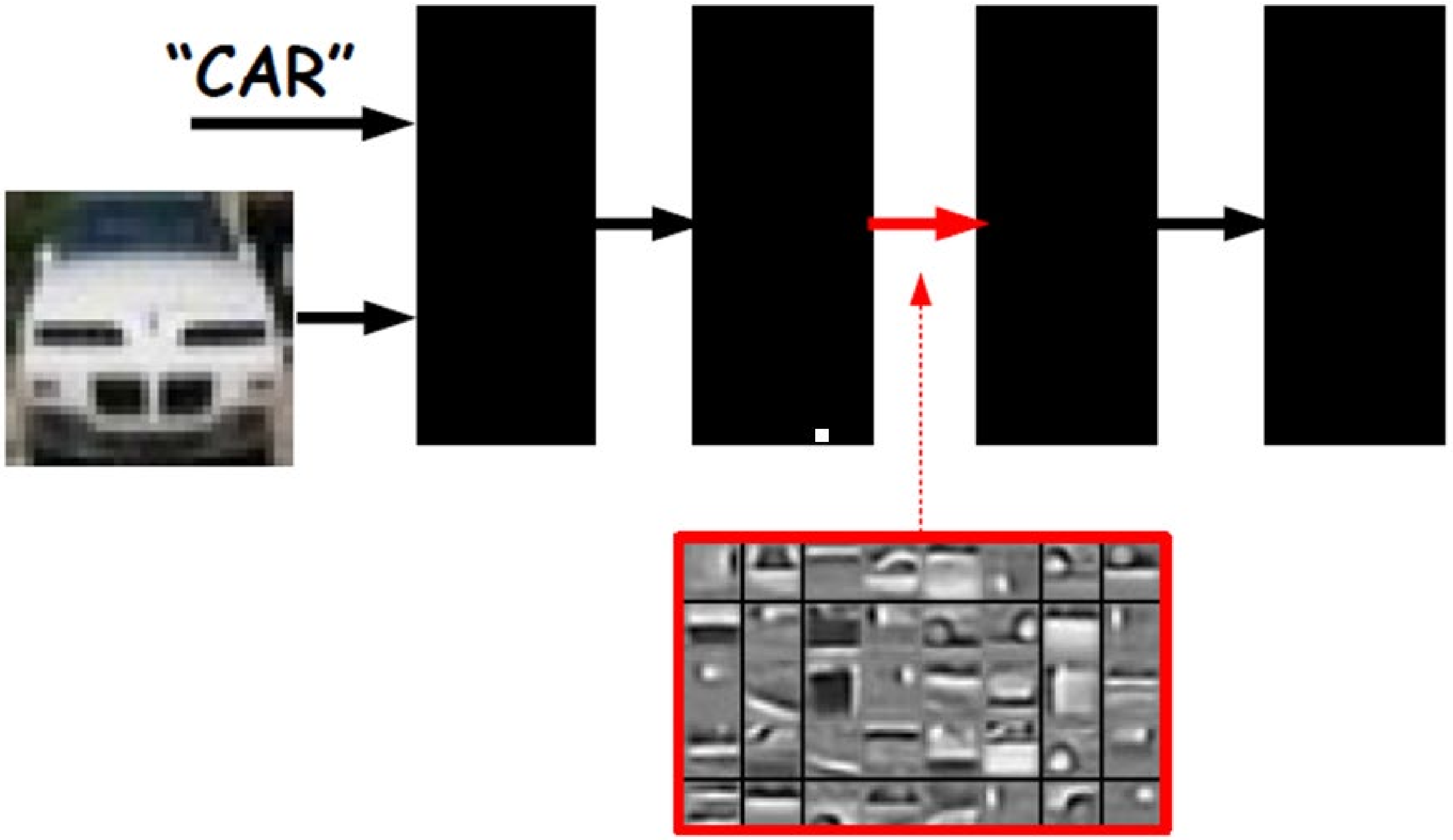


NOTE: System produces a hierarchy of features

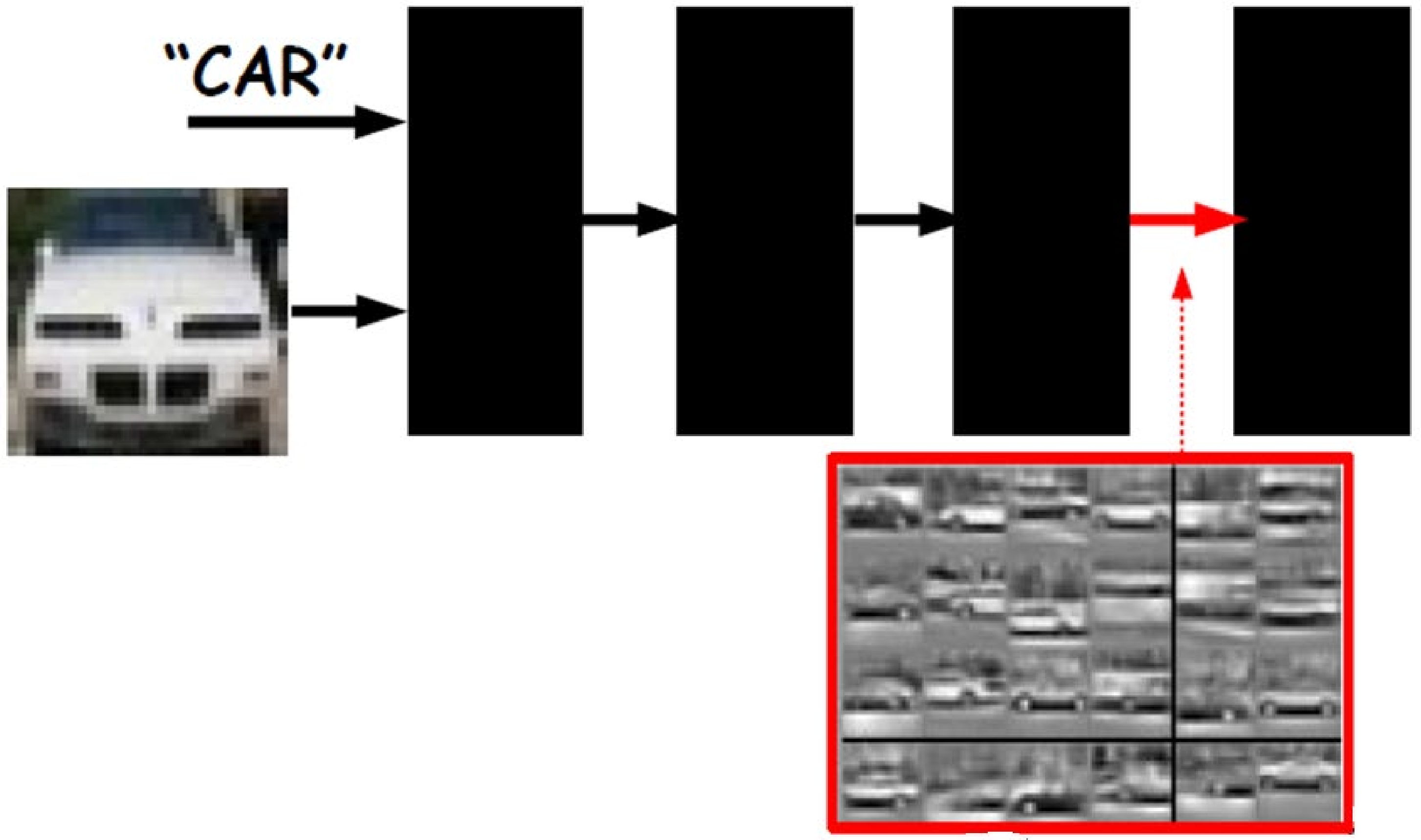
Intuition Behind Deep Neural Nets



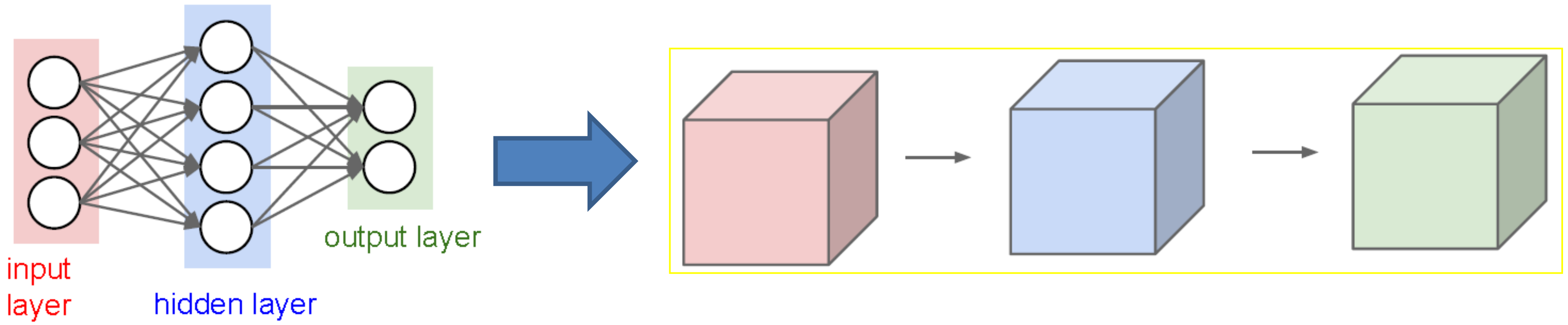
Intuition Behind Deep Neural Nets



Intuition Behind Deep Neural Nets



Convolutional NN



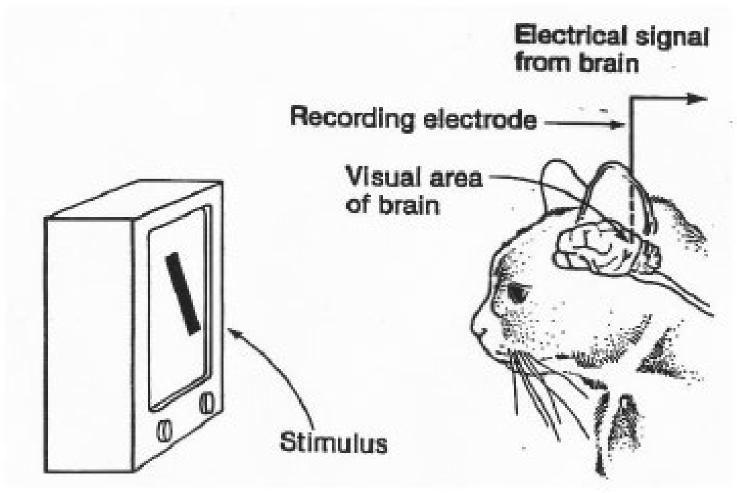
- Convolutional Neural Networks is extension of traditional Multi-layer Perceptron, based on 3 ideas:
  1. **Local receive fields**
  2. **Shared weights**
  3. **Spatial / temporal sub-sampling**

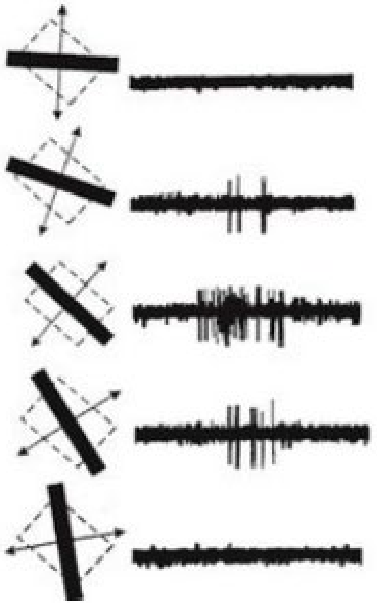
A bit of history

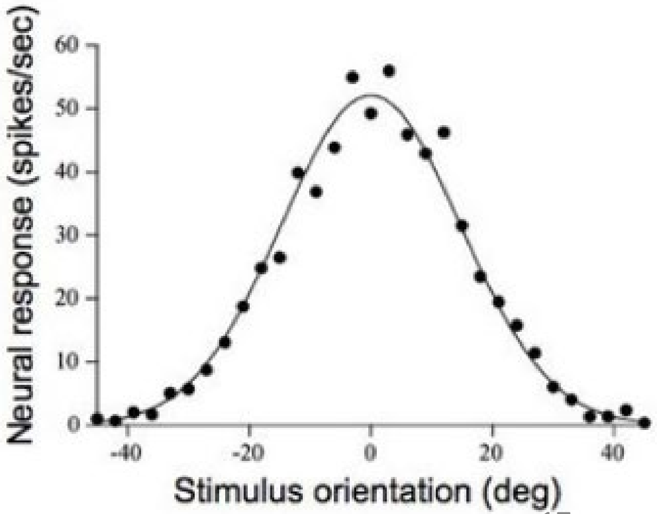
Hubel & Wiesel

1959 Paper
Receptive fields of single neurones in the cat's striate cortex

1962 Paper
Receptive fields, binocular interaction and functional architecture in cat's visual cortex



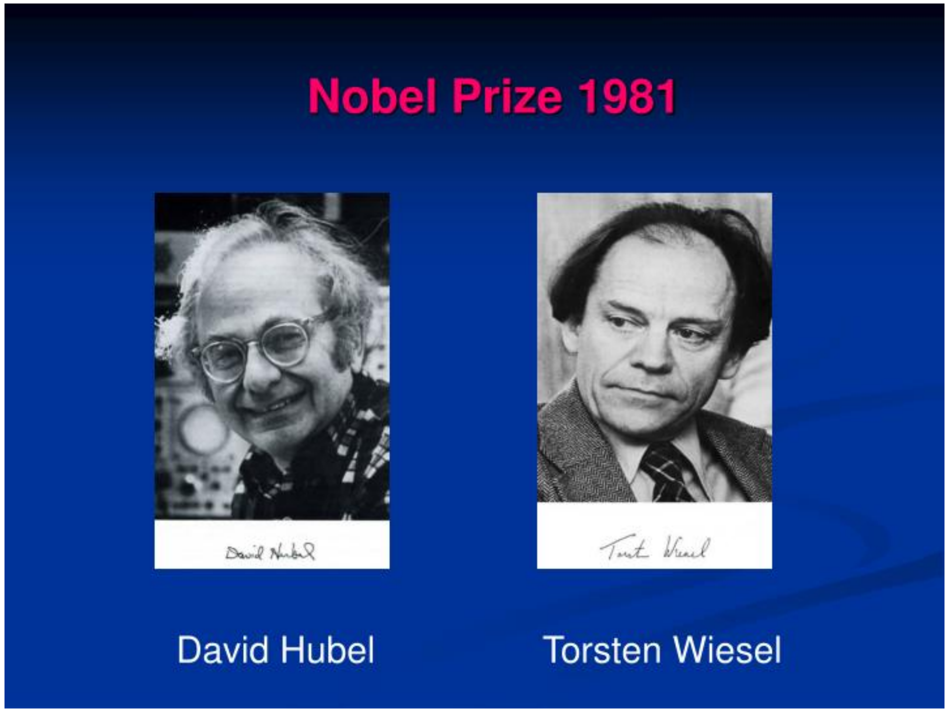


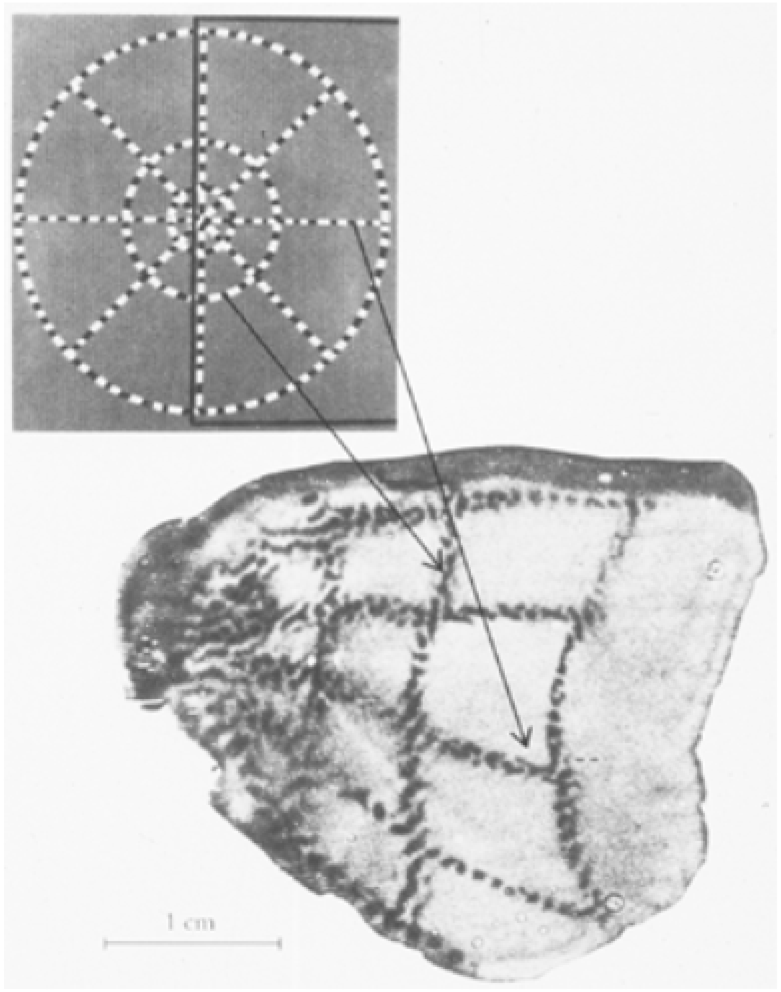


A bit of history

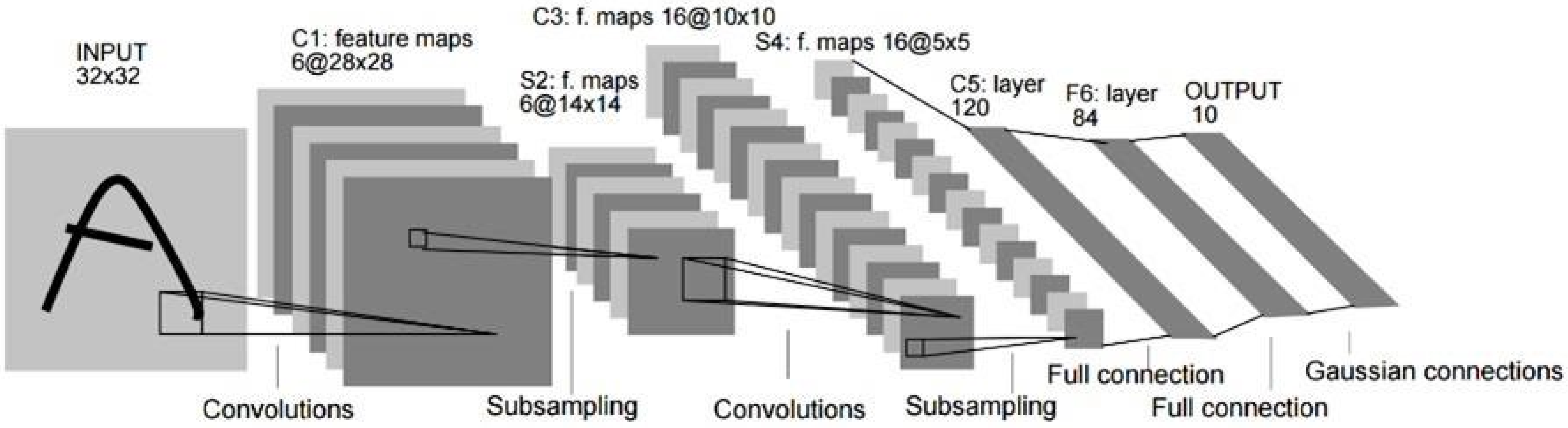
Topographical mapping in the cortex:

nearby cells in cortex represented nearby regions in the visual field





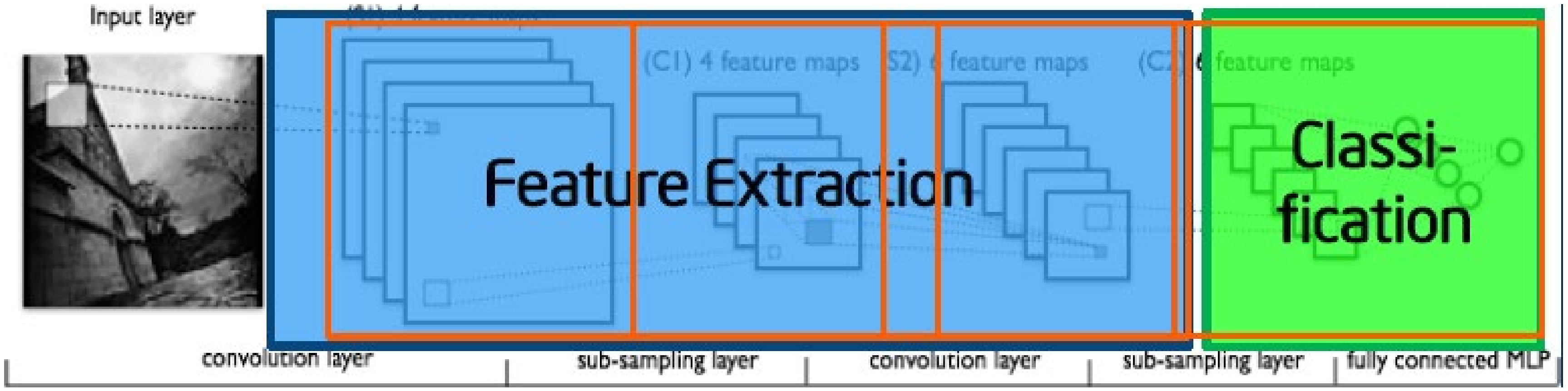
LeNet [LeCun et al. 1998]



- See LeCun paper (1998) on text recognition:

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

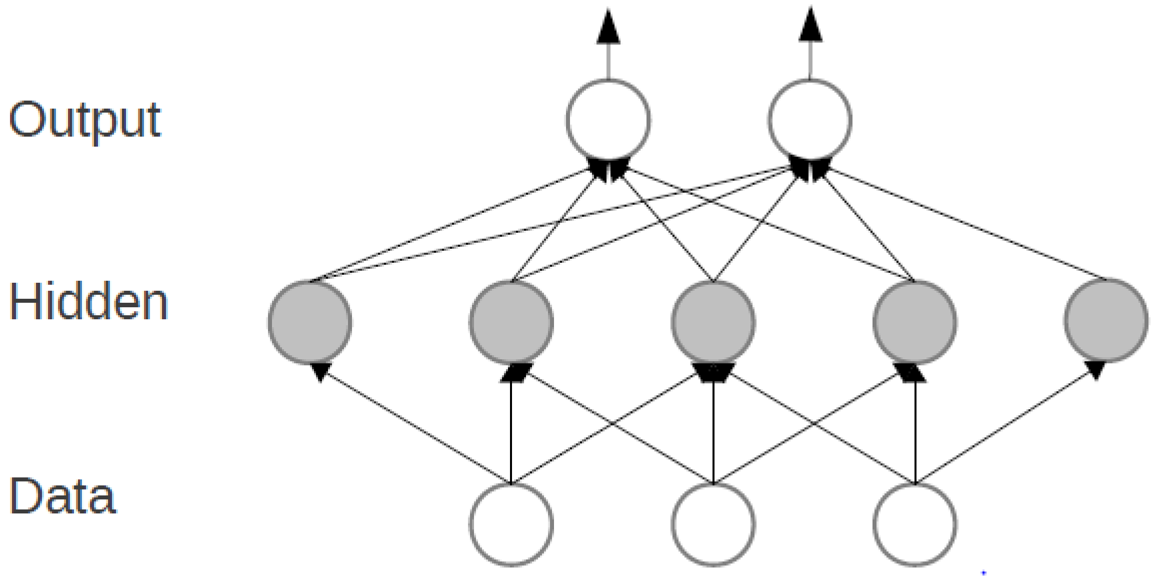
Convolutional NN (CNN)



- Convolutional layer
- Sub-sampling layer
- Fully connected layers

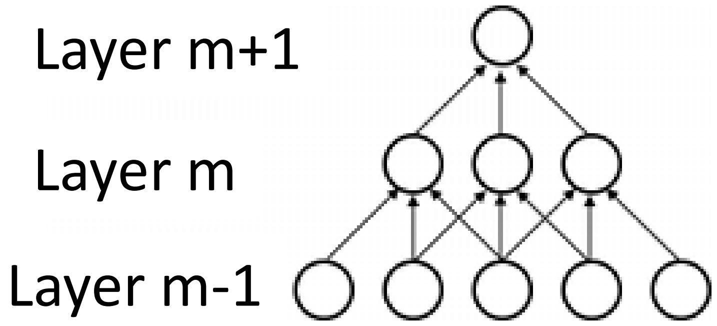
Basic Concept of CNN

- Here's a one-dimensional convolutional neural network
- Each hidden neuron applies ***the same localized, linear filter*** to the input



Sparse Connectivity

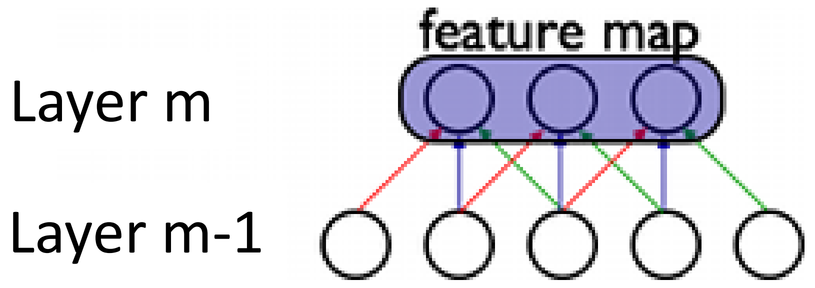
- CNNs exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers, i.e., the inputs of hidden units in layer **m** are from a subset of units in layer **m-1**, units that have spatially contiguous *receptive fields*



Imagine that layer **m-1** is the input retina. In the above figure, units in layer **m** have receptive fields of width 3 in the input retina and are thus only connected to 3 adjacent neurons in the retina layer. Units in layer **m+1** have a similar connectivity with the layer below. We say that their receptive field with respect to the layer below is also 3, but their receptive field with respect to the input is larger (5). Each unit is unresponsive to variations outside of its receptive field with respect to the retina. The architecture thus ensures that **the learnt "filters" produce the strongest response to a spatially local input pattern.**

Shared Weights

- In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a *feature map*



In the above figure, we show 3 hidden units belonging to the same feature map. Weights of the same color are shared—constrained to be identical. Gradient descent can still be used to learn such shared parameters, with only a small change to the original algorithm.

Replicating units in this way allows for features to be detected *regardless of their position in the visual field.* Additionally, weight sharing increases learning efficiency by greatly reducing the number of free parameters being learnt. The constraints on the model enable CNNs to achieve better generalization on vision problems.

Details and Notation

- A feature map is obtained by repeated application of a function across sub-regions of the entire image, in other words, by *convolution* of the input image with a linear filter, adding a bias term and then applying a non-linear function.
- If we denote the k-th feature map at a given layer as h^k, whose filters are determined by the weights W^k and bias b_k, then the feature map is obtained using convolution as follows (for *tanh* non-linearities):

$$\boxed{h^k_{ij} = tanh((W^k * \mathrm{x})_{ij} + b_k)}$$

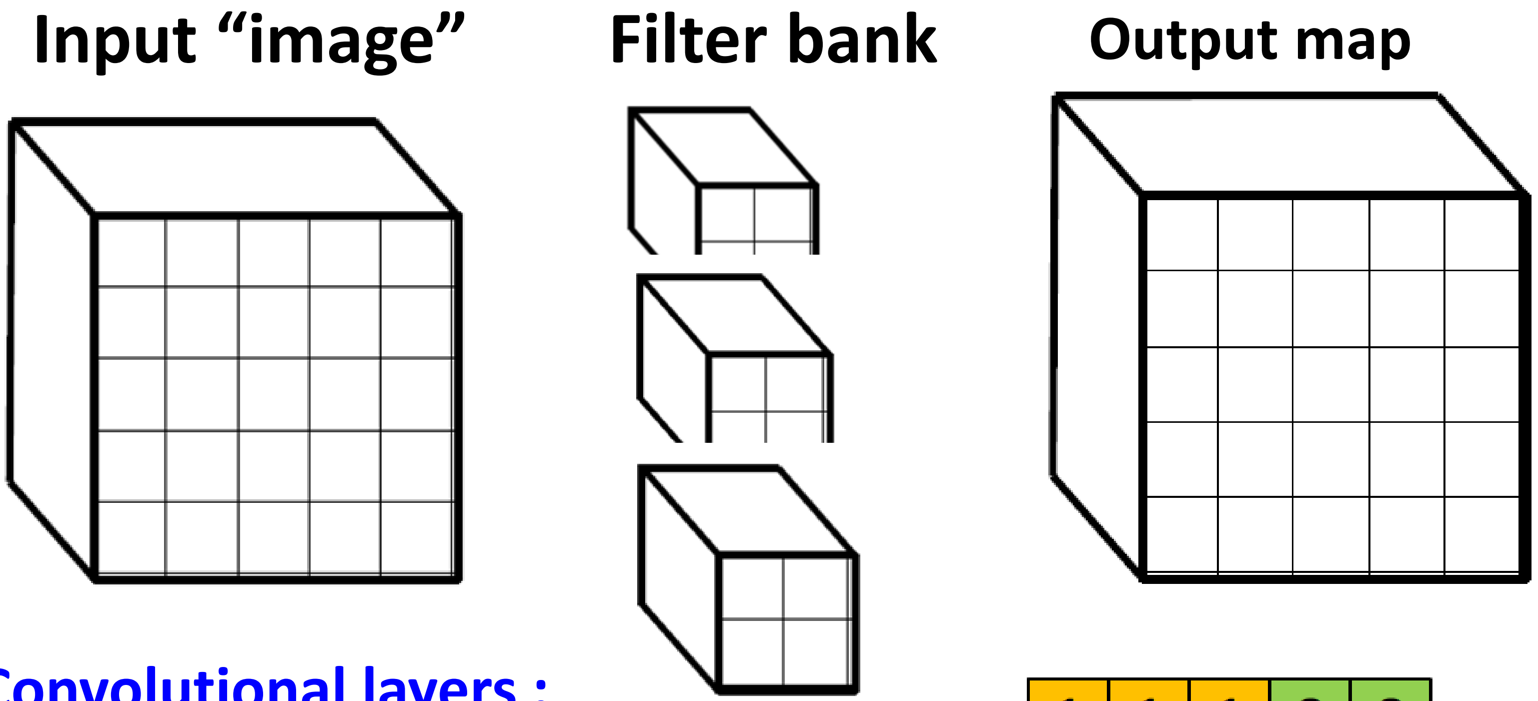
Convolutional Layers

- Suppose that we have some NxN square neuron layer which is followed by our convolutional layer. If we use an mxm filter ω, our convolutional layer output will be of size (N−m+1)x(N−m+1).

- In order to compute the pre-nonlinearity input to some unit x_{ij}^{l} in our layer, we need to sum up the contributions (weighted by the filter components) from the previous layer cells:

$$x_{ij}^{\ell} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \omega_{ab}\, y_{(i+a)(j+b)}^{\ell-1}.$$

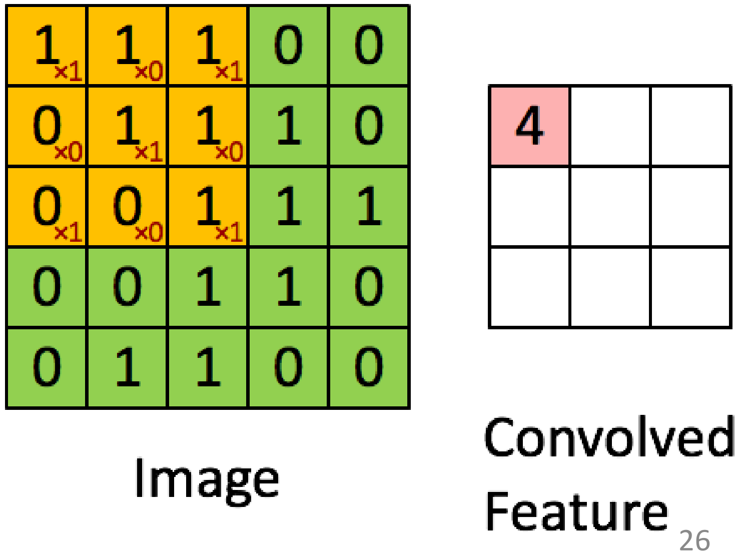
Convolution in 2D



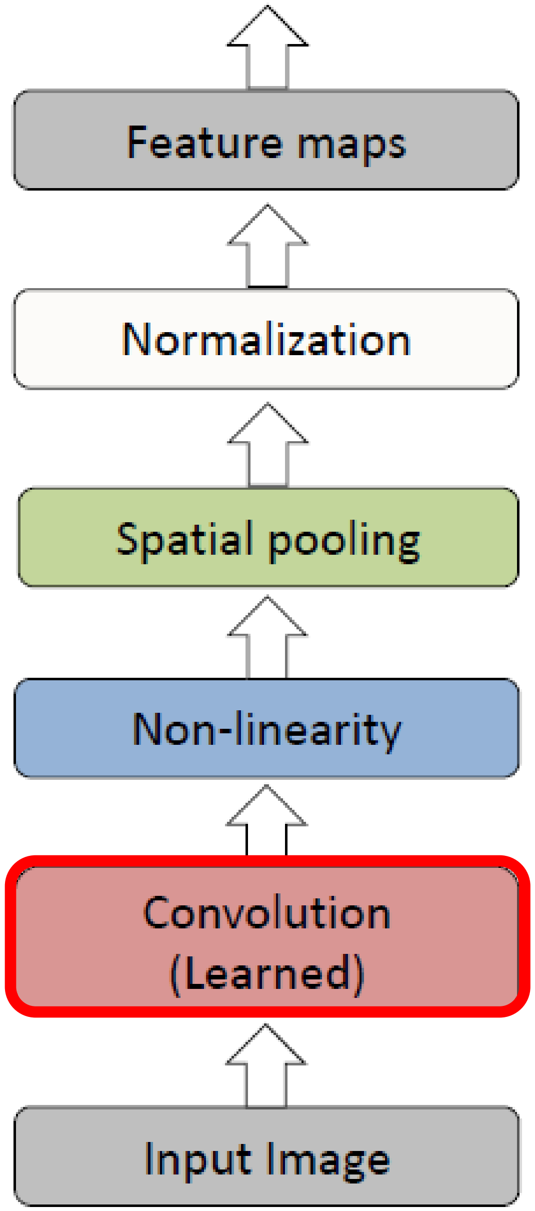
Convolutional layers :

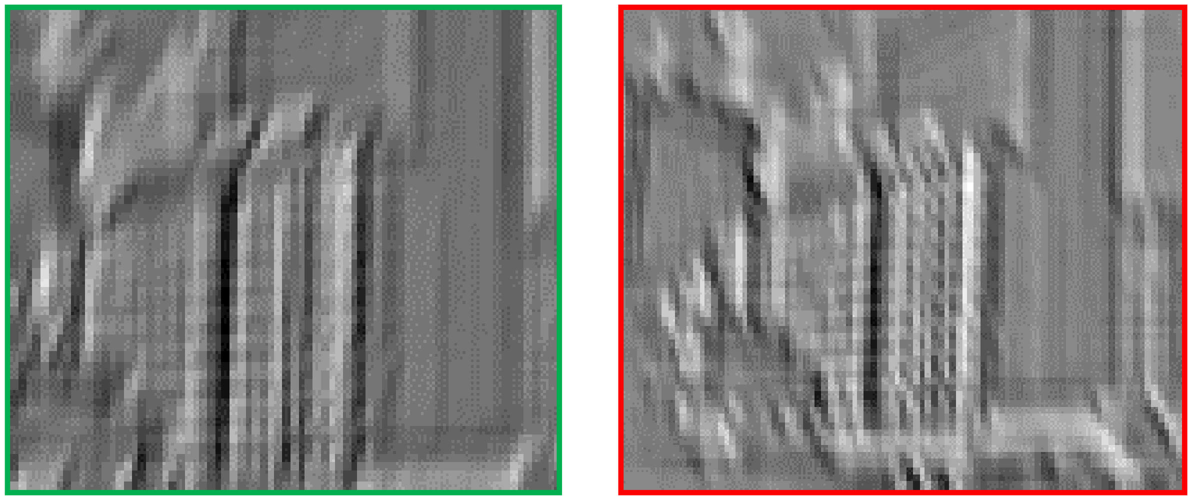
a rectangular grid of neurons.
The previous layer is also a rectangular grid of neurons.

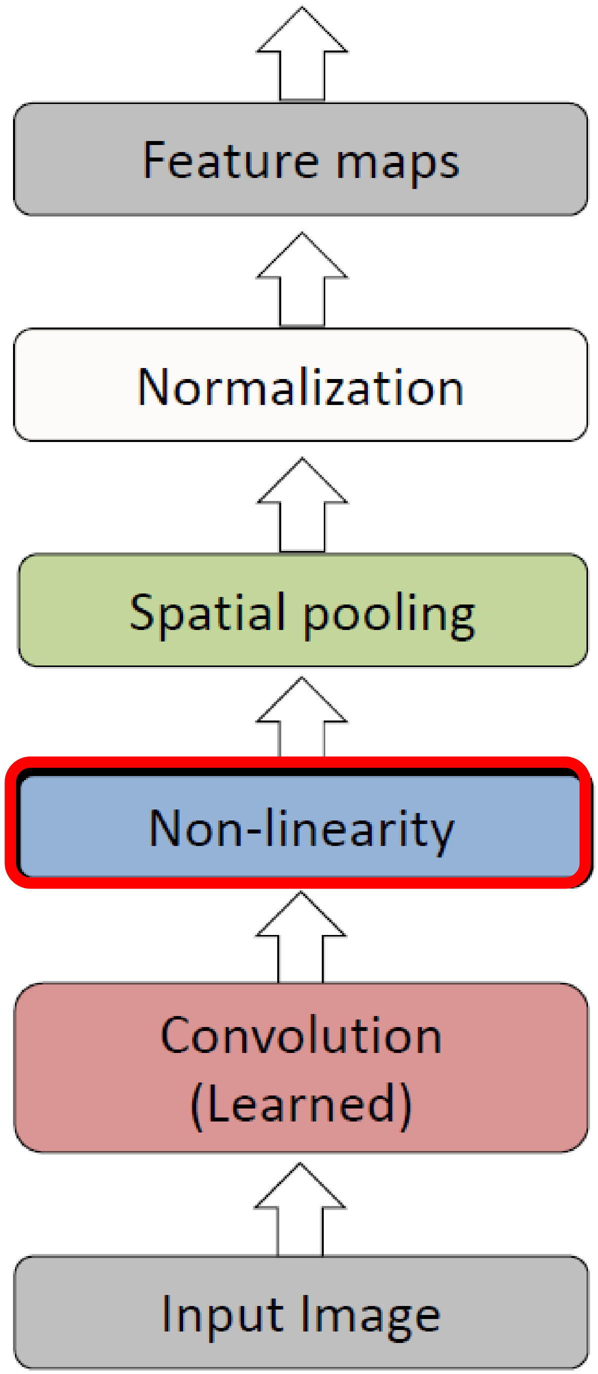
Each neuron takes inputs from a rectangular section of the previous layer; the weights for this rectangular section are the same for each neuron in the convolutional layer



Convolutional Neural Networks



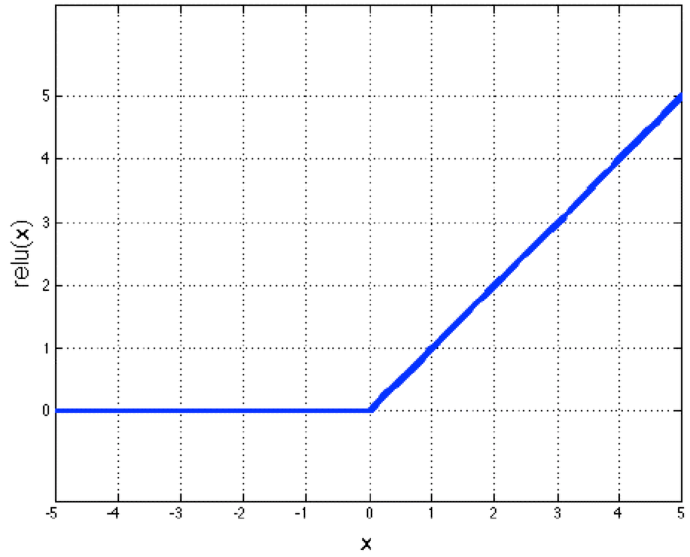




Non-saturating function

$$f(x) = \max(0, x)$$

Rectified Linear Unit (ReLU)



Benefits of using ReLU

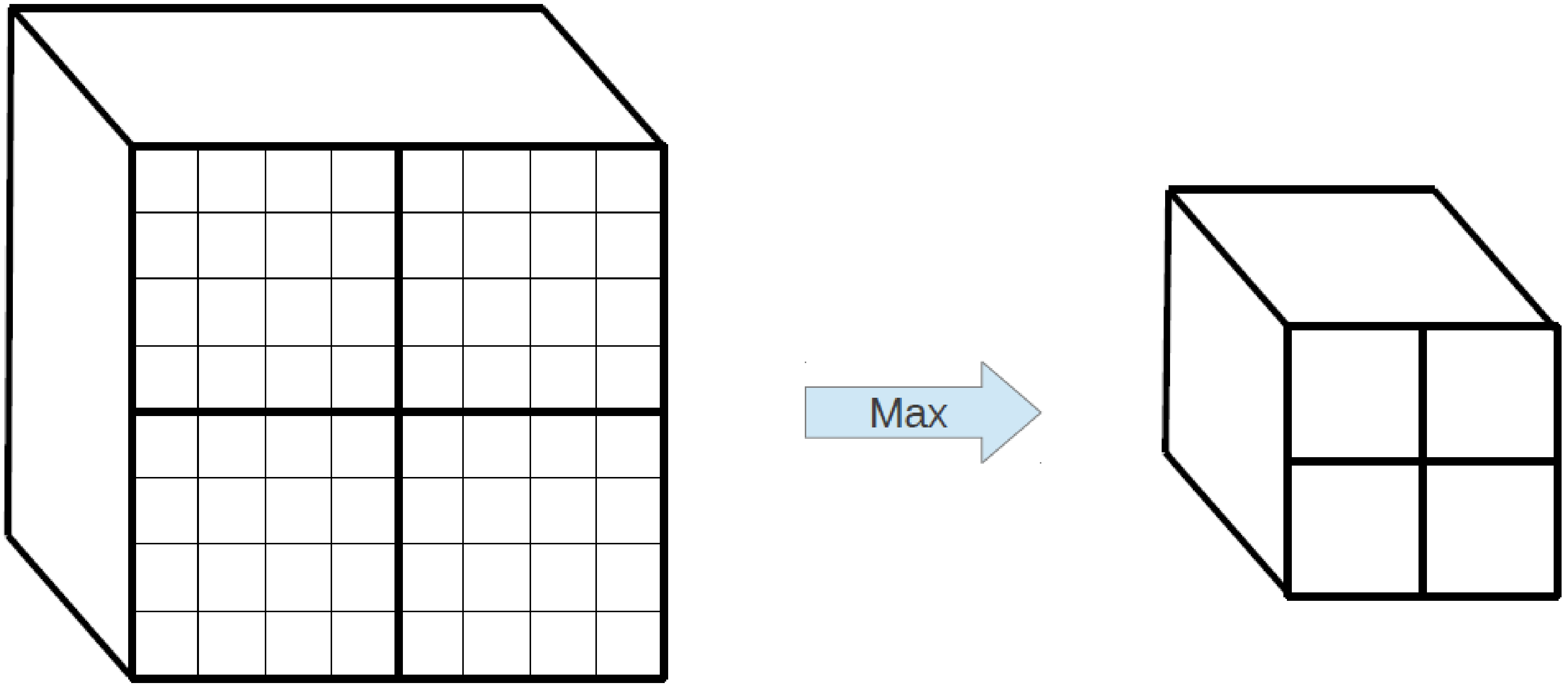
- ReLUs are much simpler computationally
  - The forward and backward passes through an ReLU are both just a simple *if* statement
  - The sigmoid activation requires computing an exponent
  - This advantage is huge when dealing with big networks with many neurons, and can significantly reduce both training and evaluation times

Benefits of using ReLU

- Sigmoid activations are easier to saturate
  - There is a comparatively narrow interval of inputs for which the sigmoid's derivative is *sufficiently nonzero*
  - In other words, once a sigmoid reaches either the left or right plateau, it is almost meaningless to make a backward pass through it, since the derivative is very close to 0
- ReLUs only saturate when the input is less than 0
  - Even this saturation can be eliminated using leaky ReLUs
- For very deep networks, saturation hampers learning, and so ReLUs provide a nice workaround

Local pooling operation

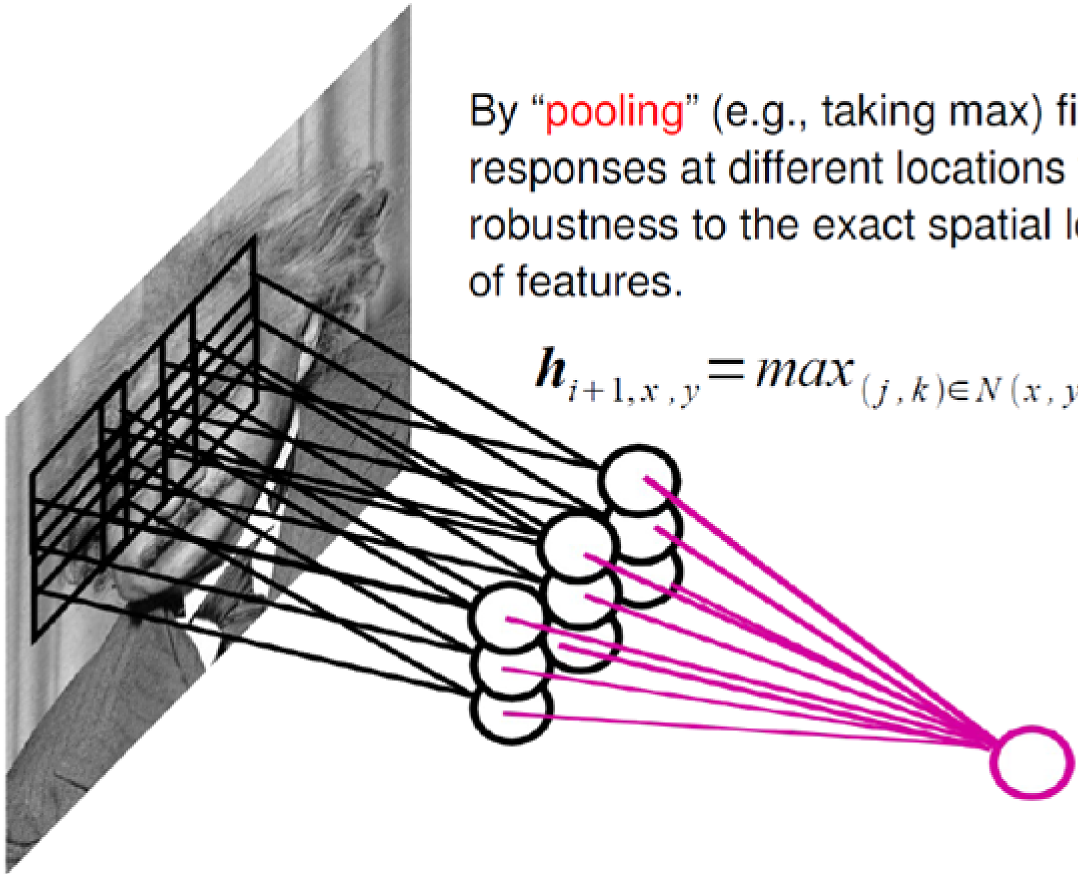
In order to reduce variance, pooling layers compute the max or average value of a particular feature over a region of the image. This will ensure that the same result will be obtained, even when image features have small translations



More on Pooling operation

Subsampling (pooling) Mechanism

- The exact positions of the extracted features are not important
- Only relative position of a feature to another feature is relevant
- Reduce spatial resolution – Reduce sensitivity to shift and distortion



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

$$h_{i+1,x,y} = max_{(j,k)\in N(x,y)} h_{i,j,k}$$

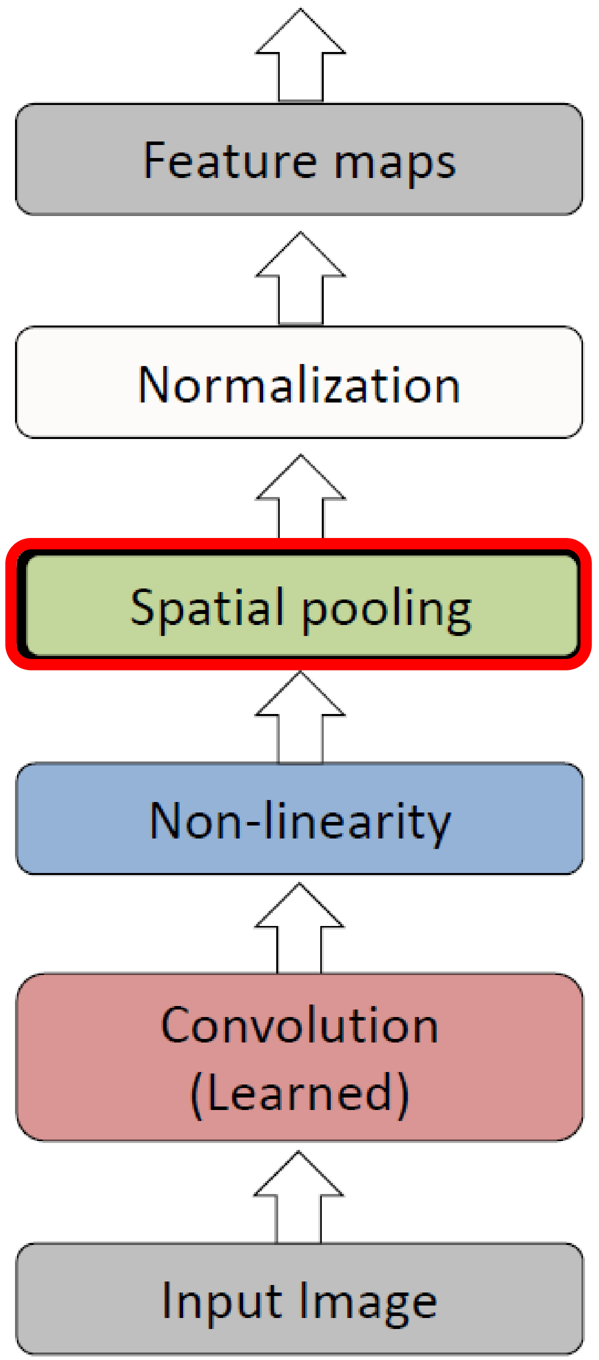
In another word …

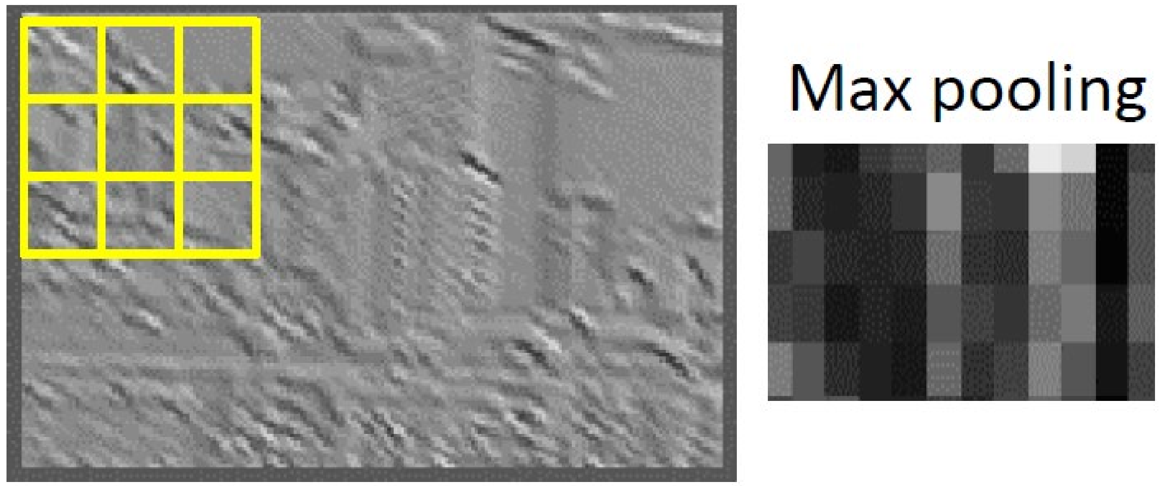
1. In general terms, the objective of pooling is to transform the joint feature representation into a new, more usable one that *preserves important information while discarding irrelevant detail*, the crux of the matter being to determine what falls in which category

2. Achieving invariance to changes in position or lighting conditions, robustness to clutter, and compactness of representation, are all common goals of pooling

Max Pooling

- ***Max pooling*** is a form of non-linear down-sampling, which partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value
- Max pooling is useful in vision for two reasons
  - By eliminating non-maximal values, it reduces computation for upper layers
  - It provides a form of translation invariance
- Since it provides additional robustness to position, max-pooling is a “smart” way of reducing the dimensionality of intermediate representations

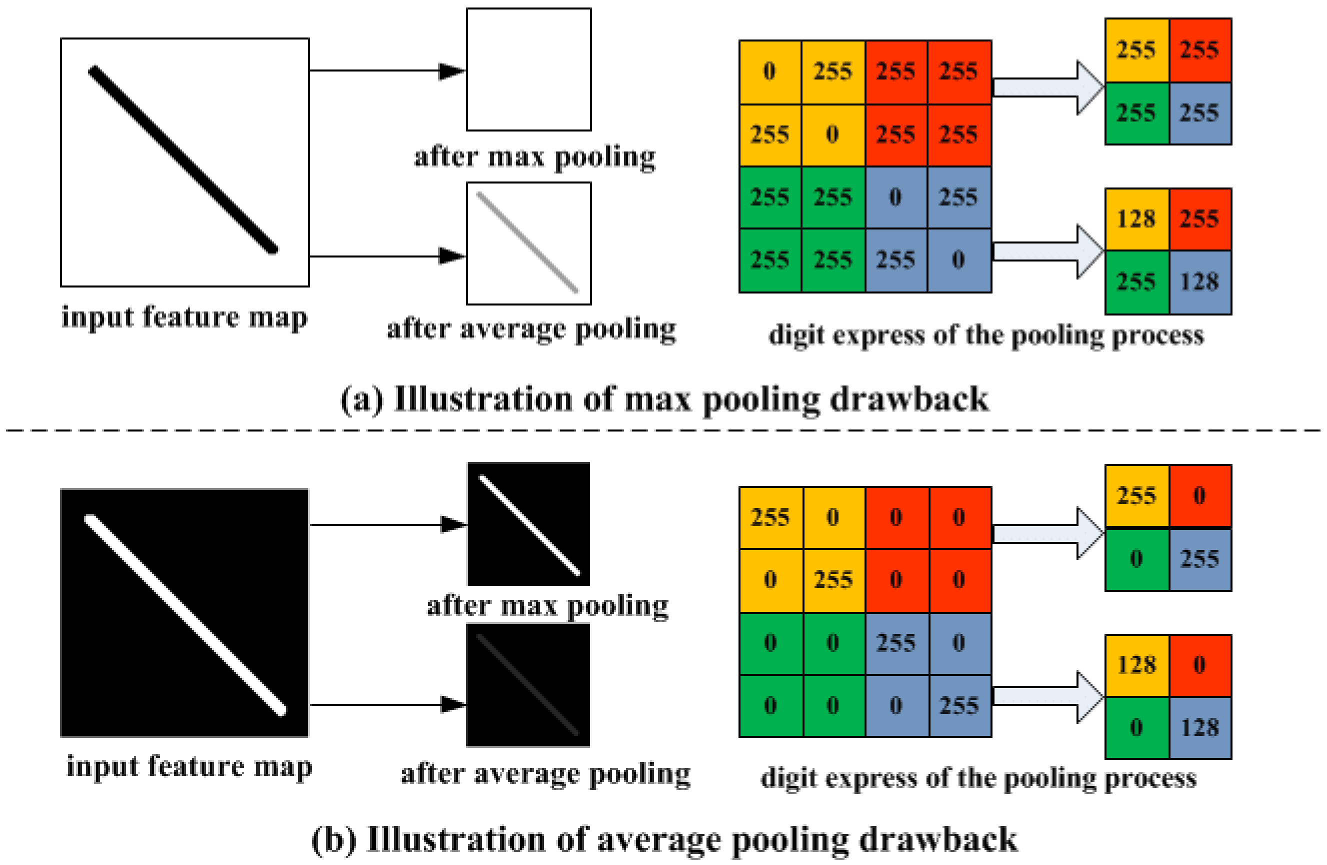




Max pooling:

- **a non-linear down-sampling**
- **Provide *translation invariance***

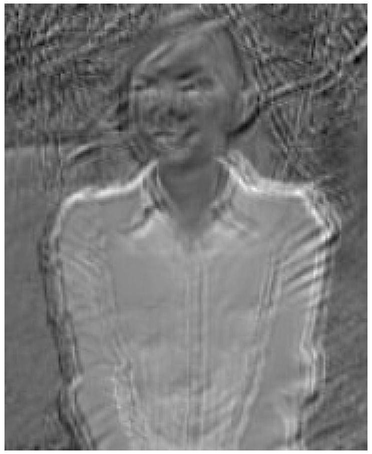
Max Pooling & Average Pooling



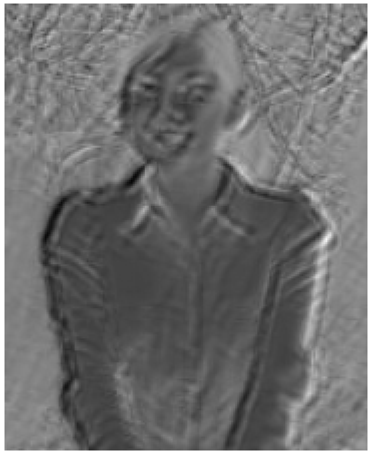
Example

origin image

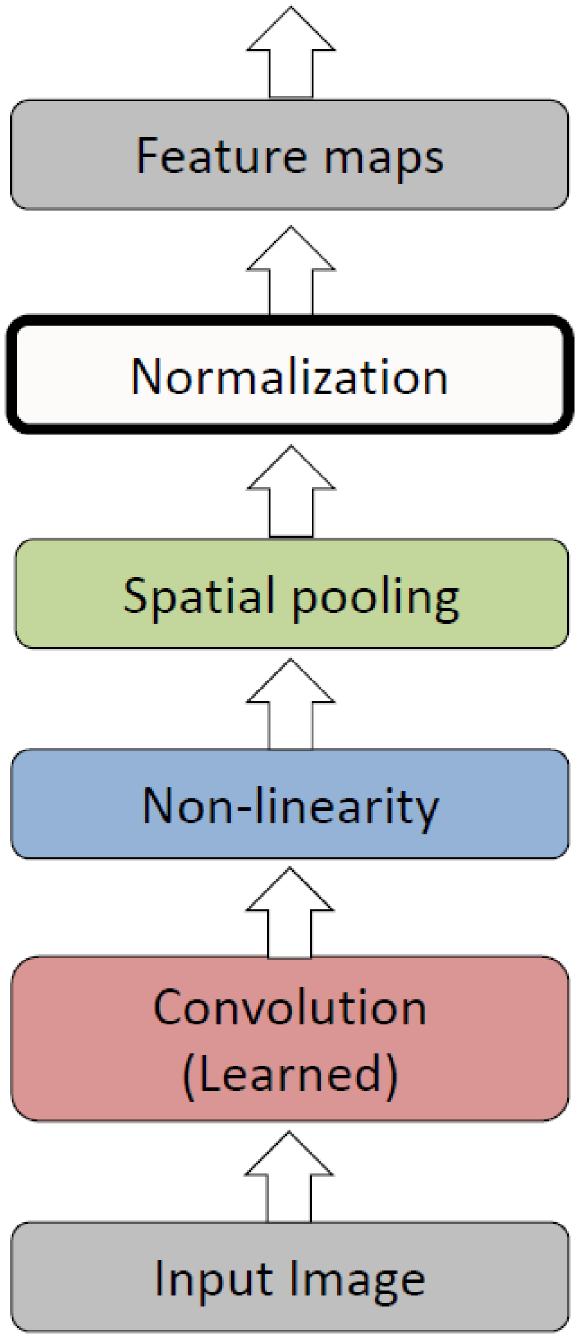
convolution 1



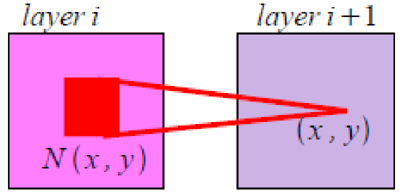
convolution 2



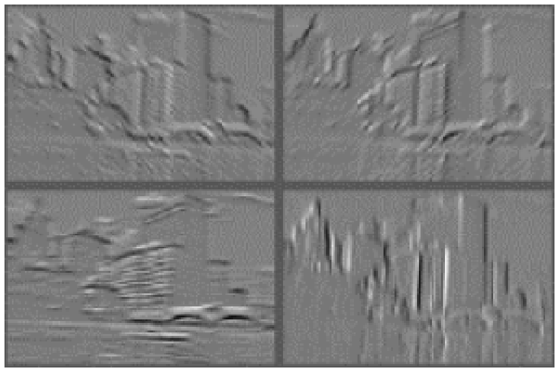
Down-sampled 1 **Down-sampled 2**



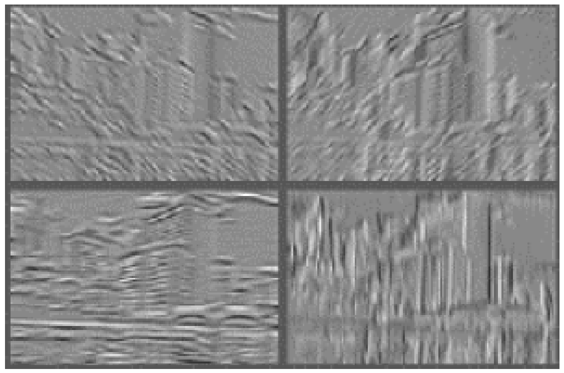
Local Contrast Normalization (over space / features)



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



Feature Maps

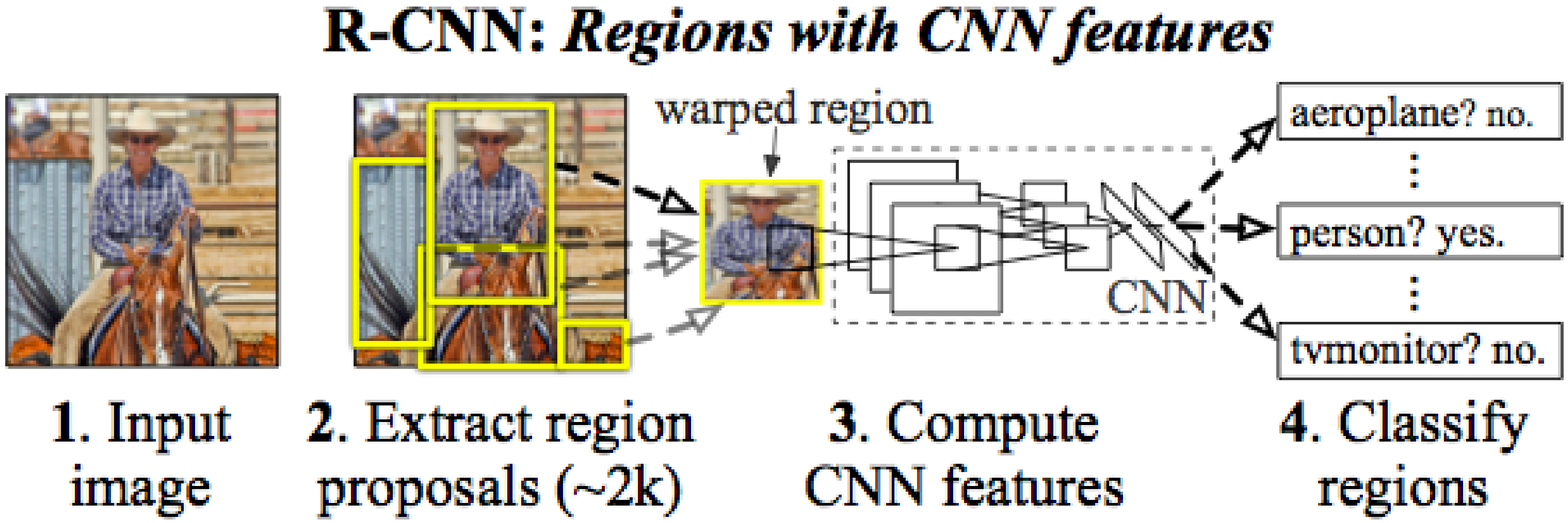


Feature Maps After Contrast Normalization

What is really important ?

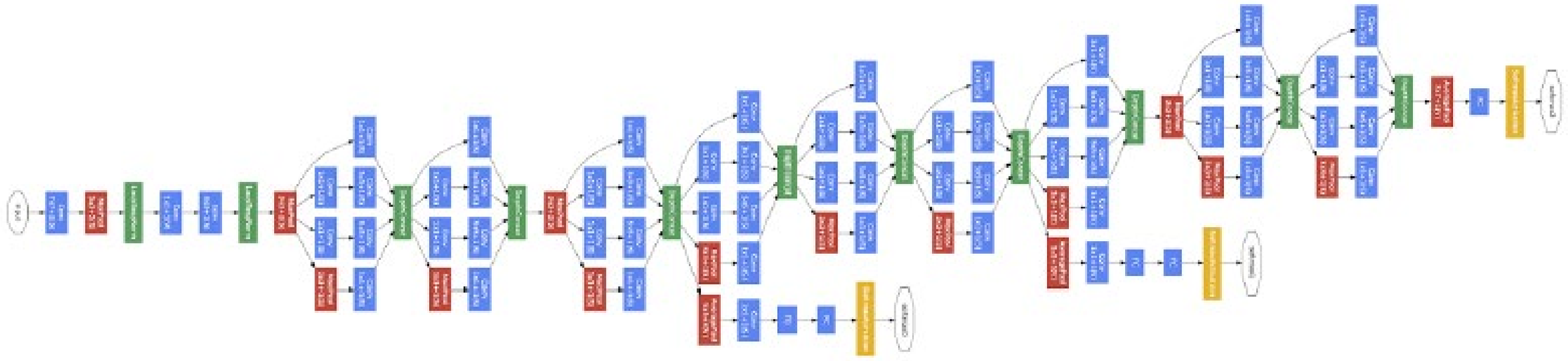
- The convolutional layers are the most important part
- A ***pre-trained*** network for image classification can be used for many different vision tasks.

Detection:



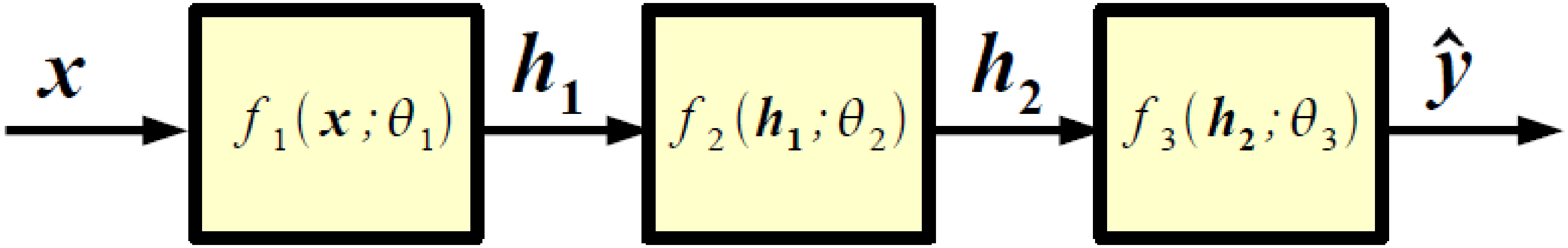
GoogLeNet

- 22 layers' deep networks



Convolution Max Pooling Softmax Concatenation

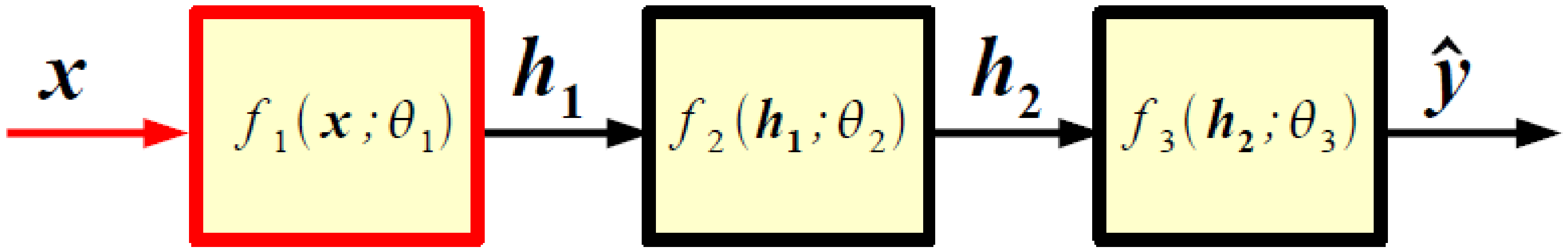
Idea of CNN training


$$\begin{tikzcd} \arrow[r, "\boldsymbol{x}"] & \boxed{f_1(\boldsymbol{x};\theta_1)} \arrow[r, "\boldsymbol{h}_1"] & \boxed{f_2(\boldsymbol{h}_1;\theta_2)} \arrow[r, "\boldsymbol{h}_2"] & \boxed{f_3(\boldsymbol{h}_2;\theta_3)} \arrow[r, "\hat{y}"] & {} \end{tikzcd}$$

NOTE:

In practice, any differentiable non-linear transformation is potentially good.

Forward Propagation (FP)



$\boldsymbol{x}$
$f_1(\boldsymbol{x};\theta_1)$
$\boldsymbol{h}_1$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\boldsymbol{h}_2$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\hat{y}$

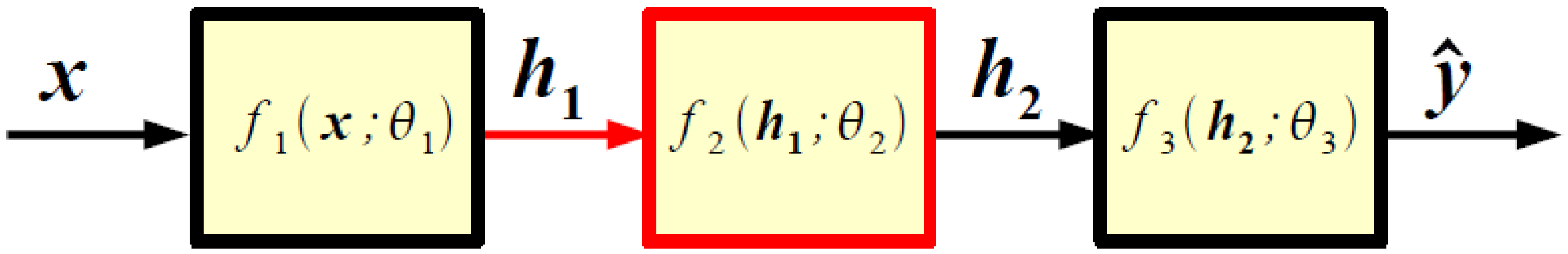
1) Given $\boldsymbol{x}$ compute: $\boldsymbol{h}_1 = f_1(\boldsymbol{x};\theta_1)$

For instance,

$$\boldsymbol{h}_1 = max(0, W_1 \boldsymbol{x} + \boldsymbol{b}_1)$$

ReLU Rectified Linear Units.

Forward Propagation (FP)

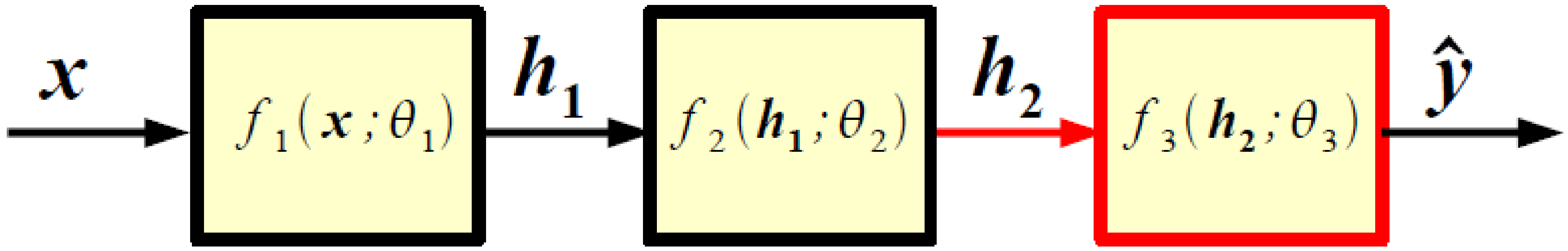


$\boldsymbol{x}$
$f_1(\boldsymbol{x};\theta_1)$
$\boldsymbol{h}_1$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\boldsymbol{h}_2$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\hat{y}$

1) Given $\boldsymbol{x}$ compute: $\boldsymbol{h}_1 = f_1(\boldsymbol{x};\theta_1)$

2) Given $\boldsymbol{h}_1$ compute: $\boldsymbol{h}_2 = f_2(\boldsymbol{h}_1;\theta_2)$

Forward Propagation (FP)



1) Given x compute: $h_1 = f_1(x;\theta_1)$

2) Given h_1 compute: $h_2 = f_2(h_1;\theta_2)$

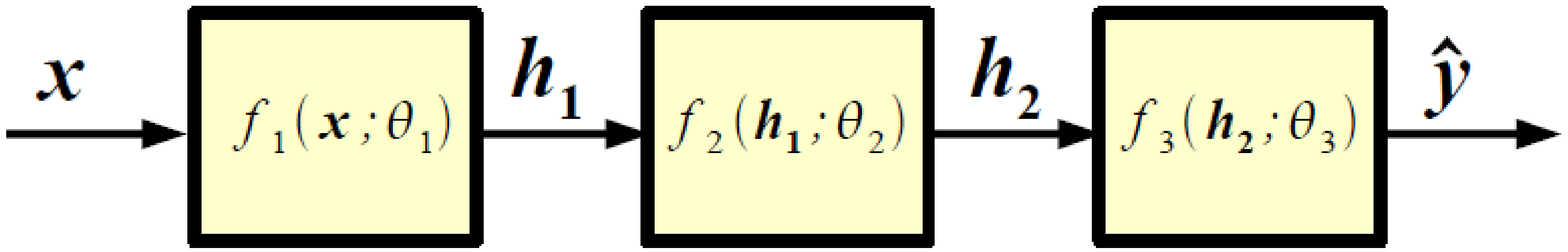
3) Given h_2 compute: $\hat{y} = f_3(h_2;\theta_3)$

For instance,

$$\hat{y}_i = p(class = i | x) = \frac{e^{W_{3i} h_2 + b_{3i}}}{\sum_k e^{W_{3k} h_2 + b_{3k}}}$$

Softmax output $\Rightarrow$ probability that input x belongs to one of the predefined classes

Forward Propagation (FP)



$\boldsymbol{x}$
$f_1(\boldsymbol{x};\theta_1)$
$\boldsymbol{h}_1$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\boldsymbol{h}_2$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\hat{y}$

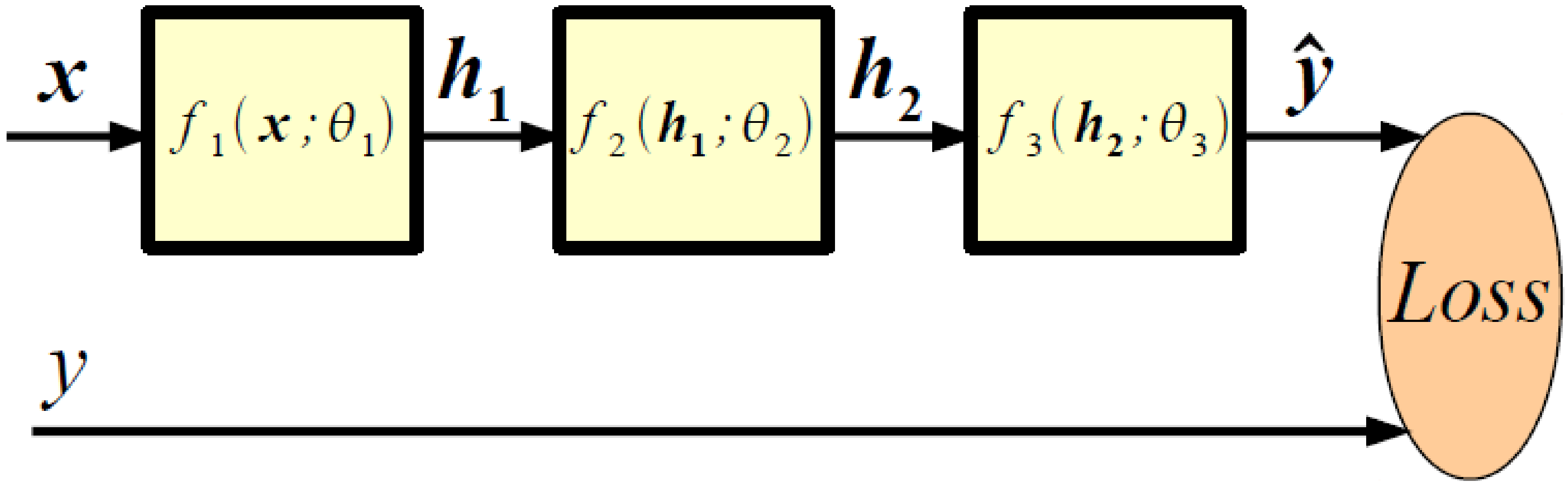
1) Given $\boldsymbol{x}$ compute: $\boldsymbol{h}_1 = f_1(\boldsymbol{x};\theta_1)$

2) Given $\boldsymbol{h}_1$ compute: $\boldsymbol{h}_2 = f_2(\boldsymbol{h}_1;\theta_2)$

3) Given $\boldsymbol{h}_2$ compute: $\hat{y} = f_3(\boldsymbol{h}_2;\theta_3)$

This is the typical processing at test time.
At training time, we need to compute an **error measure** and tune the parameters to decrease the error.

Loss Function

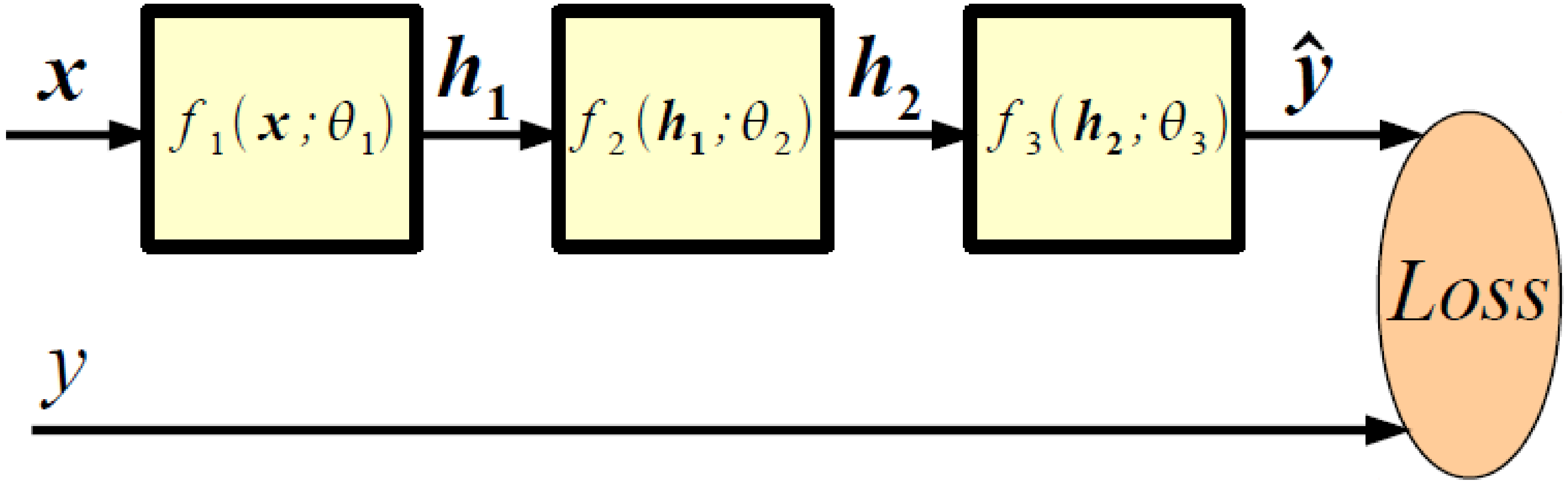


x
$f_1(\boldsymbol{x};\theta_1)$
$\boldsymbol{h}_1$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\boldsymbol{h}_2$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\hat{y}$
$Loss$
y

The measure of how well the model fits the training set can be given by a suitable ***loss function***: $L(x, y; \theta)$

The loss depends on the input x , the target label y , and the parameters θ

Backward Propagation (BP)

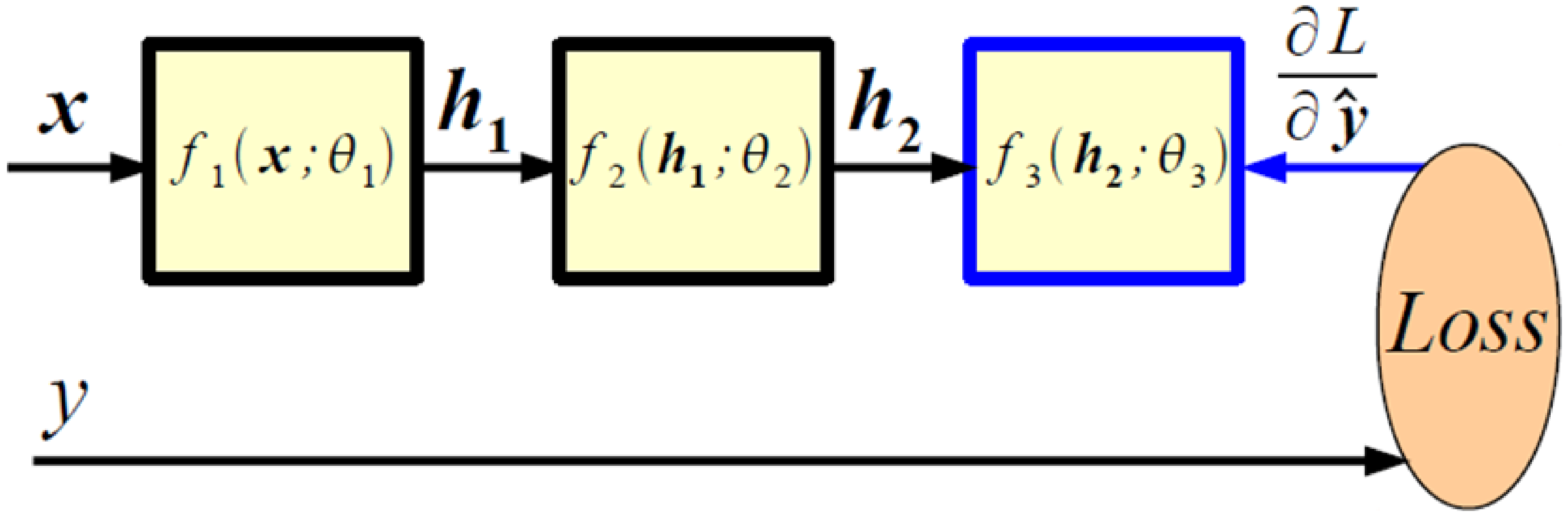


x
$f_1(\boldsymbol{x};\theta_1)$
$\boldsymbol{h}_1$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\boldsymbol{h}_2$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\hat{y}$
$Loss$
y

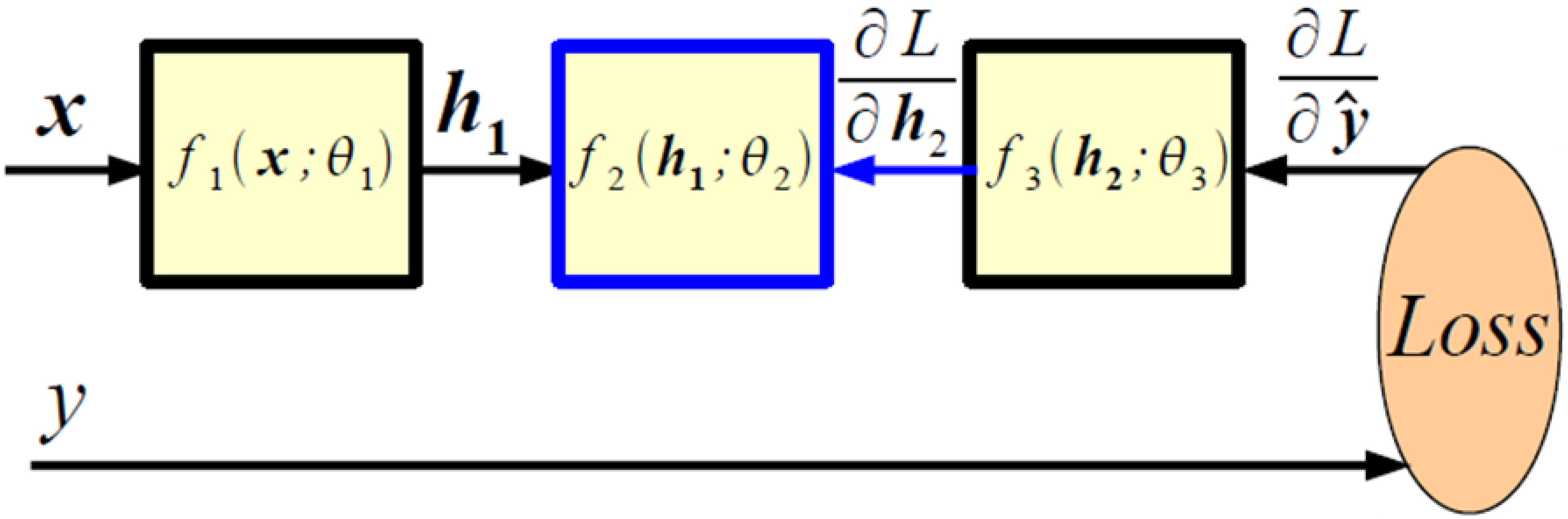
Q.: how to tune the parameters to decrease the loss?

If loss is differentiable we can compute gradients.
We can use **back-propagation,** to compute the gradients w.r.t. parameters at the lower layers

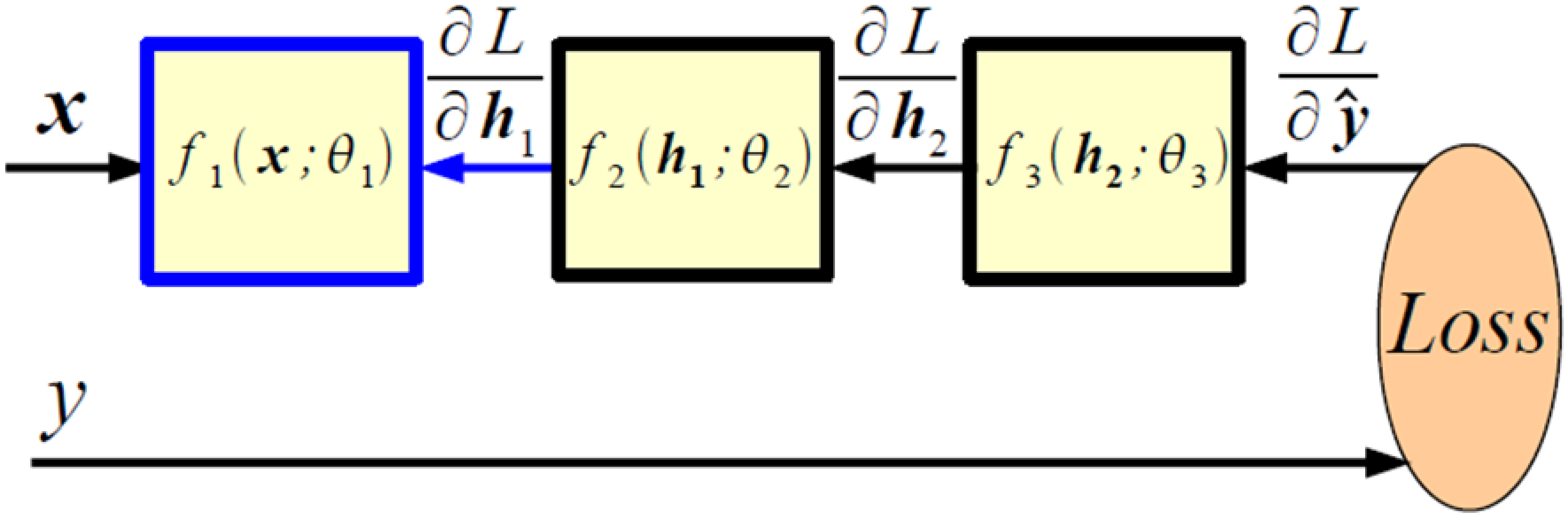
Backward Propagation (BP)



Backward Propagation (BP)



Backward Propagation (BP)


x
$f_1(\boldsymbol{x};\theta_1)$
$\frac{\partial L}{\partial \boldsymbol{h}_1}$
$f_2(\boldsymbol{h}_1;\theta_2)$
$\frac{\partial L}{\partial \boldsymbol{h}_2}$
$f_3(\boldsymbol{h}_2;\theta_3)$
$\frac{\partial L}{\partial \hat{\mathbf{y}}}$
$Loss$
y

Optimization

Stochastic Gradient Descent (on mini-batches):

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}, \eta \in R$$

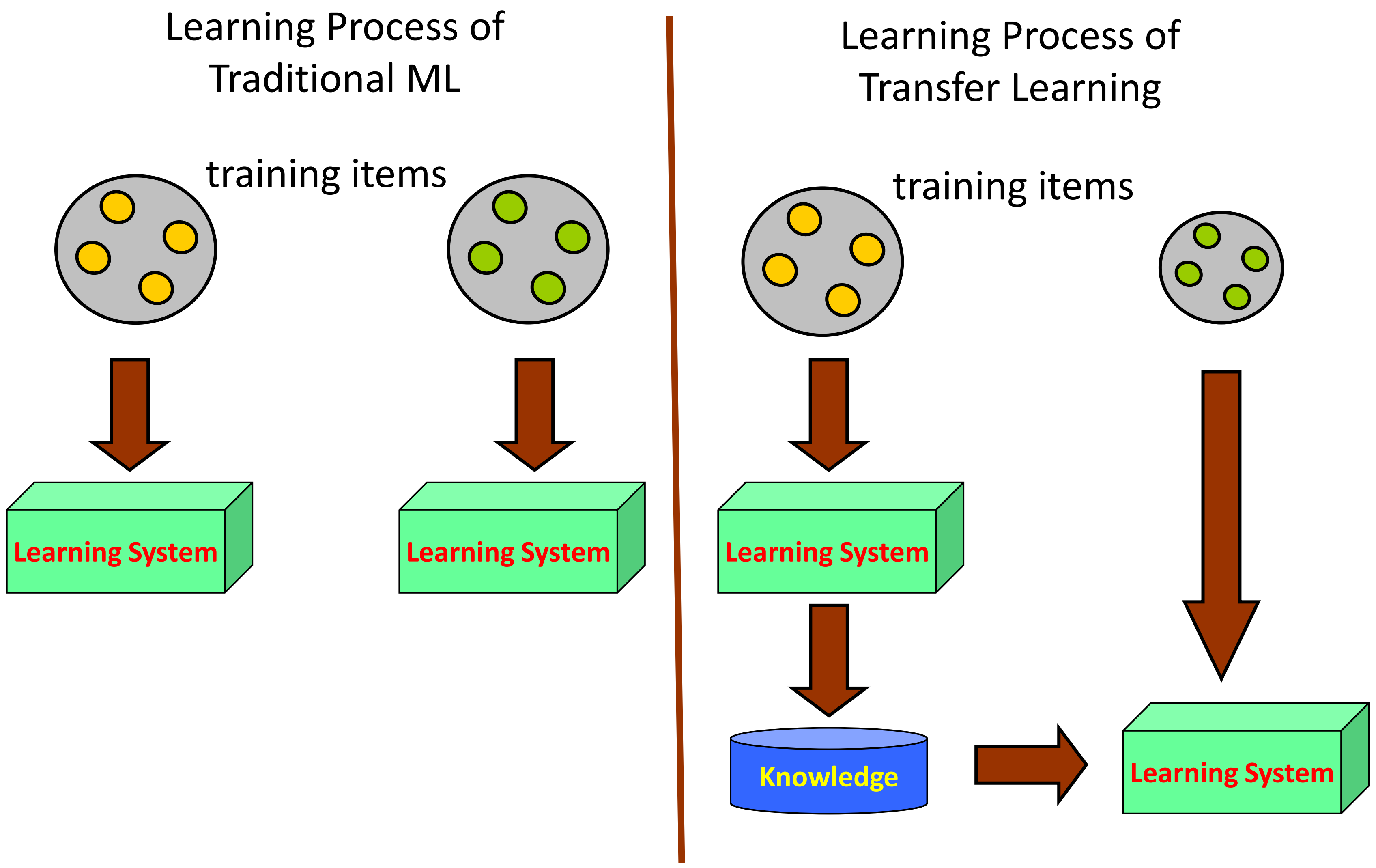
Stochastic Gradient Descent with Momentum:

$$\theta \leftarrow \theta - \eta \Delta$$
$$\Delta \leftarrow 0.9 \Delta + \frac{\partial L}{\partial \theta}$$

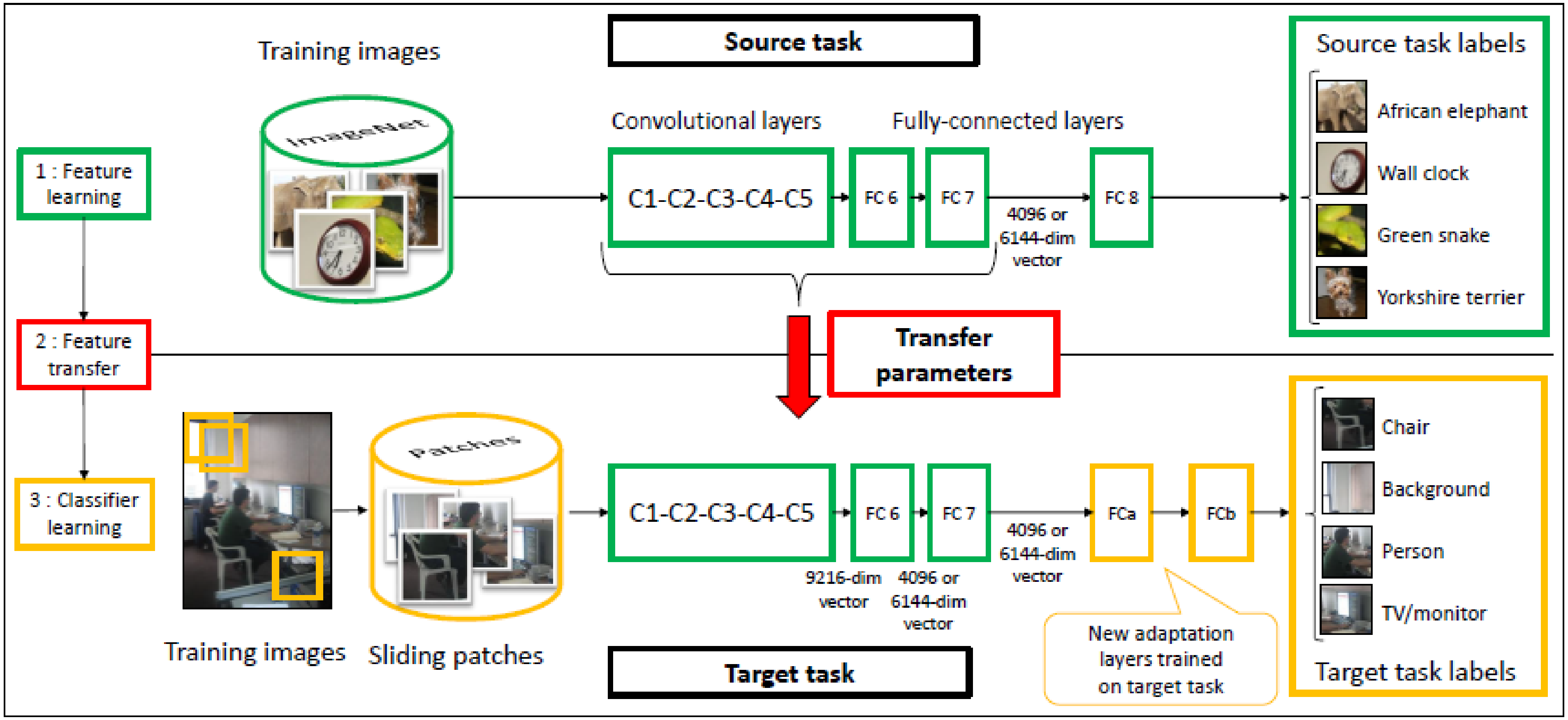
Transfer Learning (TL)

- The ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks (in new domains)
- TL is motivated by human learning: people can often transfer knowledge learnt previously to novel situations
  - Chinese $\rightarrow$ English
  - mathematics $\rightarrow$ computer science
  - network technology for internet $\rightarrow$ social network

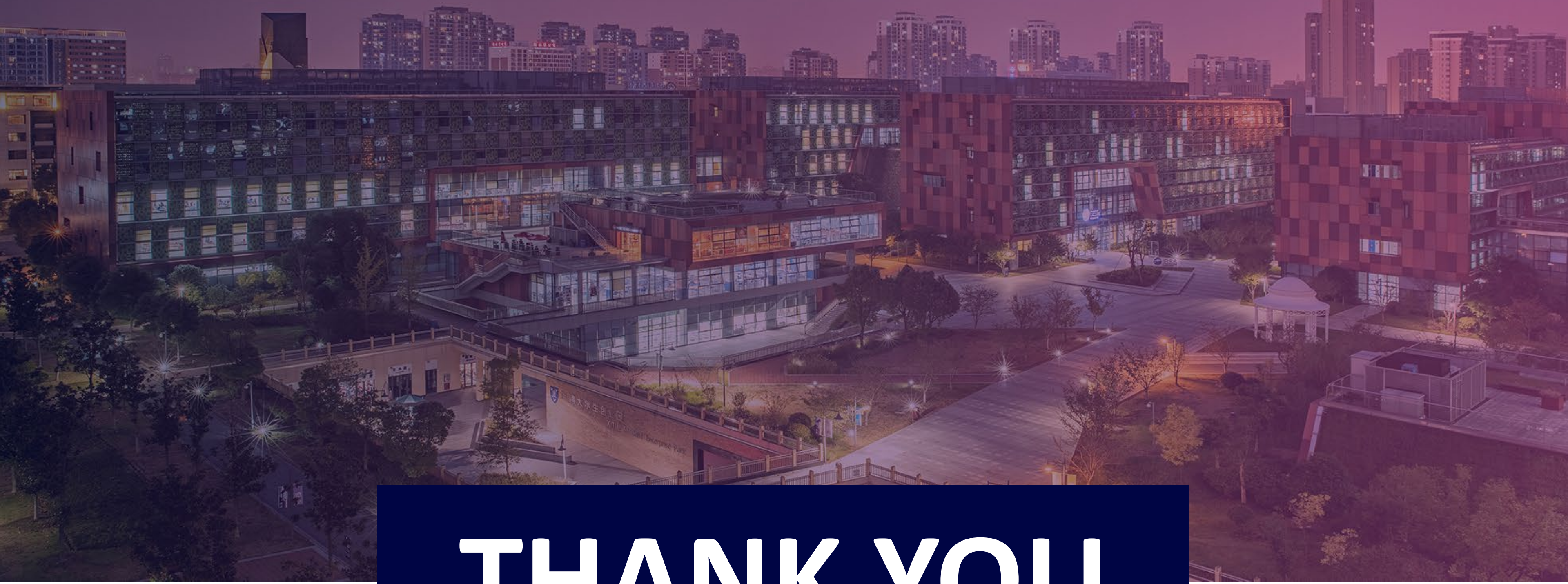
Traditional ML vs. TL



Deep CNN for Knowledge Transfer



The network is trained on the source task (top row) with a large amount of available labelled images. Pre-trained parameters of the internal layers of the network (C1-FC7) are then transferred to the target tasks (bottom row). To compensate for the different image statistics (type of objects, typical viewpoints, imaging conditions) of the source and target data, an adaptation layer (fully connected layers FCa and FCb) is added and trained on the labelled data of the target task.



THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学