So Far…

- Our goal (supervised learning)
  - Learn a set of discriminant functions
  - Bayesian framework
    - We could design an optimal classifier if we knew:
      - $P(\omega_i)$: priors and $P(x \mid \omega_i)$: class-conditional densities
      - Using training data to estimate $P(\omega_i)$ and $P(x \mid \omega_i)$
  - Directly learning discriminant functions from the training data
    - Linear Regression
    - Logistic Regression
    - SVM
    - Kernel methods

Disadvantage?

- Only focus on the classifier.
- We have to know the form of the discriminant functions (the nonlinearity).

Artificial Neural Network & Deep Learning

Deng Cai (蔡登)

College of Computer Science
Zhejiang University

dengcai@gmail.com

2019 Turing Award Winners

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built.

What is Deep Learning

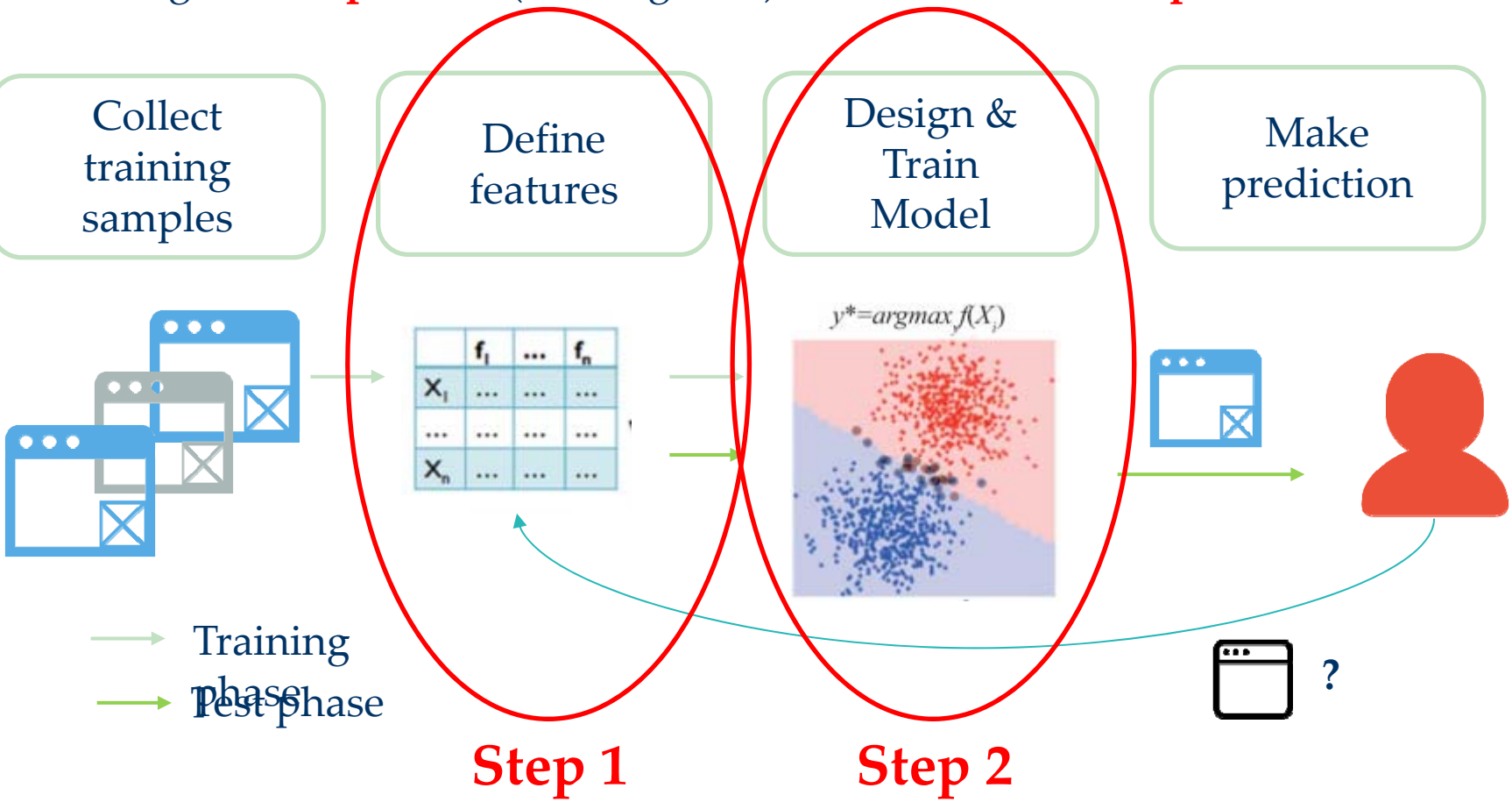
Deep learning is a branch of **machine learning** based on learning **representations** of data. It's a set of algorithms that attempt to model high-level abstractions in data by using **multiple processing layers**, with complex structures or otherwise, composed of **multiple non-linear transformations**.

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data.

Wiki

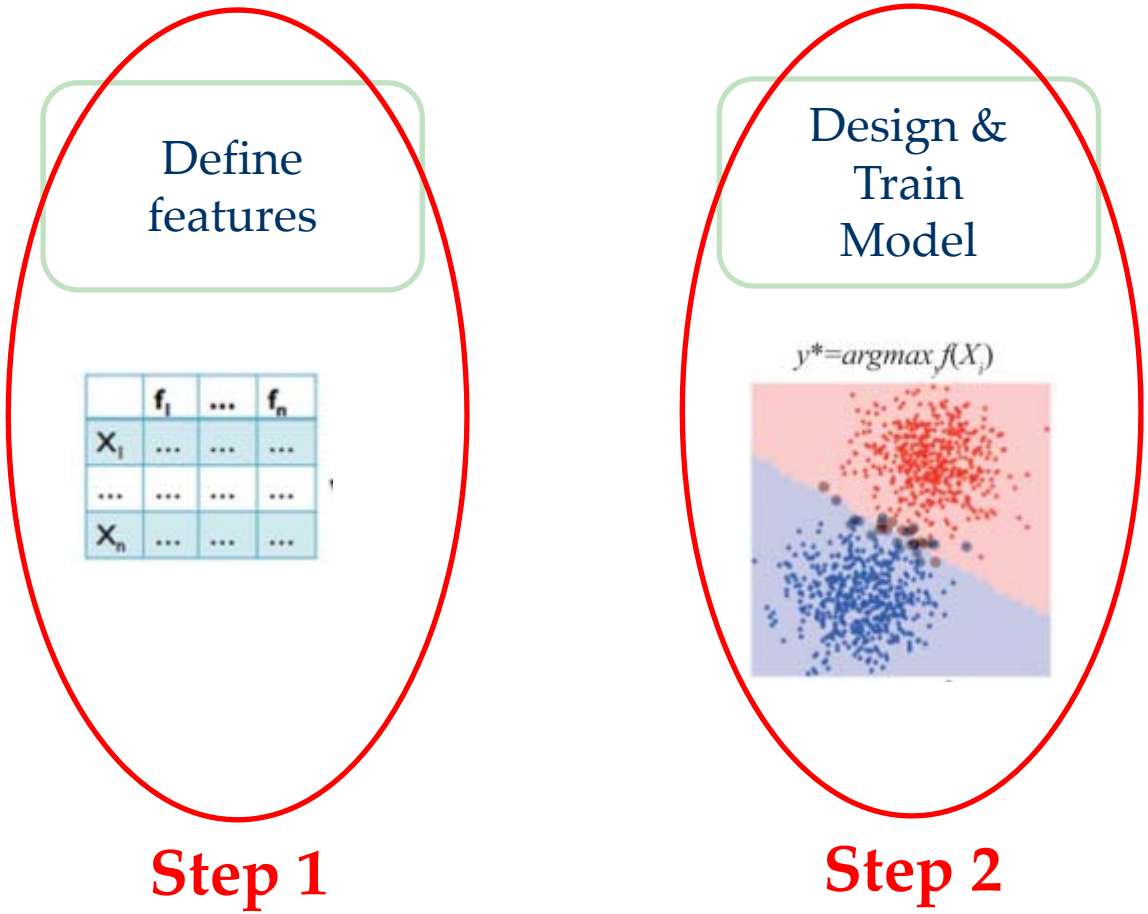
Supervised Learning

Learning from **experience**(training data), and build **model** to **predict** the future



Representation Learning

Supervised Learning



- Step 1 is more important in building a successful system.

Why general classification hard?

- Intra-class variability

The letter “T” in different typefaces

Define features

	f_1	...	f_n
x_1	...	...	...
...	...	...	...
x_n	...	...	...

Step 1 is not good enough

Same face under different expression, pose, illumination

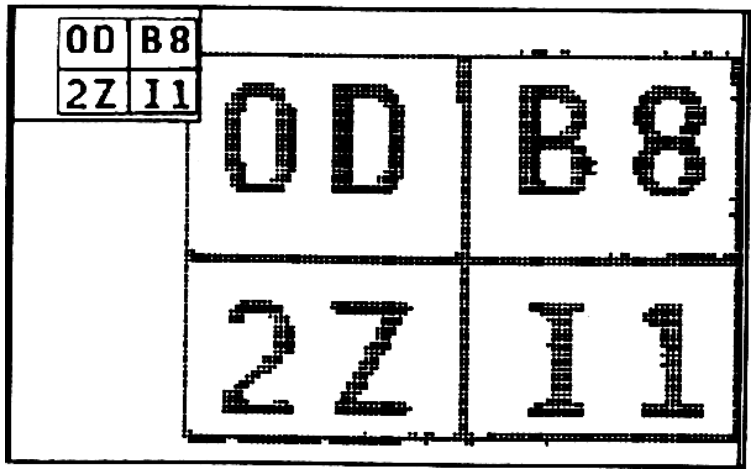
Why general classification hard?

Define features

	f_1	...	f_n
x_1	...	...	...
...	...	...	...
x_n	...	...	...

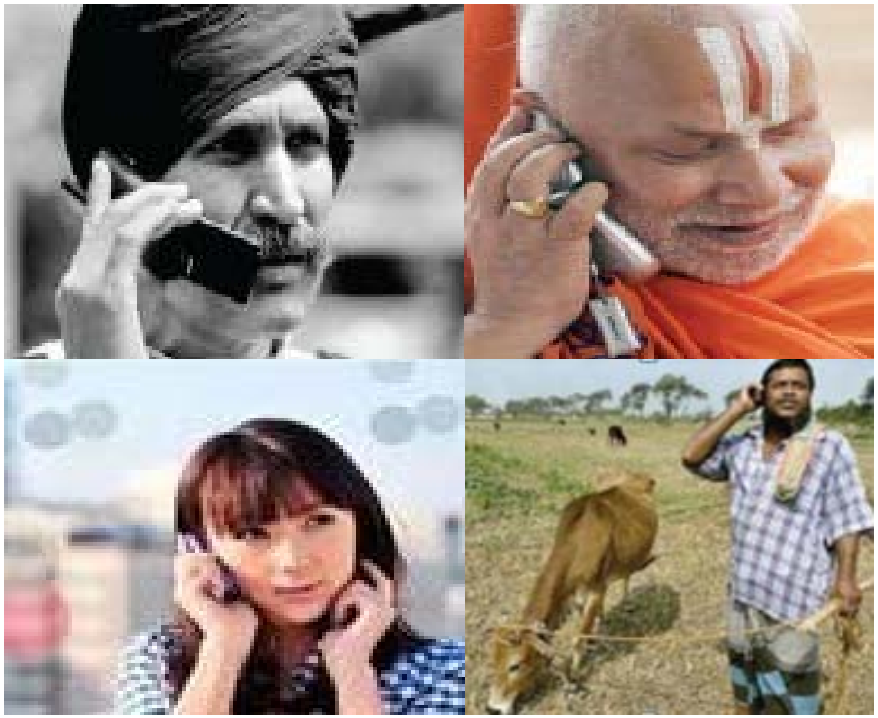
Step 1 is not good enough

- Inter-class similarity



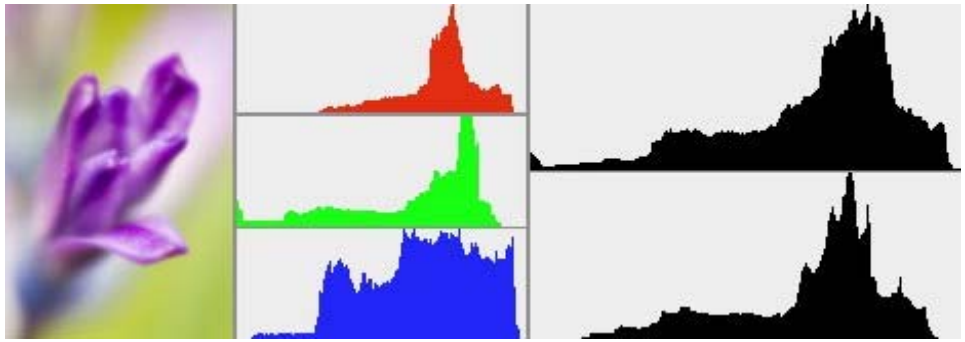
Semantic Gap

Looks similar
But semantically different

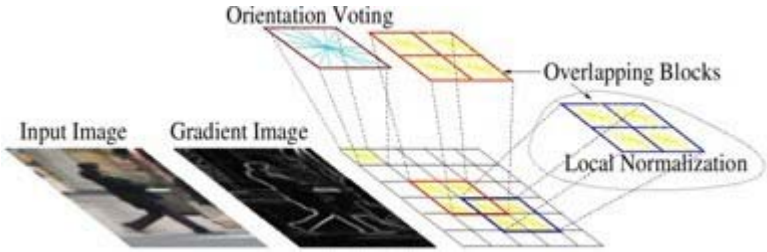


Looks different
But semantically the same

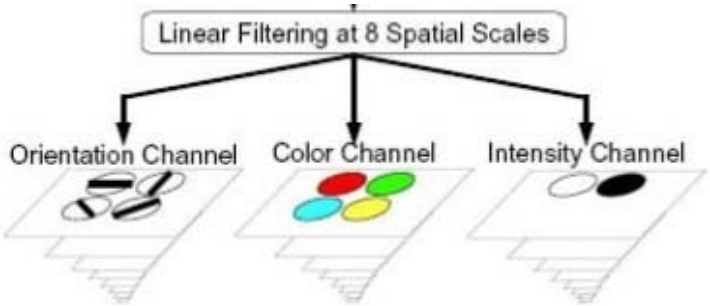
Representation (Feature)



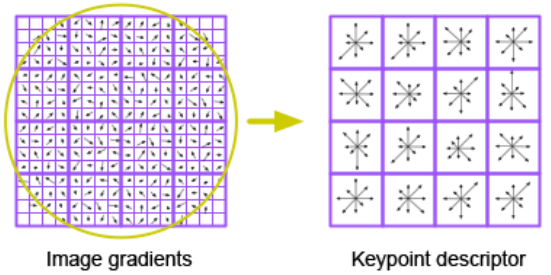
colorHistogram,1992



HOG (Histogram of Oriented Gradients),2005



GIST,2001, 2003



SIFT,1999, 2004

SIFT

Distinctive image features from scale-invariant keypoints

作者 David G Lowe

發佈日期 2004/11/1

期刊 International journal of computer vision

冊別 60

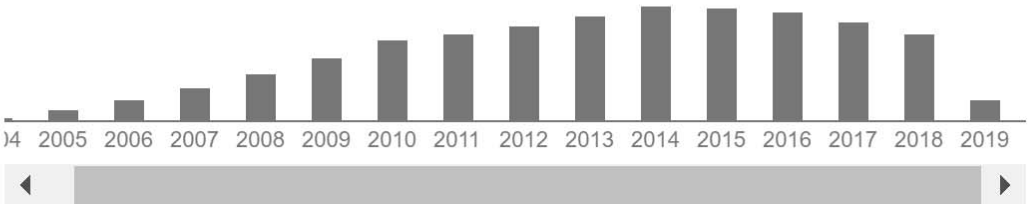
期數 2

頁數 91-110

發佈機構 Springer Netherlands

描述 This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through …

引文總數 被引用 50302 次



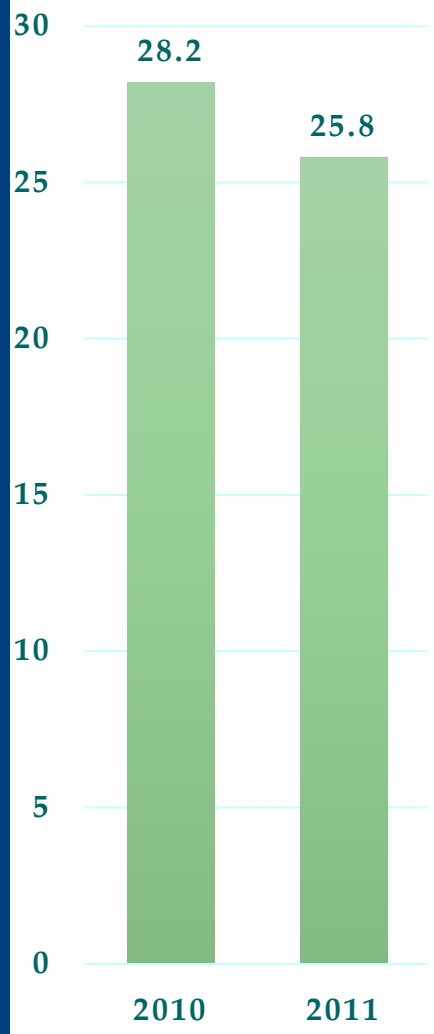
Imagenet competition

Large Scale Visual Recognition Challenge (ILSVRC)

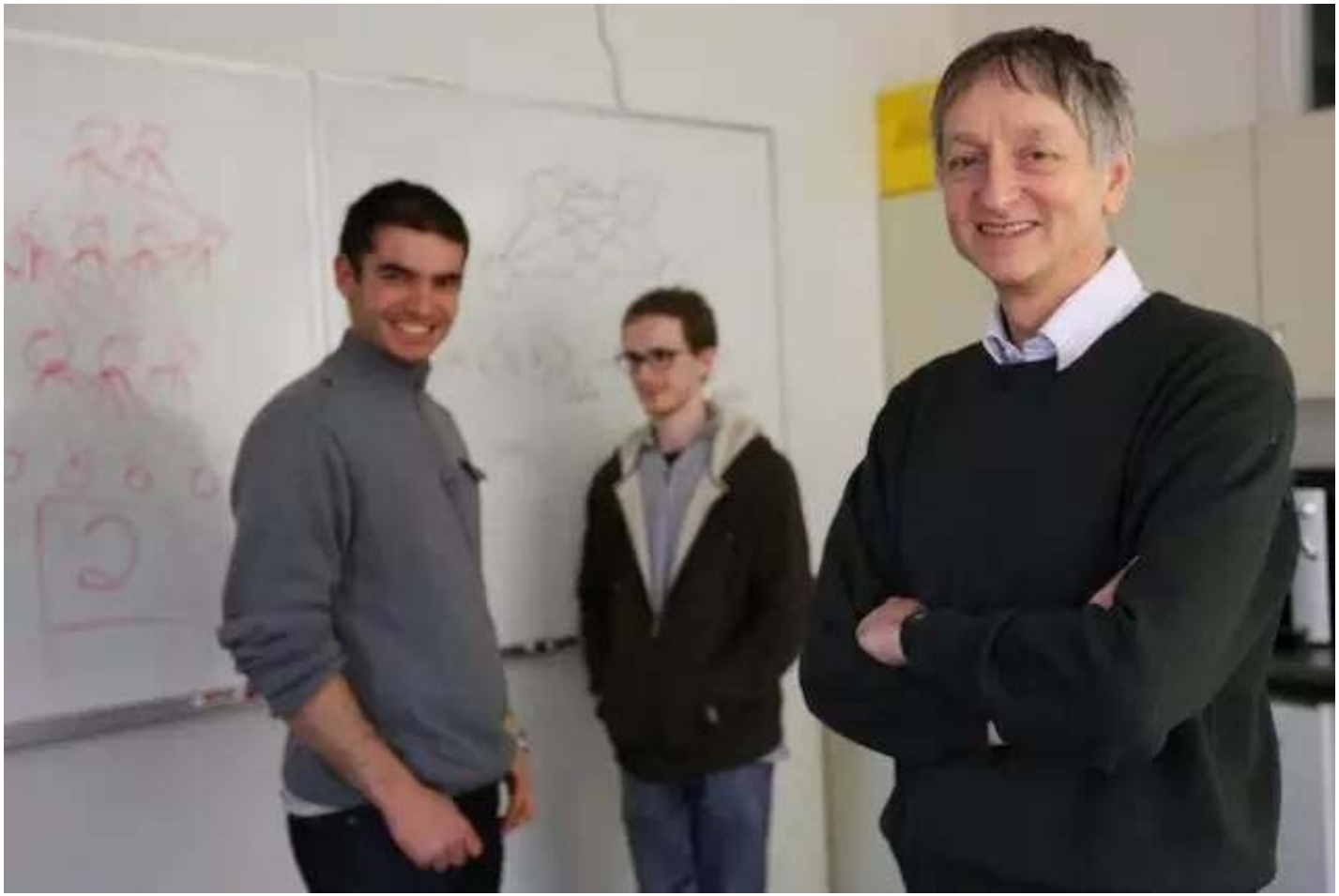
2010 ~ 2017

1000 categories, from flickr and other search engines
1.2 million training images
50,000 validation images , **100,000** test images

ImageNet Classification top-5 error(%)

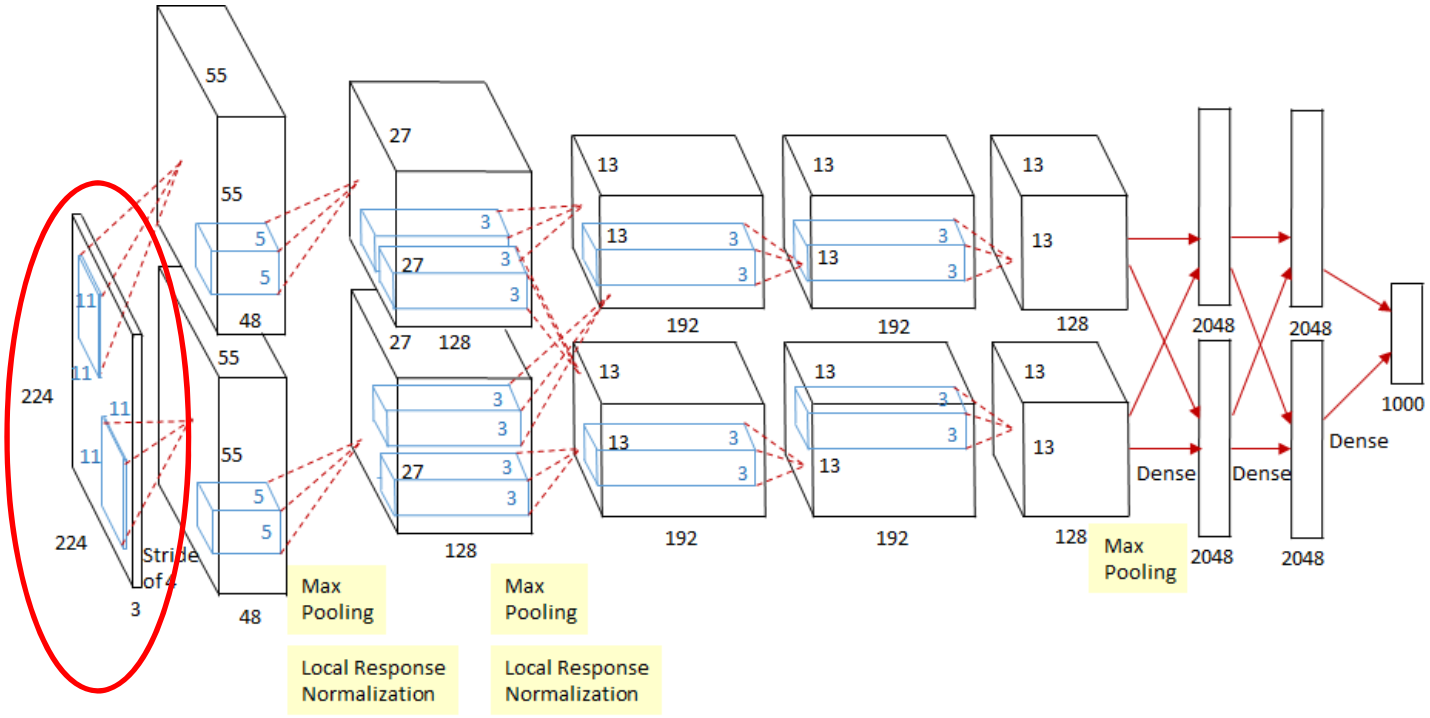


Imagenet break through

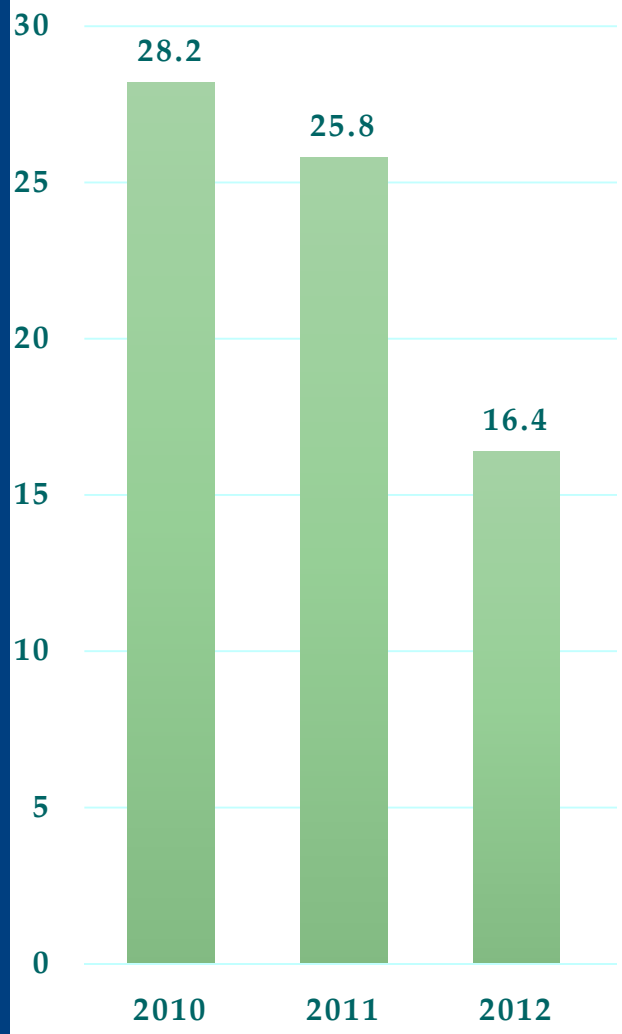


Suskever, Krizhevsky, Hinton, from left to right

AlexNet (Alex Krizhevsky)



ImageNet Classification top-5 error(%)



Deep Learning
(Neural Network, NN)
(Artificial Neural Network, ANN)

Linear model (classifier)

- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \begin{cases} > 0 & class\ 1 \\ \\ < 0 & class\ 2 \end{cases}$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$$

$$\boldsymbol{x} = [x_1, x_2, \cdots x_d, 1]^T \in R^{d+1}$$

$$\boldsymbol{w} = [w_1, w_2, \cdots w_d, b]^T \in R^{d+1}$$

Linear model (classifier)

- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = 0$$

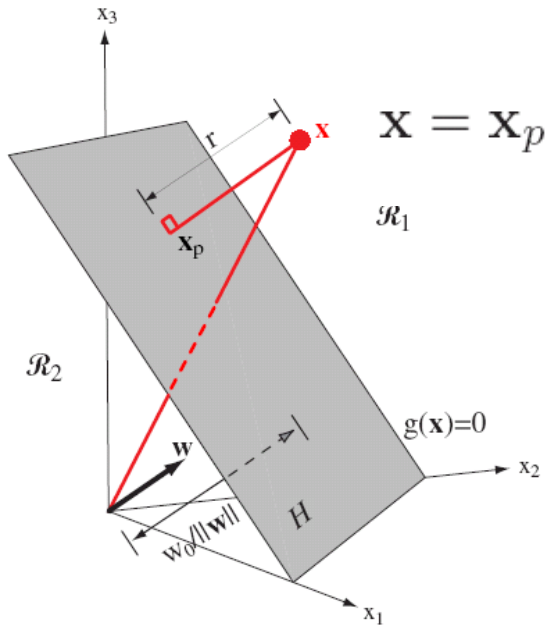
- Decision surface
- 1: What is the normal vector of this hyper-plane?

$$f(\boldsymbol{x}_1) = \boldsymbol{w}^T \boldsymbol{x}_1 + b = 0 = f(\boldsymbol{x}_2) = \boldsymbol{w}^T \boldsymbol{x}_2 + b$$

$$\boldsymbol{w}^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$$

- 2: What is the distance of any point x to this hyperplane?

Decision Surface



$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$f(\boldsymbol{x}_p) = \boldsymbol{w}^T\left(\boldsymbol{x} - r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + b = 0$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = r\|\boldsymbol{w}\| \qquad r = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

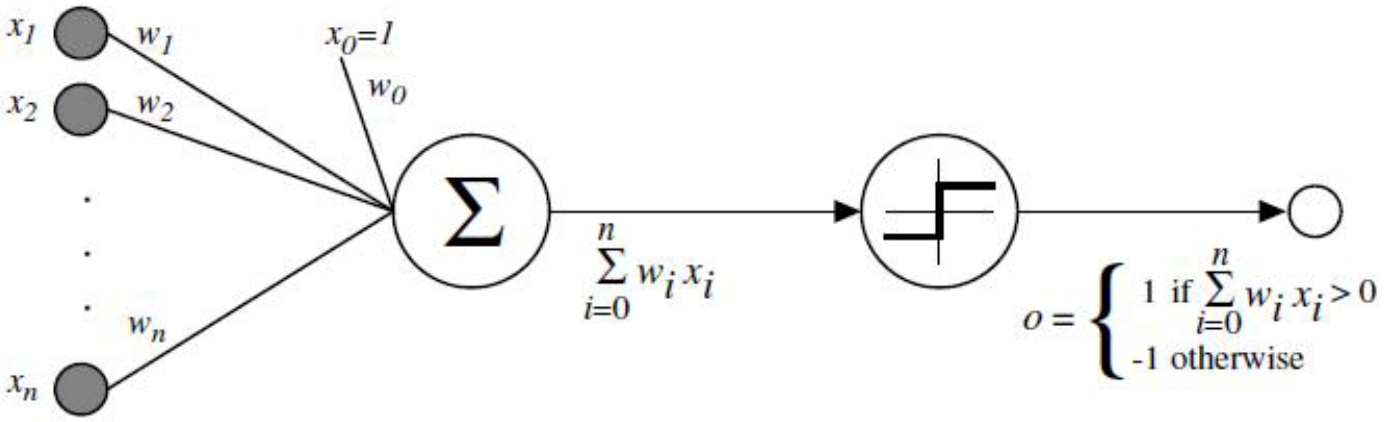
Two-category Linearly Separable Case

- If
  - $\boldsymbol{w}^T\boldsymbol{x} > 0$ for examples from the positive class.
  - $\boldsymbol{w}^T\boldsymbol{x} < 0$ for examples from the negative class.
- Such a weight vector $\boldsymbol{w}$ is called a *separating vector* or a *solution vector*
- Normalizing the input examples by multiplying them with their class label (replace all samples from class 2 by their negatives)
  - $\boldsymbol{x} \rightarrow \boldsymbol{z}$
  - $\boldsymbol{w}^T\boldsymbol{z} > 0$ for all the examples (here $\boldsymbol{z}$ is $\boldsymbol{x}$ multiplied with class label)

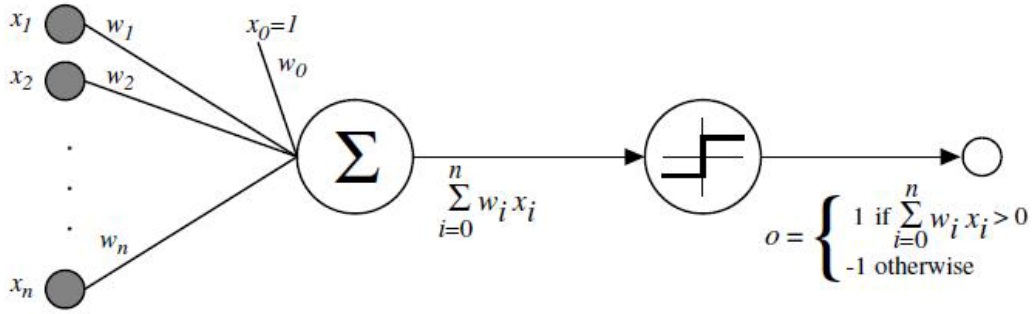
Linear model (classifier)

- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \begin{cases} > 0 & class\ 1 \\ < 0 & class\ 2 \end{cases}$$



Perceptron

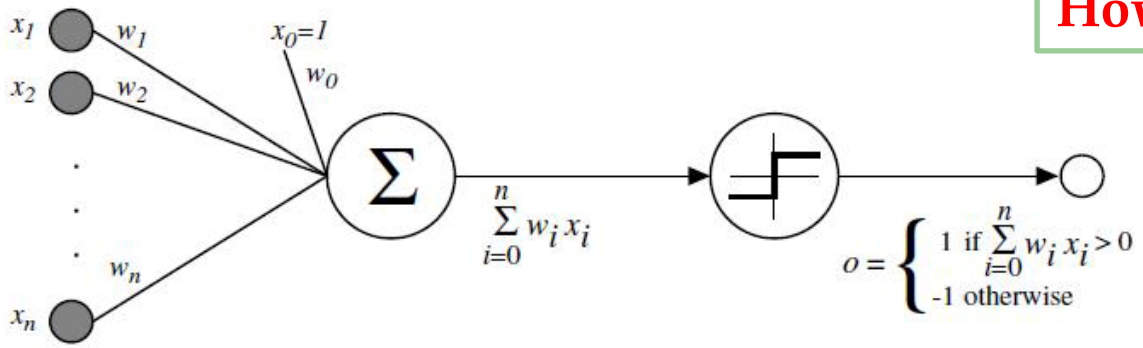


- In 1957, **Frank Rosenblatt** in Cornell University invented the **Perceptron** model. It is:
  - The first self-organizing and self-learning mathematic model.
  - The first algorithm defining Neural Network precisely.
  - The ancestor of many new Neural Network models.

Frank Rosenblatt
(1928-1971)

Perceptron

How to Learn?



- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$

- Seems correct buy why?
- Hint: If w is a solution, for any $\boldsymbol{x}$ in the training set, we have

$$\boldsymbol{w}^T \boldsymbol{x}y > 0$$

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)

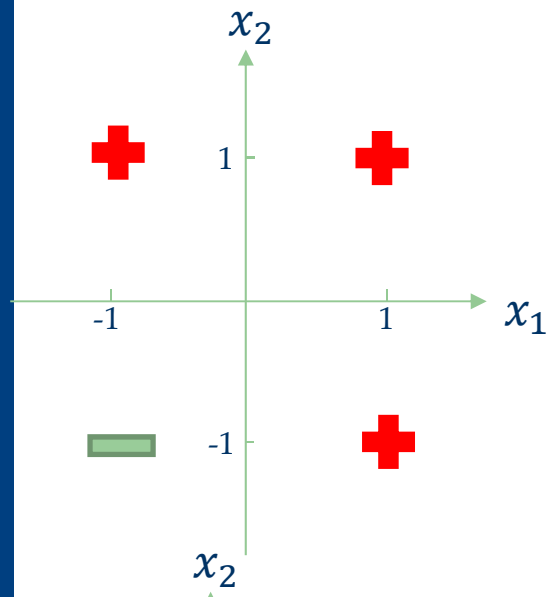
- This model had been quite popular in the 1960s.

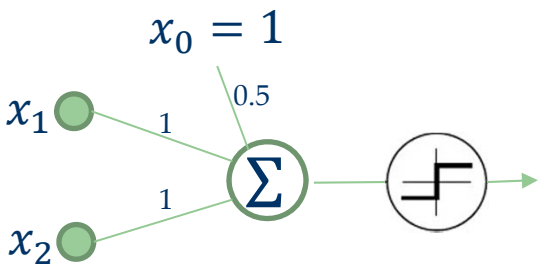
  - Rosenblatt had a hopeful view of percetron, predicting that it would be able to learn, to make decisions, and to translate.

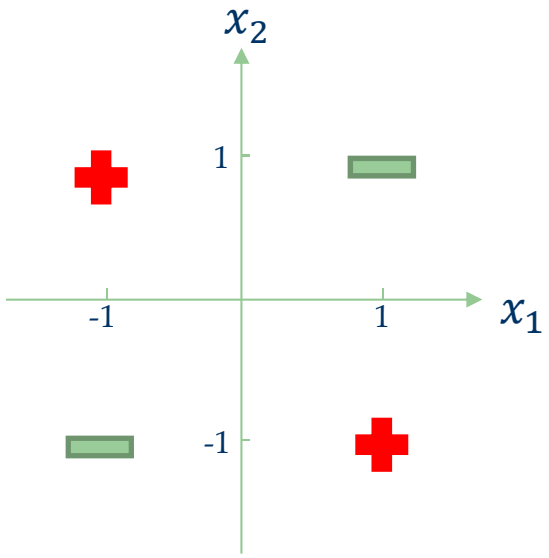
  - The US Navy used to fund its research, expecting that it "*will be able to walk, talk, see, write, reproduce itself and be conscious of its existence*", till the first AI winter came.

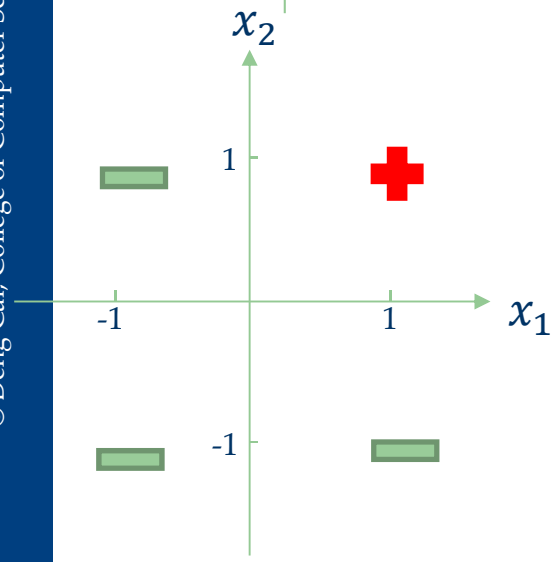
Perceptron: Drawback

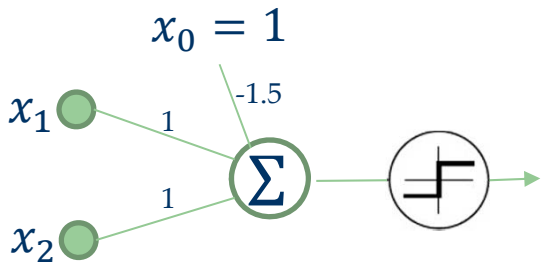
- Only capable of learning **linearly separable functions.**









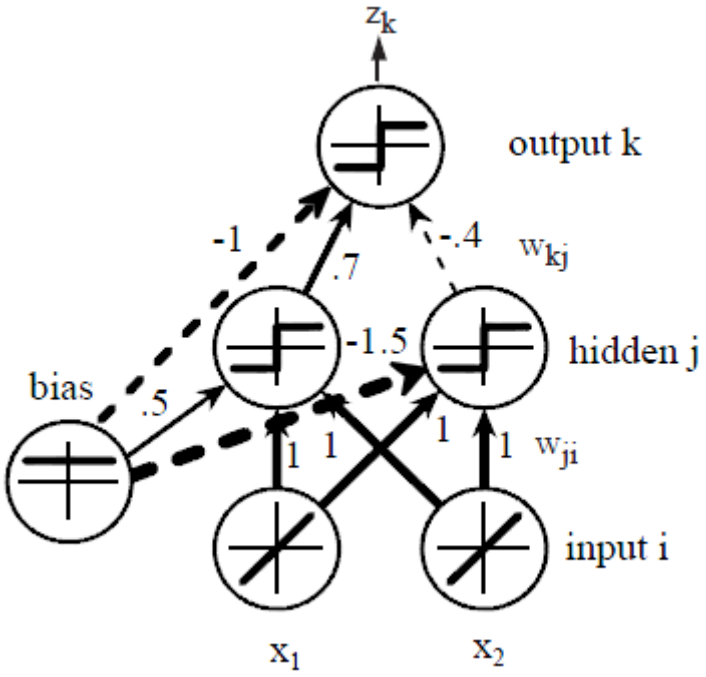


The First Winter

- In 1969, Marvin Minsky and Seymour Paper published the monograph "*Perceptrons: an introduction to computational geometry* ".
- This book revealed two fatal shortcomings of the perceptron model:
  - Single layered perceptron is not able to solve problems with non-linearity , e.g. XOR gate.
  - Its computation budget was too huge for the computers at that age.
- In the next more than ten years, the NN-based AI researches were nearly abandoned and related projects could not get any funds from the government.
- This period has been called the first AI Winter.
- Rosenblatt died in 1971 in an accident, not lucky enough to witness the renaissance of NN research.

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by "Perceptrons"

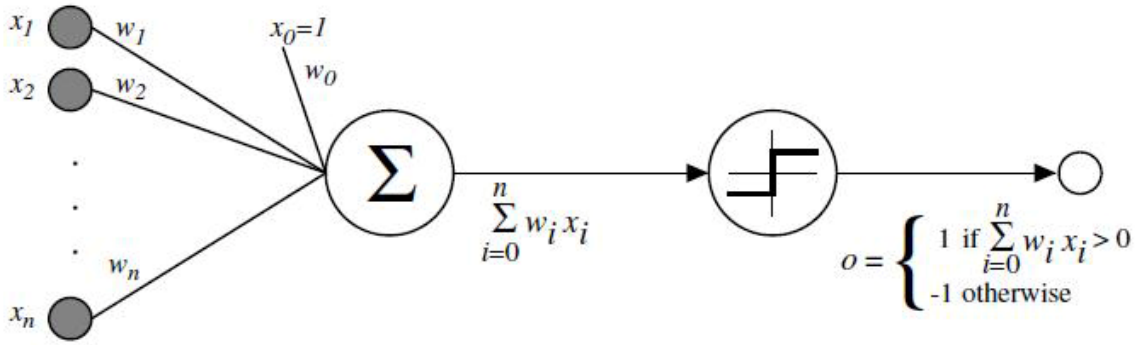


x_1	x_2	y_1	y_2	z
1	1	1	1	-1
1	-1	1	-1	1
-1	1	1	-1	1
-1	-1	-1	-1	-1

$$z = x_1 \, XOR \, x_2$$

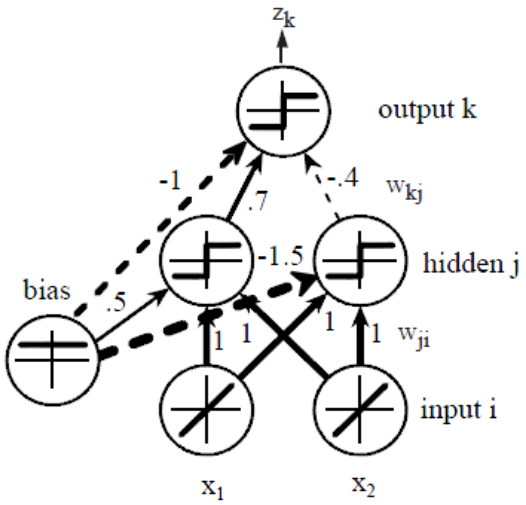
- What is the relation between z and x_i?
  - Assume x_1 and x_2 are -1, +1

Perceptron vs. Multi-layer Perceptron



- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$



How to Learn?

Principle Way To Learn Perceptron

- Define the Criterion Function (Loss Function) of Perceptron
- Then optimize(minimize) this function

- What is the criterion function of perceptron?
  - If an example can be correctly predicted, No penalty.
  - If the model made an error on an example, how to put the penalty?

Hint: $r = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|}$ $\quad J(\boldsymbol{w}) = -\sum_{i \in I_M} \boldsymbol{w}^T \boldsymbol{x}_i y_i$

The criterion is proportional to the sum of distances from the misclassified samples to the decision boundary.

Gradient Descent

Optimization Algorithm

$$J(\boldsymbol{w}) = -\sum_{i \in I_M} \boldsymbol{w}^T \boldsymbol{x}_i y_i$$

- Use gradient descent to find $\boldsymbol{w}$
  - Move in the negative direction of the gradient iteratively to reach the minima.
- The gradient vector is given by,

$$\nabla J = \sum_{i \in I_M} -\boldsymbol{x}_i y_i$$

- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let $\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$

- Starting from $\boldsymbol{w} = \boldsymbol{0}$, update $\boldsymbol{w}$ at each iteration k as follows:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta(k) \sum_{i \in I_M^k} \boldsymbol{x}_i y_i$$

Batch learning vs. Online learning

- Batch learning
  - All the training samples are available
- Online learning or mini-batch learning
  - The learning algorithm sees the training samples one by one.
- What is the advantages of online (mini-batch) learning?

Perceptron: Mistake Bound Theorem

- Maintains a weight vector $\boldsymbol{w} \in \mathcal{R}^p$, $\boldsymbol{w}_1 = (0, \cdots, 0)$.
- Upon receiving an example $\boldsymbol{x} \in \mathcal{R}^p$
- Predicts according to the linear threshold function $\text{sgn}(\boldsymbol{w}^T \boldsymbol{x})$.

Theorem [Novikoff,1963] *Let* $(\boldsymbol{x}_1, y_1) \cdots (\boldsymbol{x}_t, y_t)$, *be a sequence of labeled examples with* $\boldsymbol{x}_i \in \mathcal{R}^p$, $\|\boldsymbol{x}_i\| \leq R$ *and* $y_i \in \{-1,1\}$ *for all i. Let* $\boldsymbol{u} \in \mathcal{R}^p$, $\gamma > 0$ *be such that* $y_i \boldsymbol{u}^T \boldsymbol{x}_i \geq \gamma$ *for all i. Then Perceptron makes at most* $\frac{\|\boldsymbol{u}\|^2 R^2}{\gamma^2}$ *mistakes on this example sequence.*

Margin Complexity Parameter

$$\frac{\max_i \|\boldsymbol{x}_i\|^2 \, \|\boldsymbol{u}\|^2}{\min_i (y_i \boldsymbol{u}^T \boldsymbol{x}_i)^2}$$

Perceptron-Mistake Bound

Proof: Let $\boldsymbol{u}$ be any solution vector, so that $y_i \boldsymbol{u}^T \boldsymbol{x}_i$ is strictly positive for all i, and let α be a positive scale factor.

$$\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u} = (\boldsymbol{w}_k - \alpha \boldsymbol{u}) + y_k \boldsymbol{x}_k$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 = \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 + 2(\boldsymbol{w}_k - \alpha \boldsymbol{u})^T y_k \boldsymbol{x}_k + \|\boldsymbol{x}_k\|^2$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \leq \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 - 2\alpha \boldsymbol{u}^T y_k \boldsymbol{x}_k + \|\boldsymbol{x}_k\|^2$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \leq \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 - 2\alpha\gamma + R^2$$

Let $\alpha = \frac{R^2}{\gamma}$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \leq \|\boldsymbol{w}_1 - \alpha \boldsymbol{u}\|^2 - kR^2$$

Assumptions

$y_i \boldsymbol{u}^T \boldsymbol{x}_i \geq \gamma$

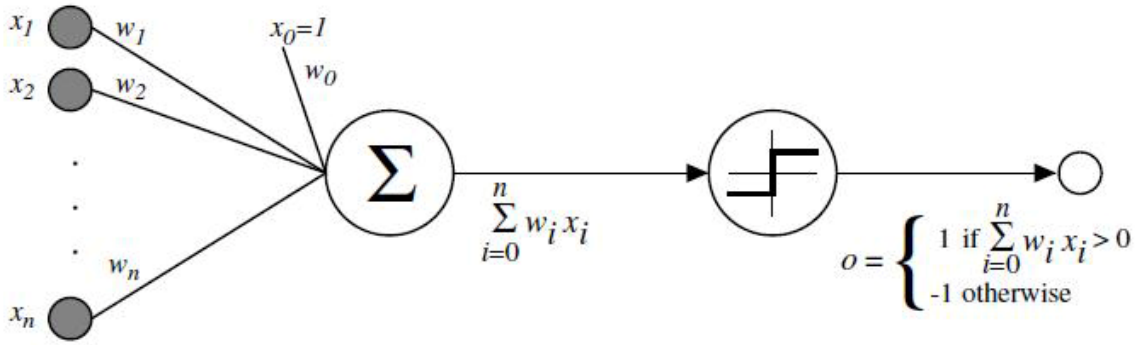
$\|\boldsymbol{x}_i\| \leq R$

$\boldsymbol{w}_1 = (0, \cdots, 0)$

$$k_{max} = \frac{\|\boldsymbol{w}_1 - \alpha \boldsymbol{u}\|^2}{R^2}$$

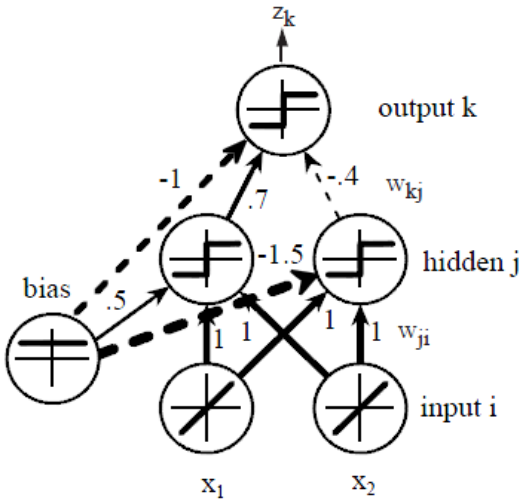
$$k_{max} = \frac{\alpha^2 \|\boldsymbol{u}\|^2}{R^2} = \frac{\|\boldsymbol{u}\|^2 R^2}{\gamma^2}$$

Perceptron vs. Multi-layer Perceptron



Gradient descent

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta(k) \sum_{i \in I_M^k} \boldsymbol{x}_i y_i$$



How to Learn?

Backpropagation

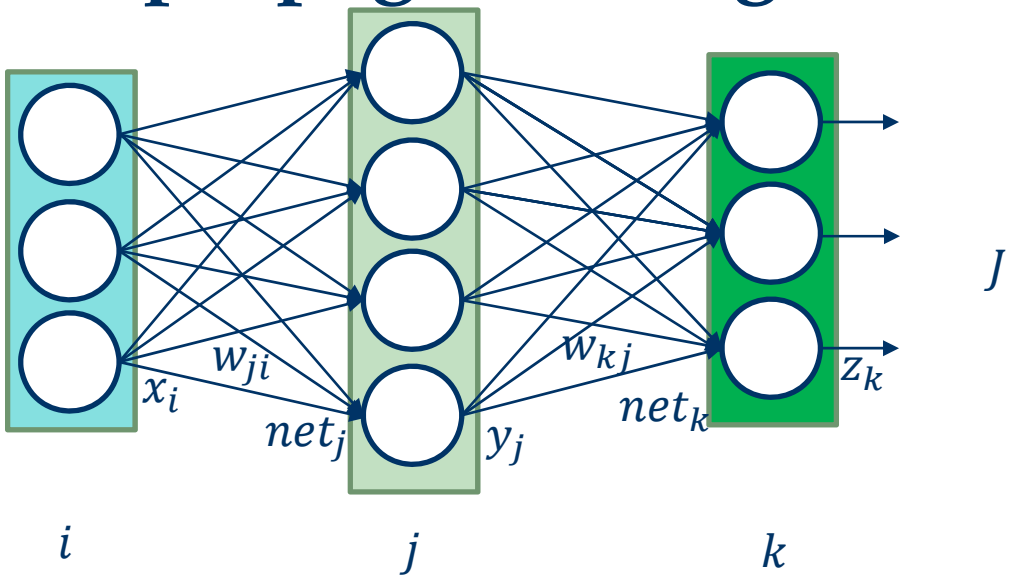
Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by "Perceptrons"
- 1970s, Backpropagation was introduced and wasn't valued
  - Paul J. Werbos (born 1947)
  - *Paul Werbos (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University.*
  - "Mentioned the possibility of applying this principle to artificial neural network"
- 1986, Backpropagation was reinvented
  - *Rumelhart, David E.; **Hinton, Geoffrey E.**; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088): 533–536.*
  - "Showed through computer experiments that this method can generate useful internal representations of incoming data in hidden layers of neural networks"

BP & Geoffrey Hinton

- In 1970, when the first AI winter came, **Geoffrey Hinton**, a 23-years-old young man, gained his bachelor's degree of experimental psychology from Cambridge.
- And then, he chose to get a PhD at the University of Edinburgh, on AI, nearly if not exactly equal to NN at that time.
- His friends were confused about his decision and thought he must be crazy, because NN had been proved to be sheer nonsense, by the book "*Perceptrons*".

Backpropagation Algorithm



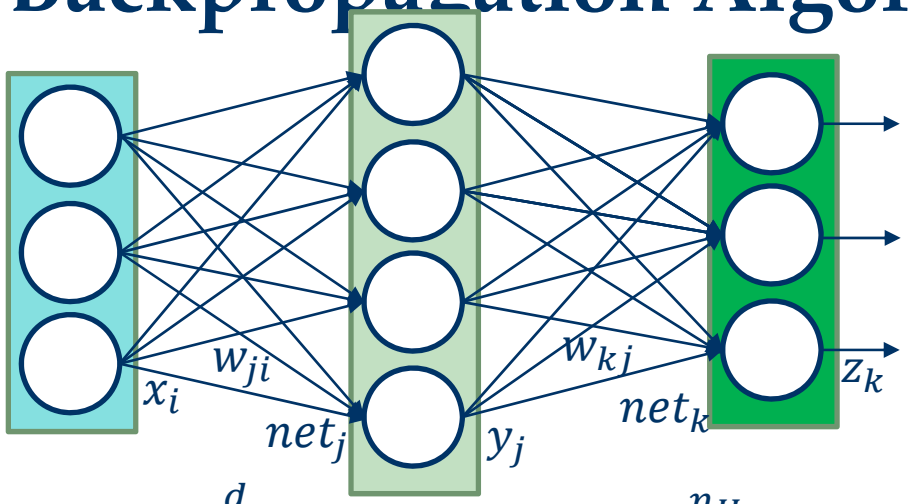
$$\Delta \boldsymbol{w} = -\eta \nabla J = -\eta \frac{\partial J}{\partial \boldsymbol{w}} \qquad \Delta w_{mn} = -\eta \frac{\partial J}{\partial w_{mn}}$$

- Where η is the learning rate which indicates the relative size of the change in weights

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t)}$$

- where t indexes the particular pattern presentation

Backpropagation Algorithm



$$J = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2$$

$$net_j = \sum_{j=0}^{d} x_i w_{ji} \qquad net_k = \sum_{j=0}^{n_H} y_j w_{kj}$$

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k}\cdot\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k)\cdot f'(net_k)\cdot y_j$$

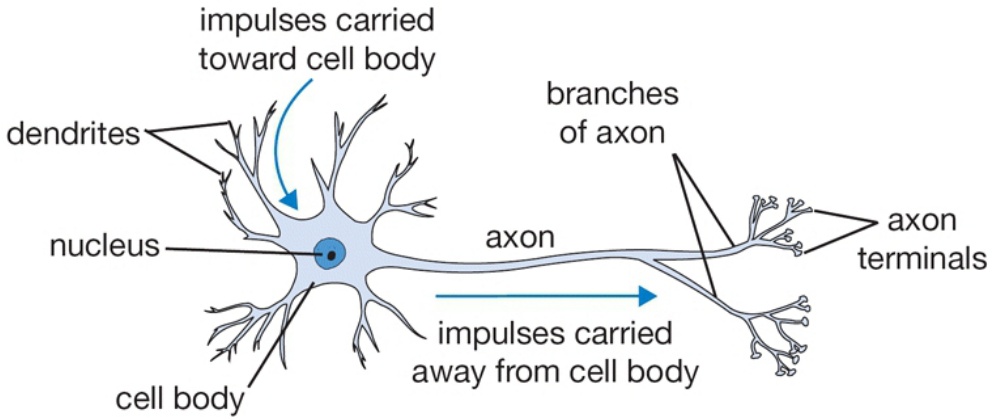
$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

$$= \left(\sum_{k=1}^{c}(z_k - t_k)\cdot f'(net_k)\cdot w_{kj}\right)\cdot f'(net_j)\cdot x_i$$

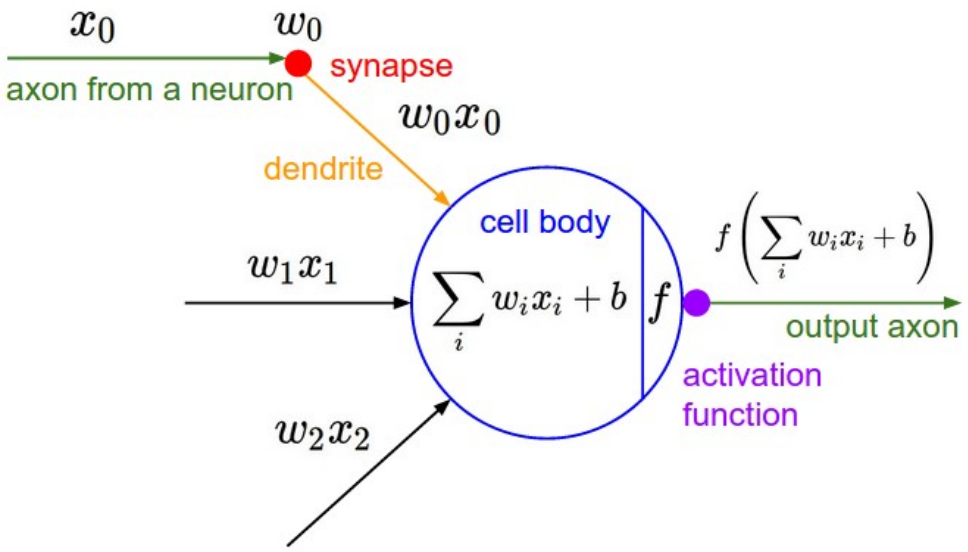
Natural Neural Net Models

- Human brain consists of very large number of neurons (between 10^{10} to 10^{12})
- No. of interconnections per neuron is between 1K to 10K
- Total number of interconnections is about 10^{14}
- Damage to a few neurons or synapse (links) does not appear to impair overall performance significantly (robustness)

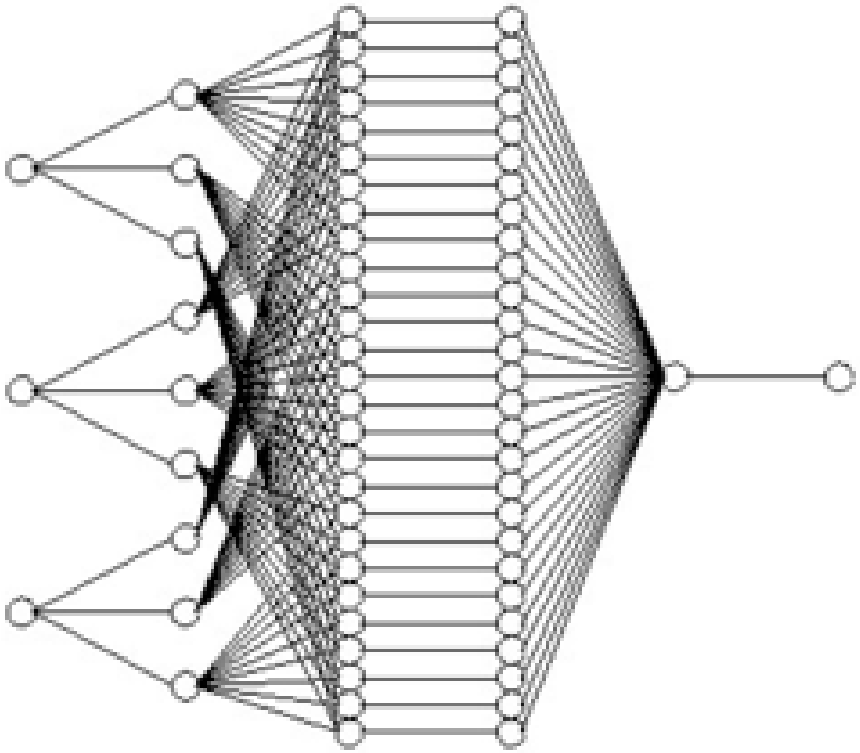
The Artificial Neural Network



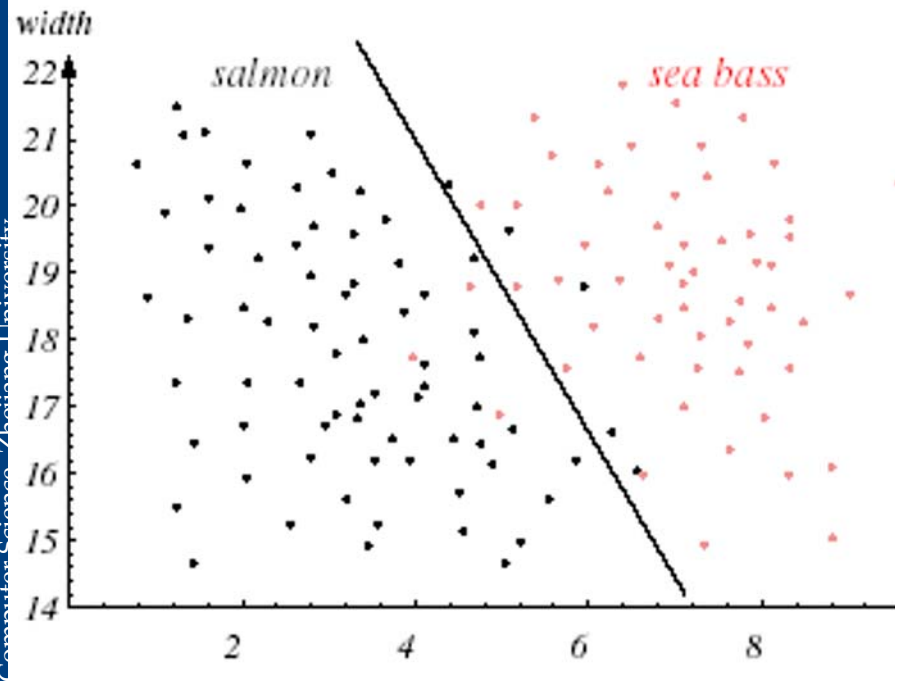
- A cartoon drawing of a biological neuron

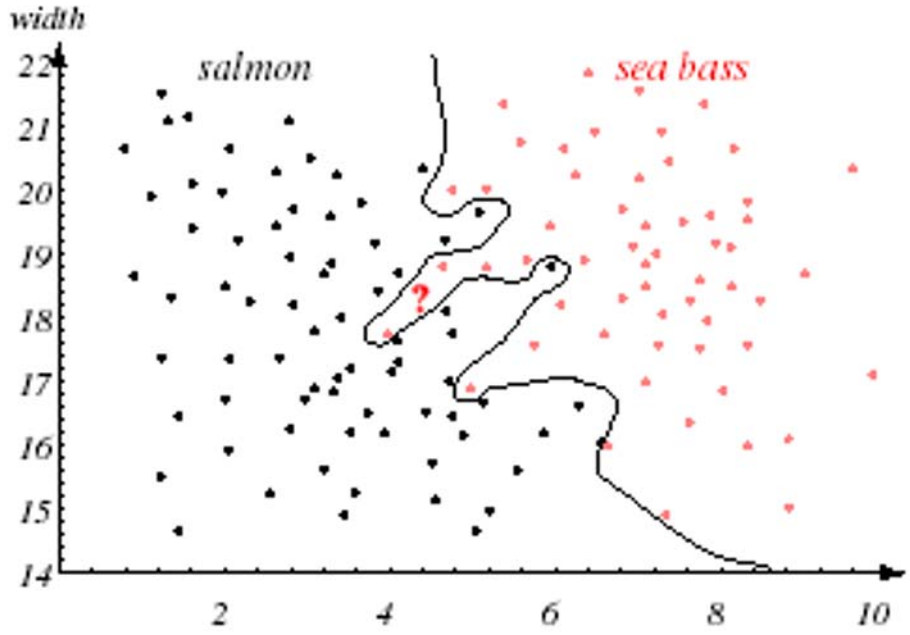


The Artificial Neural Network (Deep Learning)



Linear (simple) decision boundary VS. Complex decision boundary





- Which one is better?

Bias-variance Decomposition

- $\mathrm{EPE}(f) = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$
- Expected prediction error (expected loss) =

  (bias)^2 + variance + noise

- (bias)^2:

$$\int \{E_D(f(\boldsymbol{x}; D)) - E(y|\boldsymbol{x})\}^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

- variance:

$$\int E_D\left\{[f(\boldsymbol{x}; D) - E_D(f(\boldsymbol{x}; D))]^2\right\} p(\boldsymbol{x}) d\boldsymbol{x}$$

- noise:

$$\int \mathrm{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

General Feedforward Operation

- Case of c output units

$$g_k(\boldsymbol{x}) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

$$k = 1, \cdots, c$$

- Hidden units enable us to express more complicated nonlinear functions and extend classification capability
- Assume for now that all activation functions are identical
- Question: Can every decision boundary be implemented by a three-layer network described by the above equation?

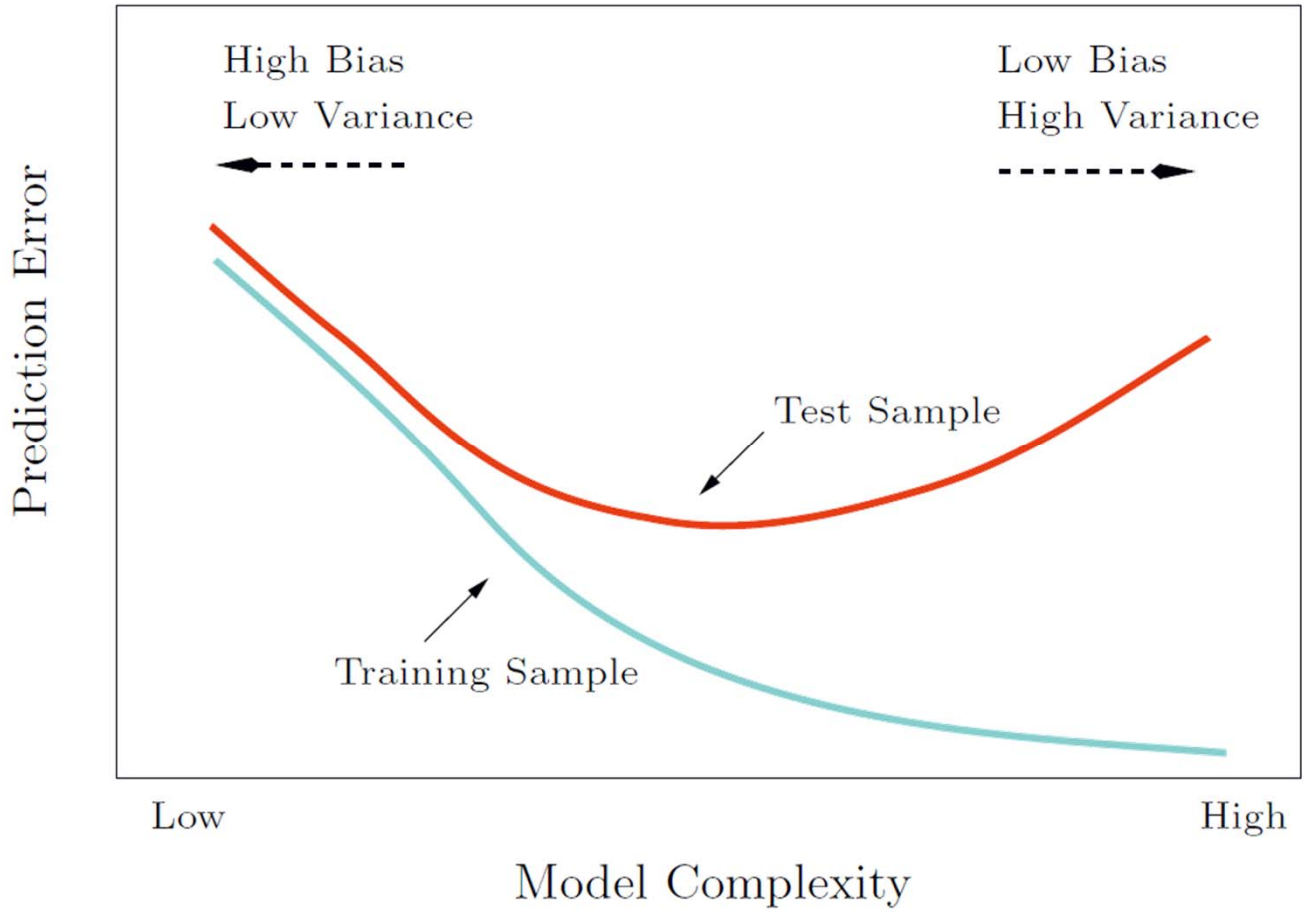
Expressive Power of Multi-layer Networks

- Answer: Yes (due to A. Kolmogorov)
  - Any continuous function from input to output can be implemented in a three-layer net, given sufficient number of hidden units n_H, proper nonlinearities, and weights.

- Any continuous function $g(\boldsymbol{x})$ defined on the unit hypercube $I^n (I = [0,1] \text{ and } n \geq 2)$ can be represented in the following form:

$$g(\boldsymbol{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \Phi_{ij}(x_i) \right)$$

for properly chosen functions Ξ_j and Φ_{ij}



Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules

Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules
    - Backpropagation

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by “Perceptrons”
- 1970s, Backpropagation was introduced and wasn’t valued
- 1986, Backpropagation was reinvented
- 1989, **Convolutional Neural Network (CNN)** by Yann LeCun
- 1980s, Recurrent Neural Network was created
- 1997, **Long Short Term Memory networks(LSTM)** by Hochreiter & Schmidhuber

Yann Lecun

- **Yann Lecun**
- Burn in Paris in 1960.
- In 1987, after receiving PhD in France, he worked as a postdoctoral researcher with **Hinton** for one year, and then in Bell Labs.

BP & CNN

- In 1989, **Yann Lecun** published the paper "*Backprop-agation applied to handwritten zip code recognition*".

- In the paper, he trained a CNN model with about 10,000 hand-written-digit images provided by the US Postal System and achieved an test error rate of 5%.

- A CNN-based commercial software was also developed by Lecun to recognize the hand-written digits on checks. It had occupied about 20% of the market share in the late 1990s US.

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by “Perceptrons”
- 1970s, Backpropagation was introduced and wasn’t valued
- 1986, Backpropagation was reinvented
- 1989, **Convolutional Neural Network (CNN)** by Yann LeCun
- 1980s, Recurrent Neural Network was created
- 1997, **Long Short Term Memory networks(LSTM)** by Hochreiter & Schmidhuber
- For about 20 years, the second winter brought by SVM

The second winter

- However, at the time when CNN was ready to enjoy its booming, a researcher in Bell Labs, whose office was quite near to Yann Lecun's, brought the second winter to NN research.

- **Vladmir Vapnik**
- Burn in the former Soviet Union in 1936
- Migrated to US in 1990 and worked in **Bell Labs**.
- Early in 1963, Vapnik invented the **Support Vector Machine (SVM)** algorithm.

The second winter

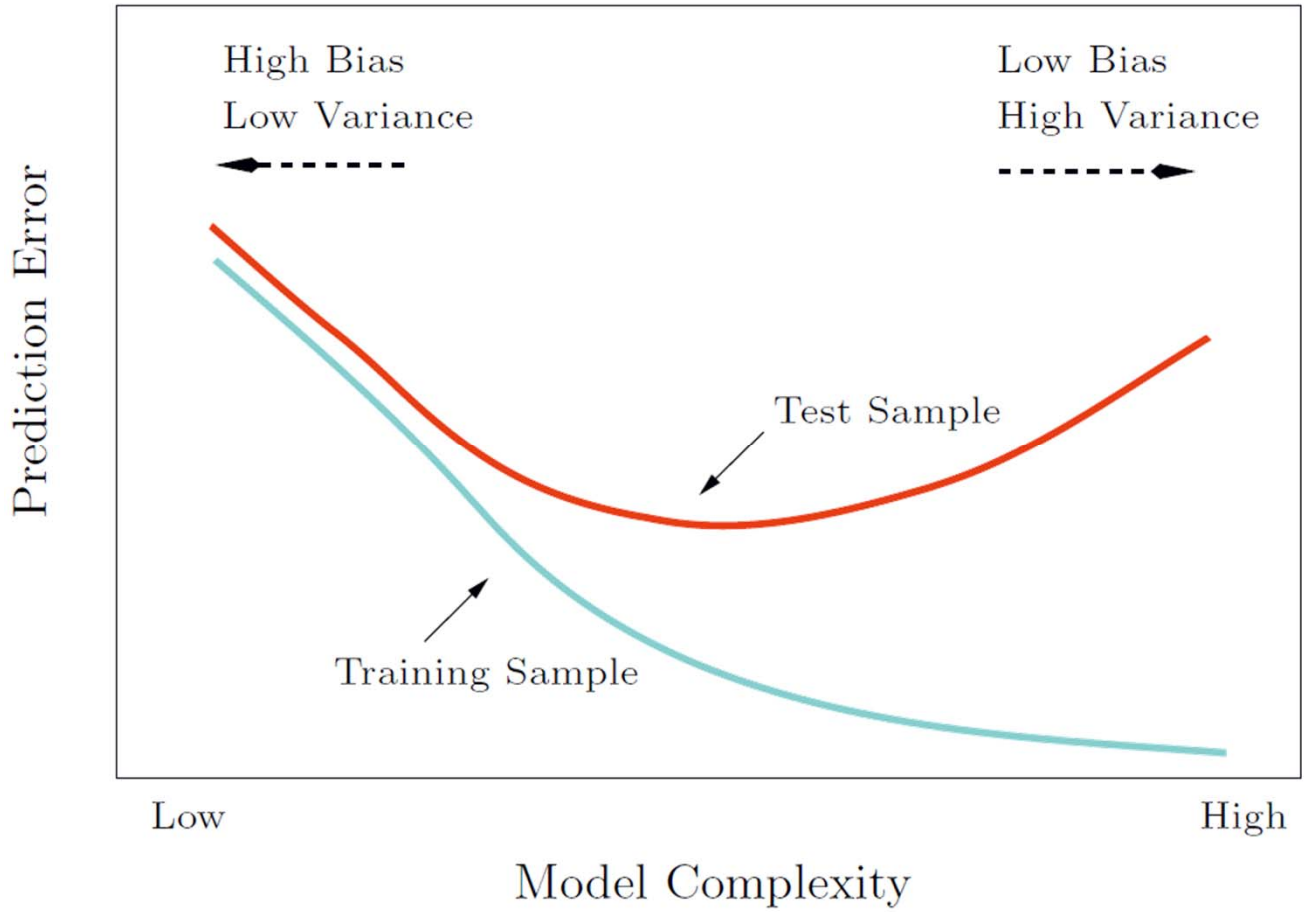
- The SVM, as an exquisite classification algorithm, started to find its prosperity in image and voice recognition, in the early 1990s.
- In Bell Labs on the corridor, **Yann Lecun and Vapnik** often had fever discussions on the strength and weakness of NN and SVM.
- On the recognition task of hand-written digits, the SVM had kept making progress:
  - By 1998, the error rate had been reduced to 0.8% .
  - By 2002, 0.56%
- It had obviously exceeded the contemporary performances of NN.

The second winter

Geoff Hinton sits at the head of the table at the 2003 Vancouver workshop, where he and his fellow academics convinced the Canadian Institute for Advanced Research (CIFAR) to agree to fund their work.

The second winter

- In 2004, Canadian Institute for Advanced Research (CIFAR) agreed to support the NN research by giving about $10 million over 10 years.
- By then, **CIFAR** was **the only organization** funding the research of NN.
- **Without the support of CIFAR, the AI research of human beings may have to stagger in darkness for many more years.**



Something about ‘deep’ and ‘learning’

- For neural networks, being deeper means being more powerful and expressive.

- So, why not going deeper at the very beginning?
  - The vanishing gradient problem.
  - Expensive computation budget.

- The advances in handling these problems witnessed by the last decade are the prerequisites for the popularization of deep neural networks, i.e., deep learning.

Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules

Activation Function

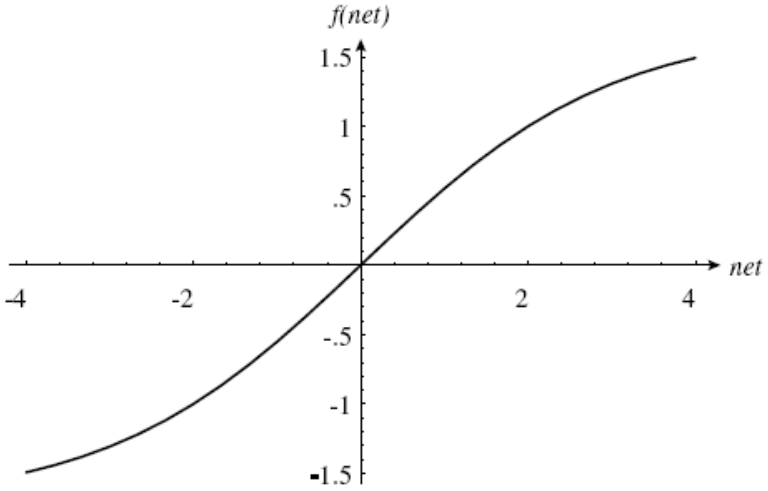
- Activation Function *f*
  - Must be non-linear (otherwise, 3-layer network is just a linear discriminant) and saturate (have max and min value) to keep weights and activation functions bounded
  - Activation function and its derivative must be continuous and smooth; optionally monotonic
  - Choice may depend on the problem. Eg. Gaussian activation if the data comes from a mixture of Gaussians
  - Eg: sigmoid (most popular), polynomial, tanh

- Parameters of activation function (e.g. Sigmoid)
  - Centered at 0, odd function f(-net) = -f(net) (anti-symmetric); leads to faster learning
  - Depend on the range of the input values

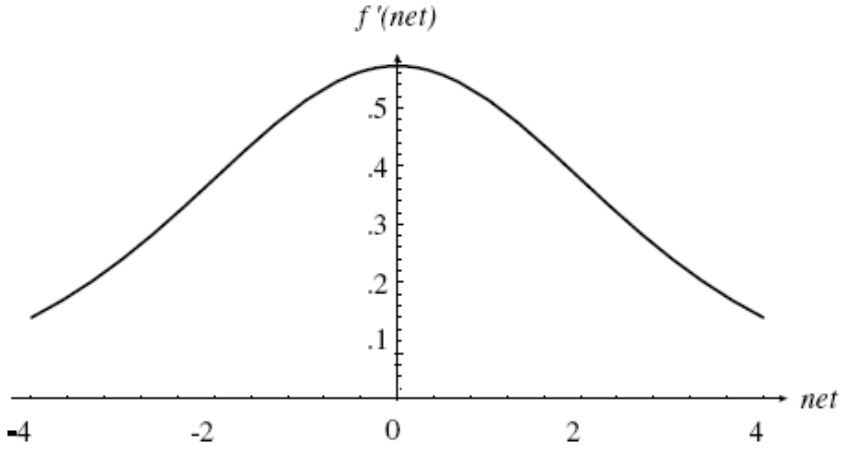
Activation Function

$$f(net) = a\ \tanh(b\ net) = a\left[\frac{1 - e^{b\ net}}{1 + e^{b\ net}}\right] = \frac{2a}{1 + e^{-b\ net}} - a$$

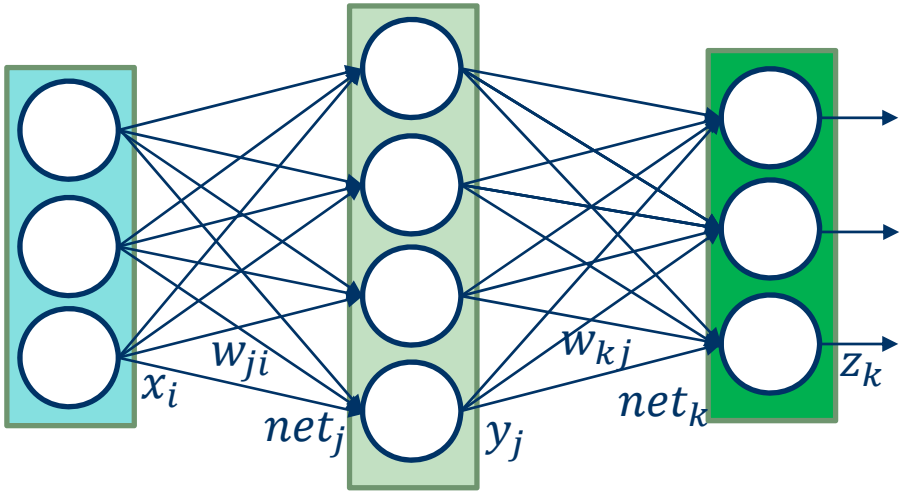
The anti-symmetric sigmoid function:
f(-x) = -f(x).
a = 1.716, b = 2/3.



First order derivative



Backpropagation Algorithm

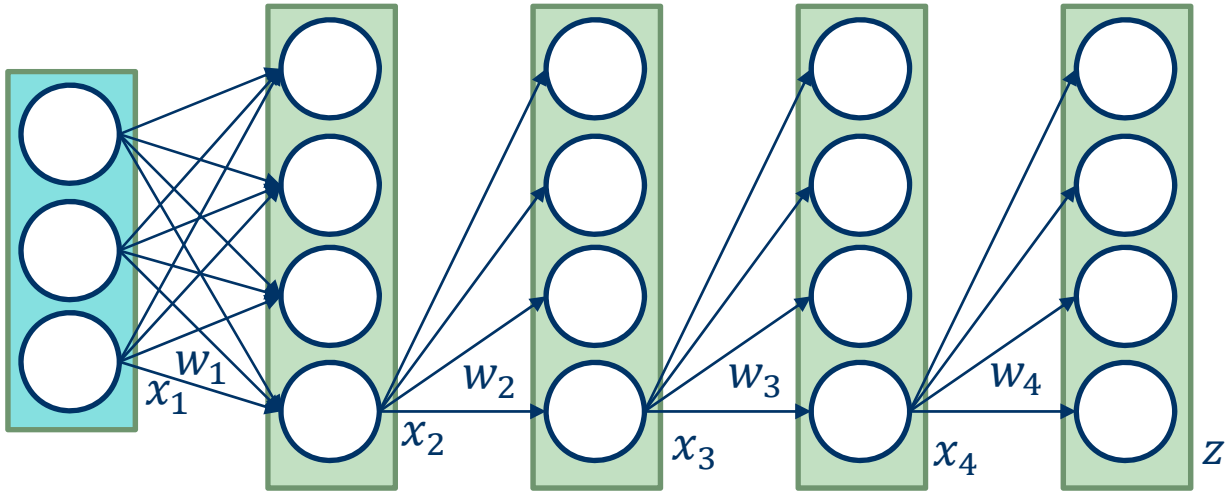


$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k) \cdot f'(net_k) \cdot y_j$$

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \left(\sum_{k=1}^{c} (z_k - t_k) \cdot f'(net_k) \cdot w_{kj} \right) \cdot f'(net_j) \cdot x_i$$

Backpropagation Algorithm



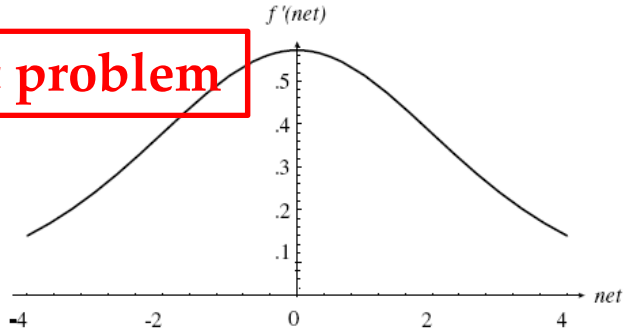
vanishing gradient problem

$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot x_4$$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot x_3$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot w_3 f'(net_2) \cdot x_2$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z} \cdot \underbrace{f'(net_4) \cdot w_4}_{<1} \cdot \underbrace{f'(net_3) \cdot w_3}_{<1} \underbrace{f'(net_2) \cdot w_2}_{<1} \underbrace{f'(net_1)}_{<1} \cdot x_1$$

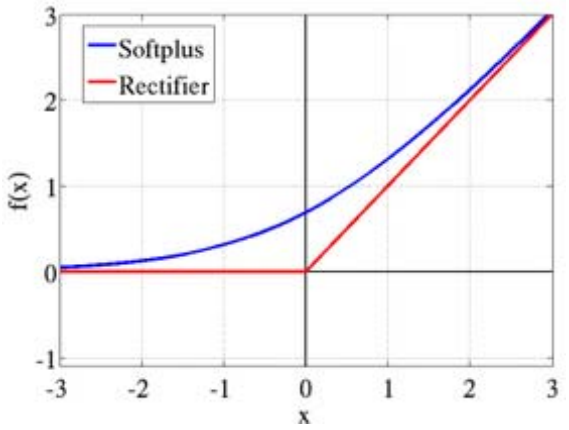


First order derivative

2011, ReLU RevoLUtion

- In 2011, Canadian scholar Xavier Glorot and Yoshua Bengio published the paper "*Deep Sparse Rectifier Neural Networks* ".
- The paper proposed a new activation function, **Rectified Linear Unit (ReLU)**:

$$\mathrm{ReLU}(x) = \max(0, x)$$

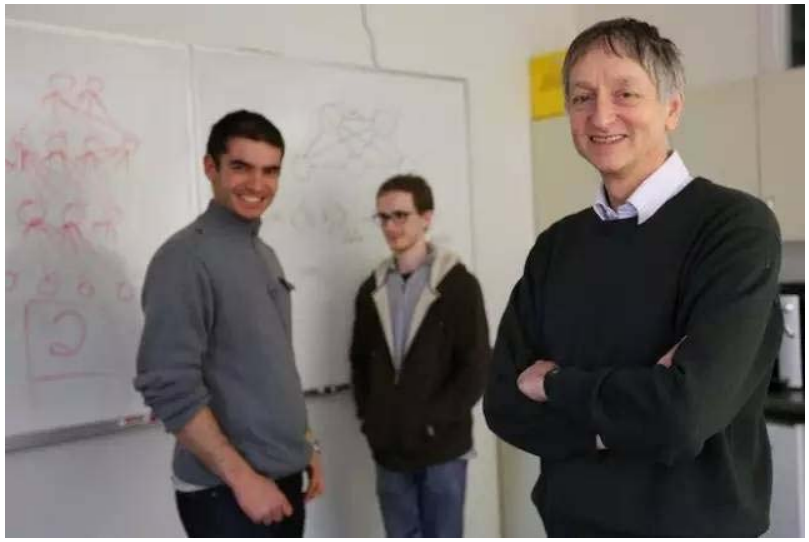


Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by "Perceptrons"
- 1970s, Backpropagation was introduced and wasn't valued
- 1986, Backpropagation was reinvented
- 1989, **Convolutional Neural Network (CNN)** by Yann LeCun
- 1980s, Recurrent Neural Network was created
- 1997, **Long Short Term Memory networks(LSTM)** by Hochreiter & Schmidhuber
- For about 20 years, the second winter brought by SVM
- 2006, DBN pre-training made deep learning becoming hot
- 2007, GPU
- 2009, ImageNet
- 2011, ReLU
- 2012, breakthrough on ImageNet

Break through on ImageNet (2012)

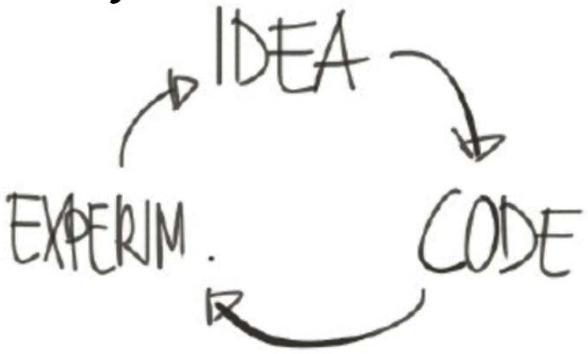
- Hinton's team got their model trained on two NVIDIA GTX 580 GPUs with 1.2 million images for nearly 6 days.
- The top 5-error rate of the trained model was **15.3%**, the champion for sure.
- It's noteworthy that the top-5 error rate of **the second place team using SVM was 25.6%**.



Suskever, Krizhevsky, Hinton, from left to right

What happened during the past 20 years?

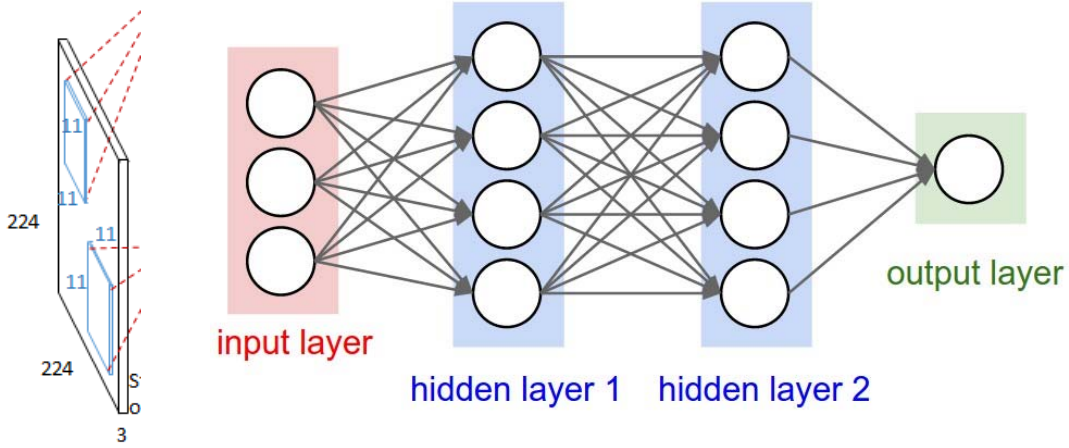
- Huge datasets (labeled) like **ImageNet**
  - 1000 categories, 1,281,167 training data
  - mnist: 10 categories, 60,000 training data
- Smart algorithms and ideas like **ReLU**
- More powerful computers equipped with **GPUs**
  - Research cycle



Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules

Neural Network for Image Input



- The typical images are large, often with lots of pixels.
  - If the input layer has 224×224 units, the first hidden layer has 224×224 units. Then we need $224^4 \approx 2.5 \times 10^9$ parameters.
- How Convolutional Neural Networks (CNN) solve this problem?
  - Weight sharing
  - Images have a strong 2D local structure: pixels that are spatially or temporally nearby are highly correlated.

Convolution

- In mathematics (in particular, functional analysis) convolution is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

- In image processing

5	3	0	4	3	2	3
4	2	5	0	1	5	4
5	4	3	4	3	0	2
2	0	2	5	3	3	5
4	5	3	2	0	4	3
0	4	2	0	3	5	0
3	5	4	1	2	1	4

image

9	3	1
2	0 anchor	5
7	4	4

convolution kernel

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

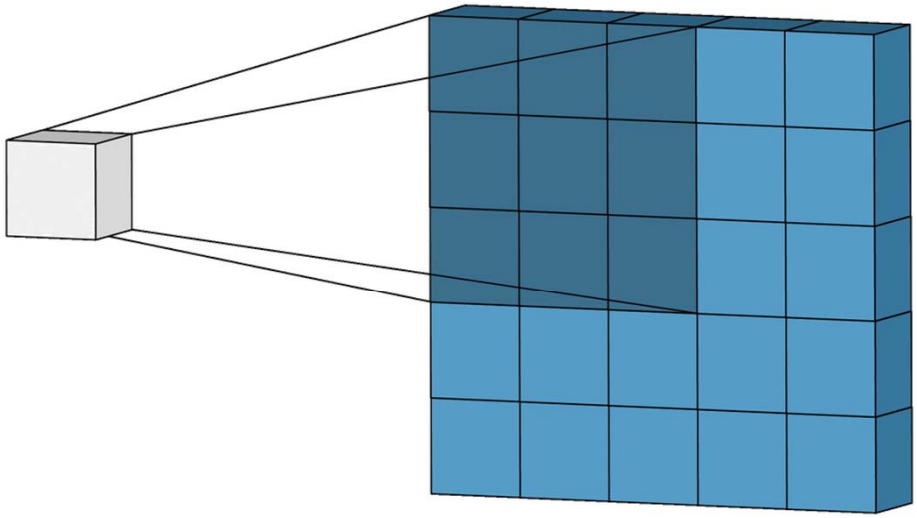
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

The Result of Convolution

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

-1	-1	-1
-1	8	-1
-1	-1	-1

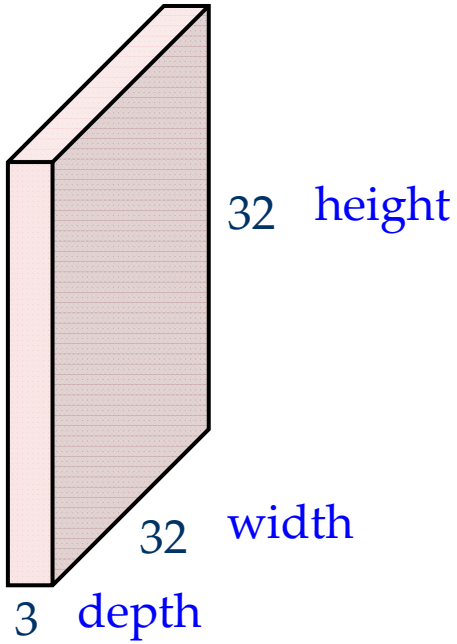
Convolution for Weight Sharing



- 5x5 image, 3x3 hidden layer
  - Full connection: 5x5x3x3 = 225 weights
  - Convolution with one kernel 3x3=9 weights

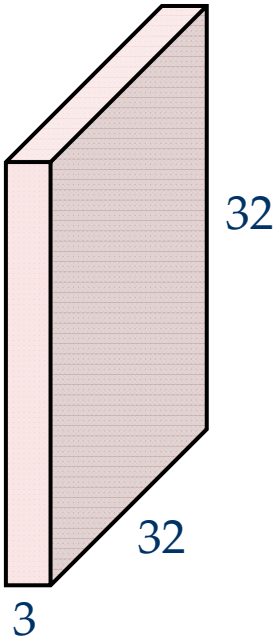
Convolution Layer

32x32x3 image

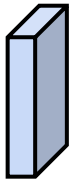


Convolution Layer

32x32x3 image



5x5x3 filter

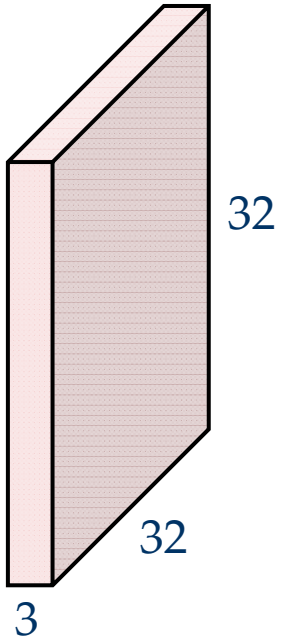


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

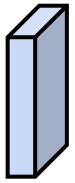
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

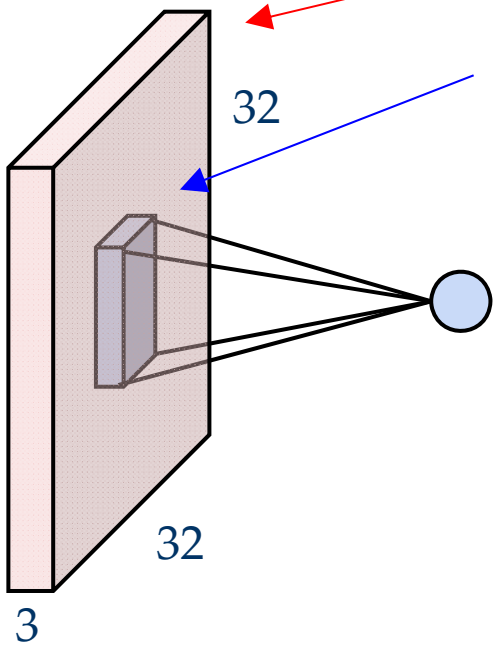


5x5x3 filter



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

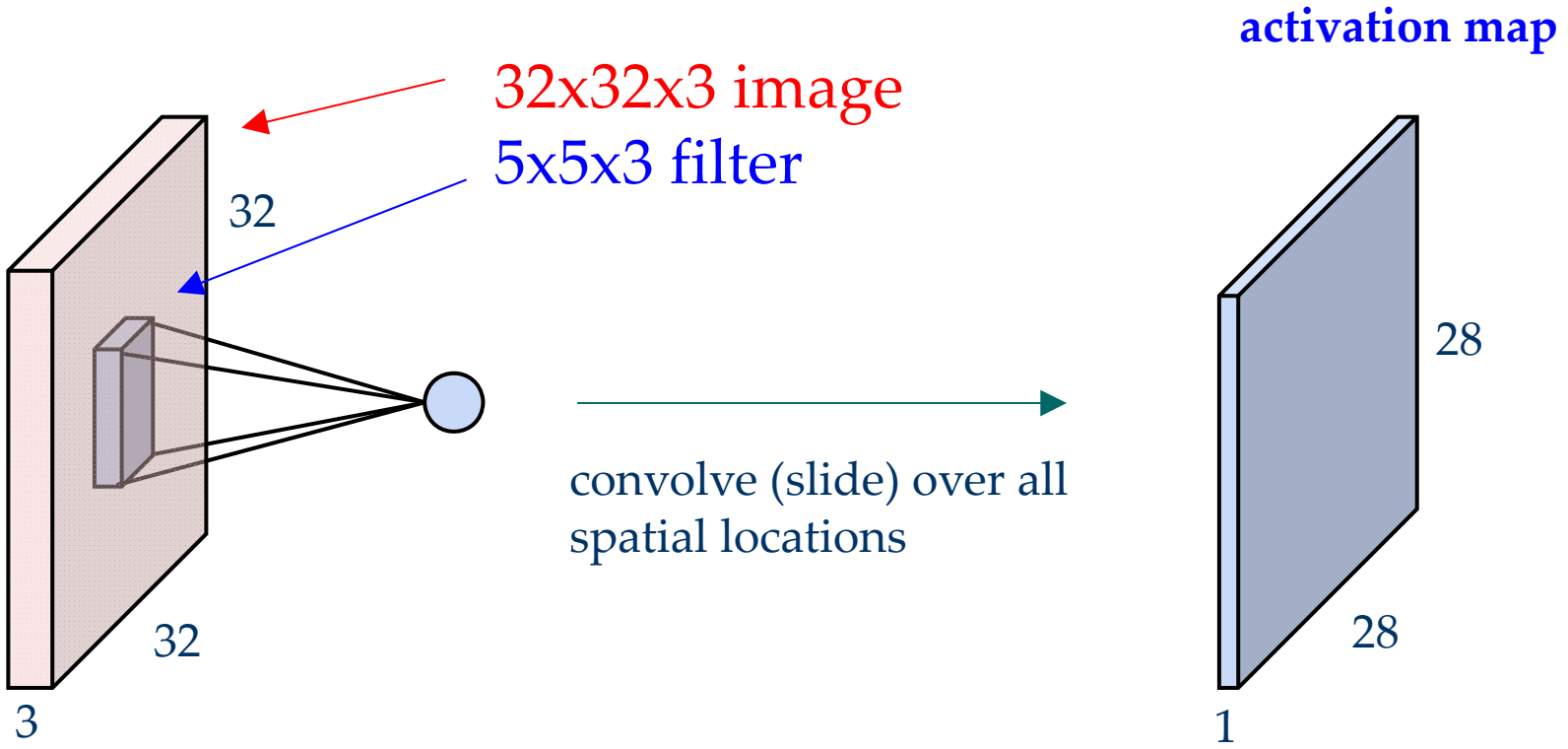
Convolution Layer



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)

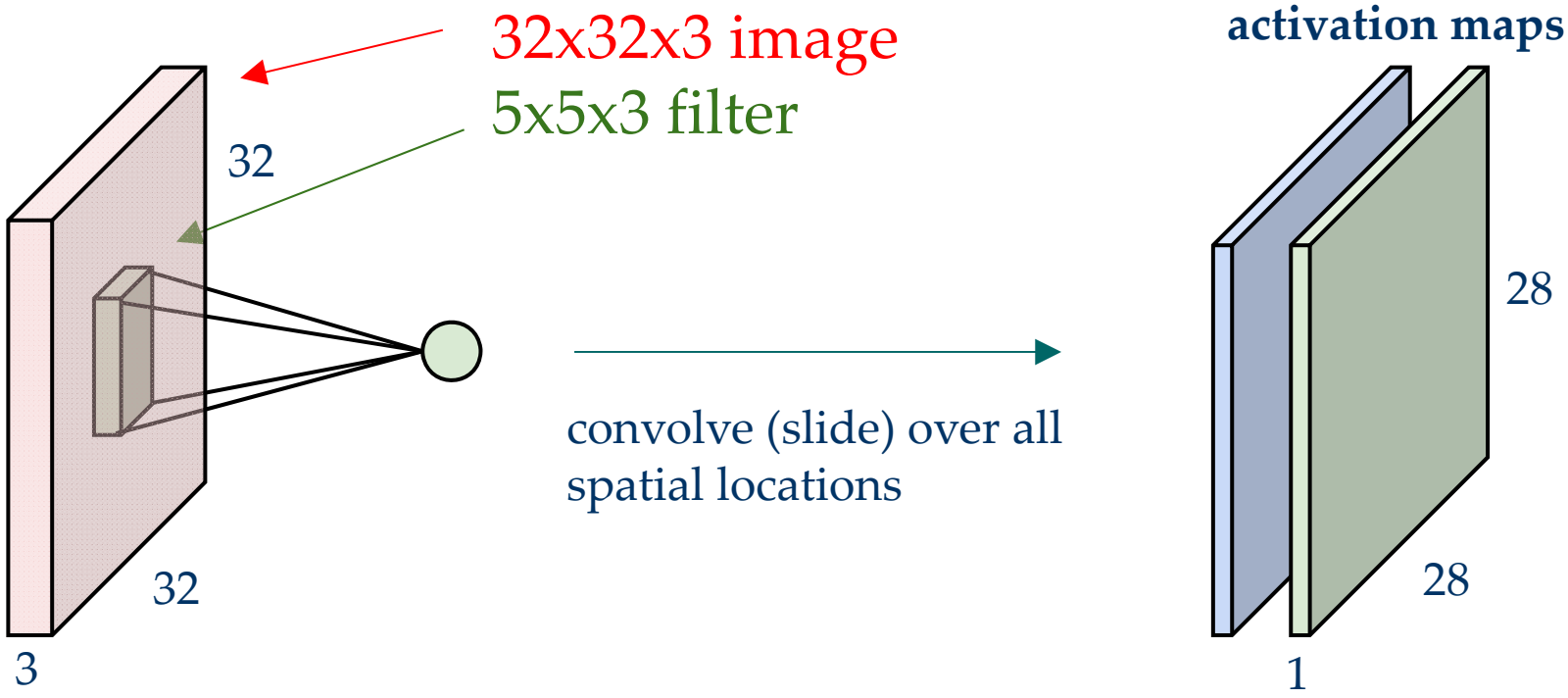
$$w^T x + b$$

Convolution Layer

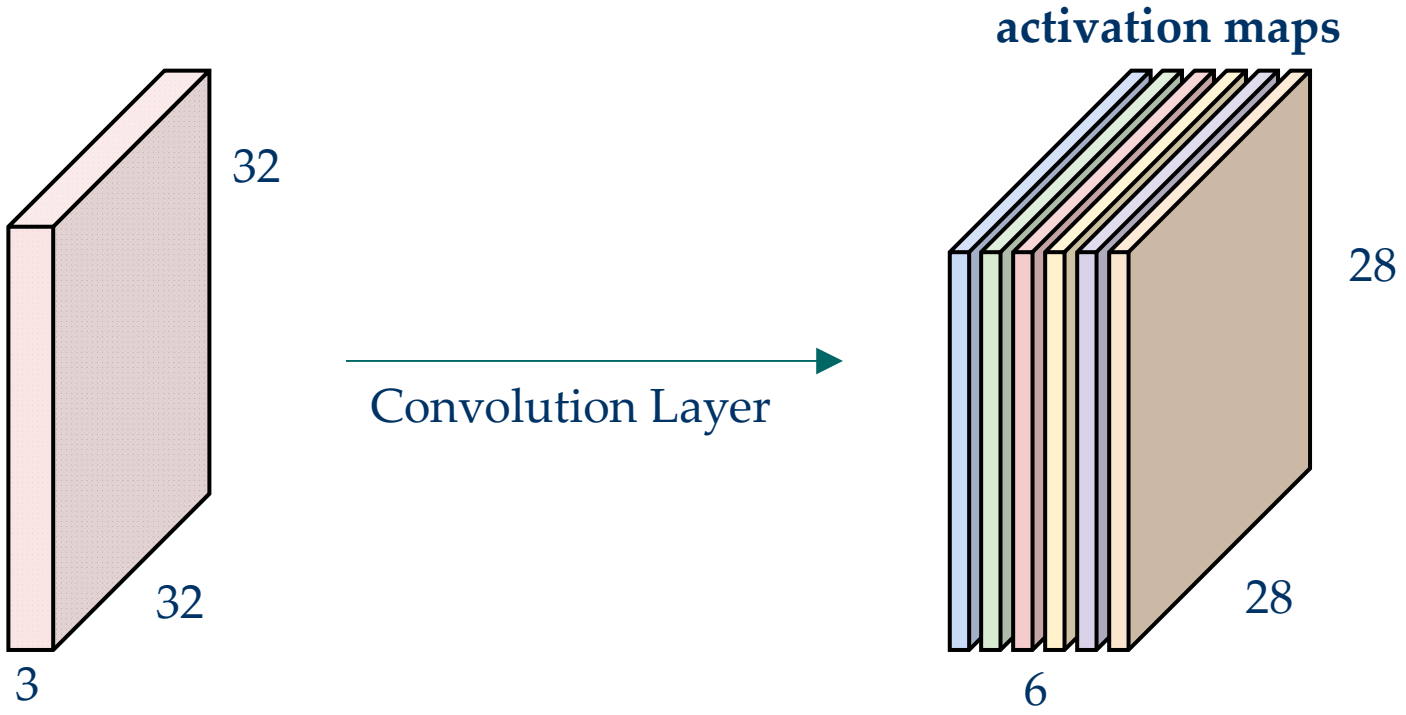


Convolution Layer

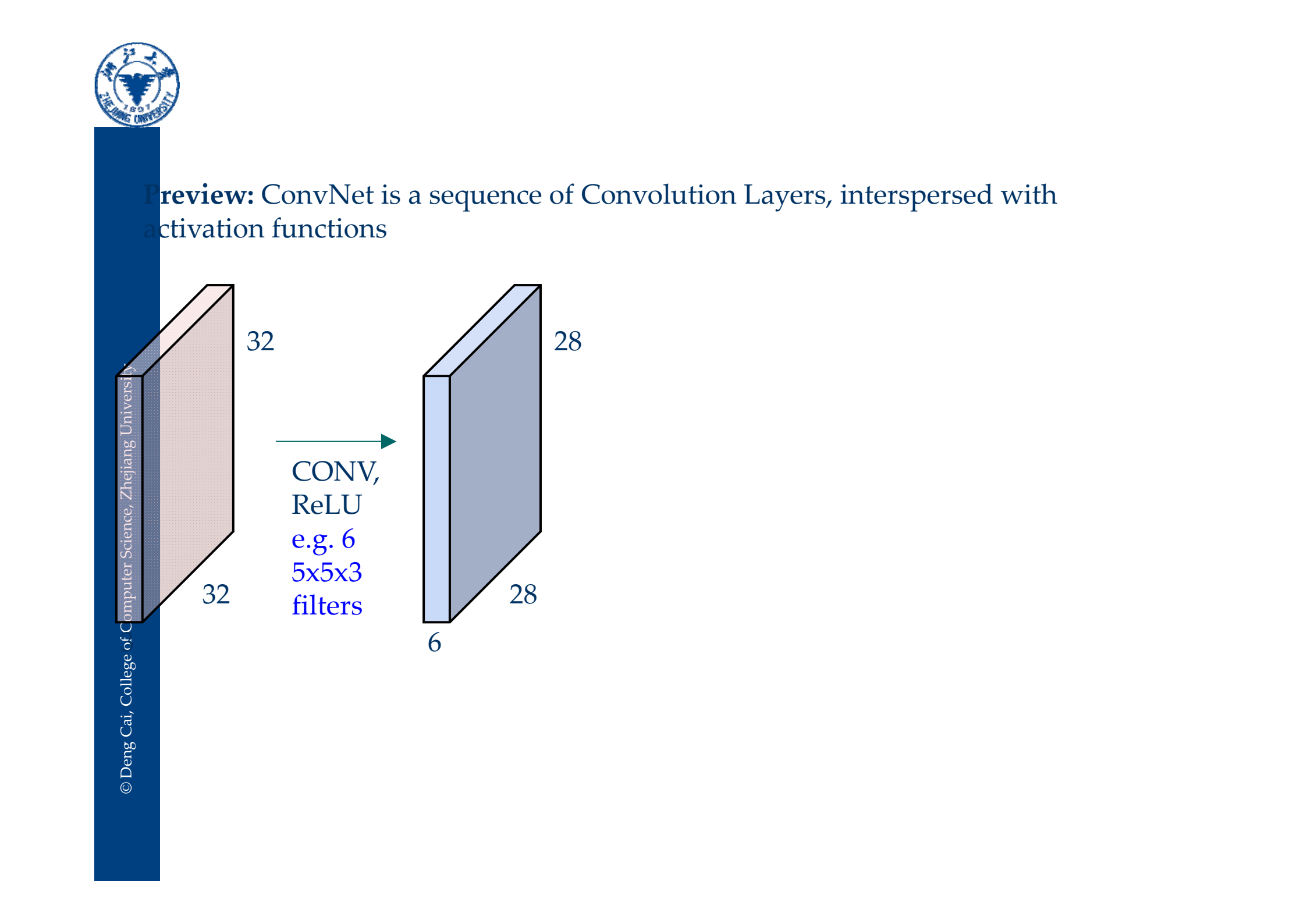
consider a second, green filter



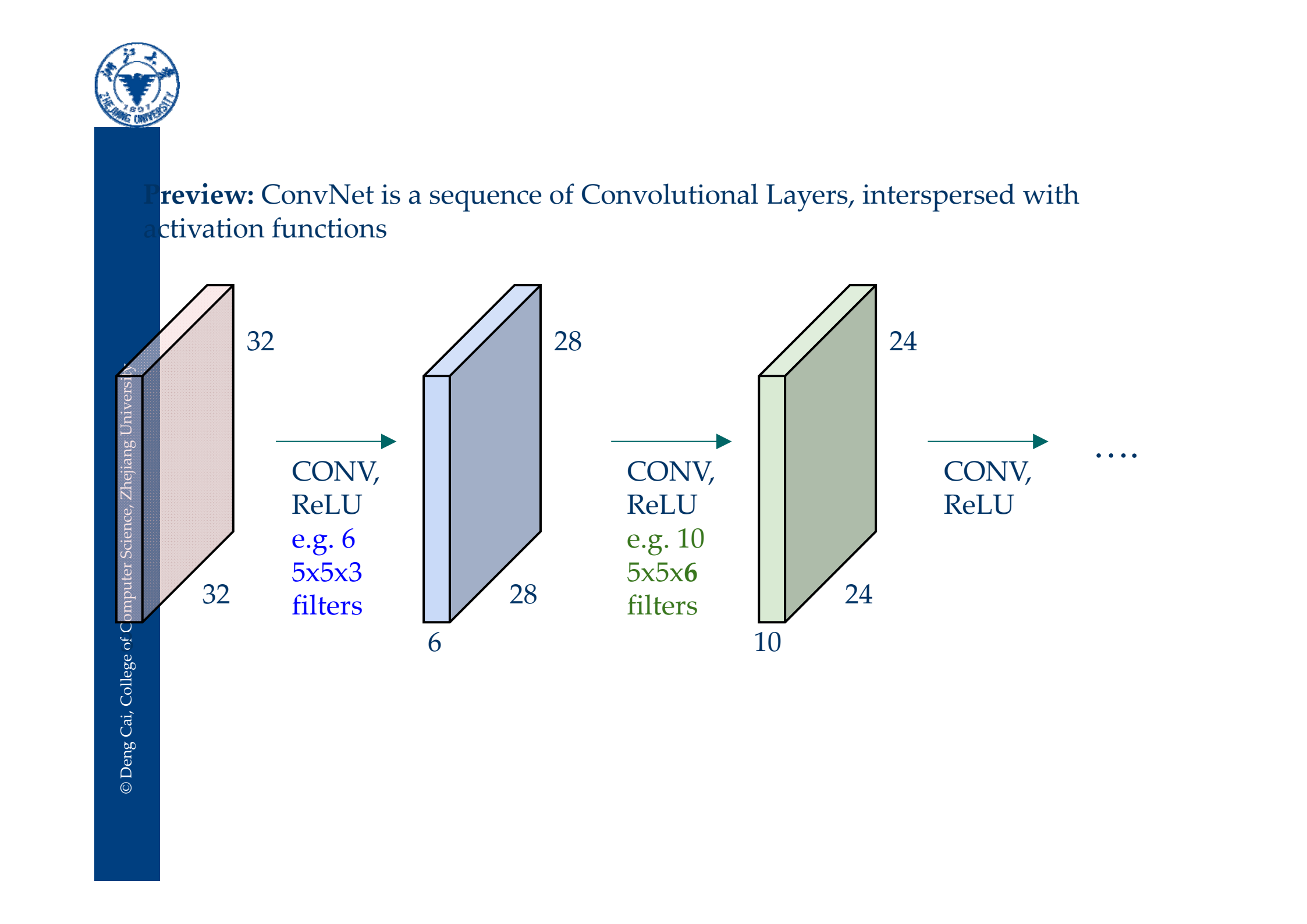
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

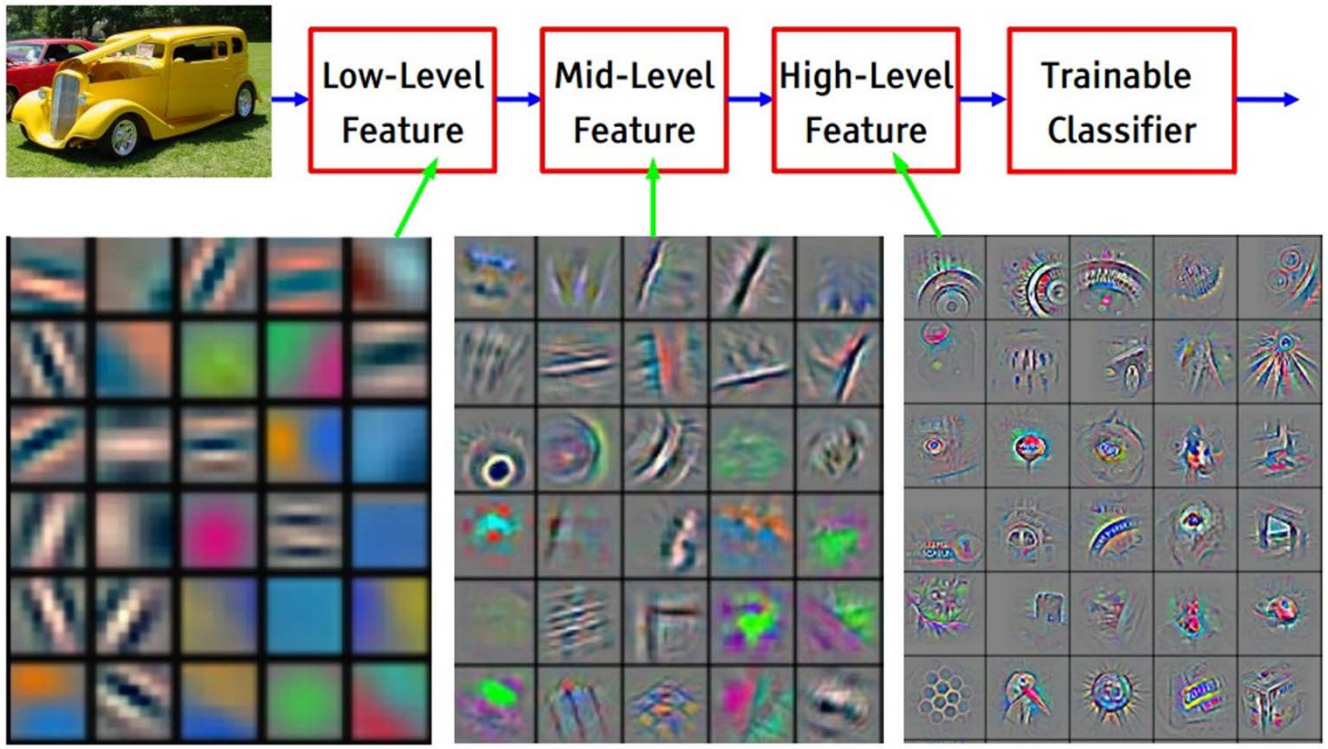


Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Preview

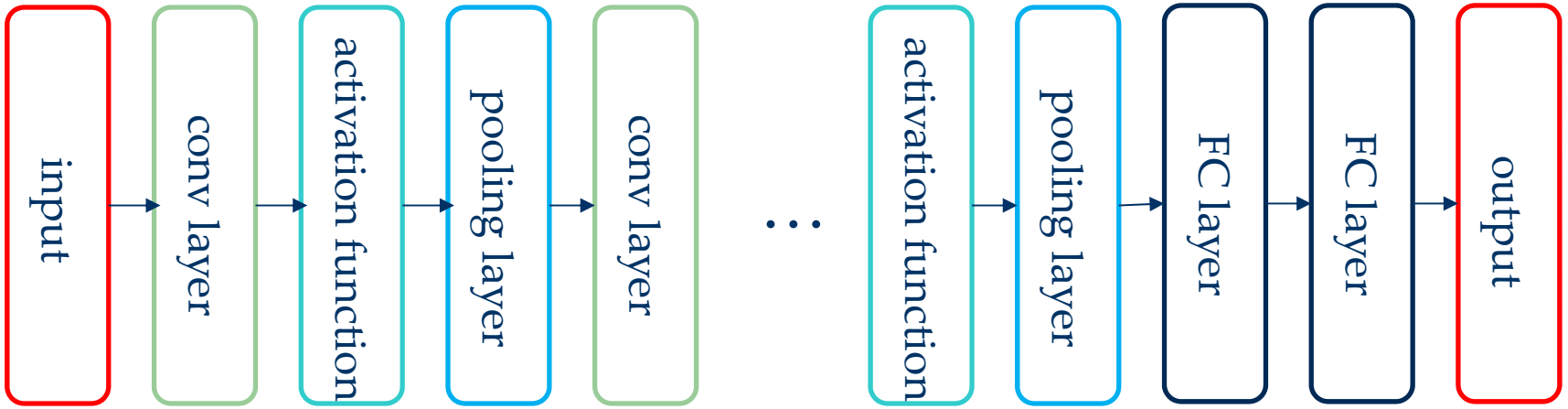
[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNNs/ConvNets

- Architecture Overview



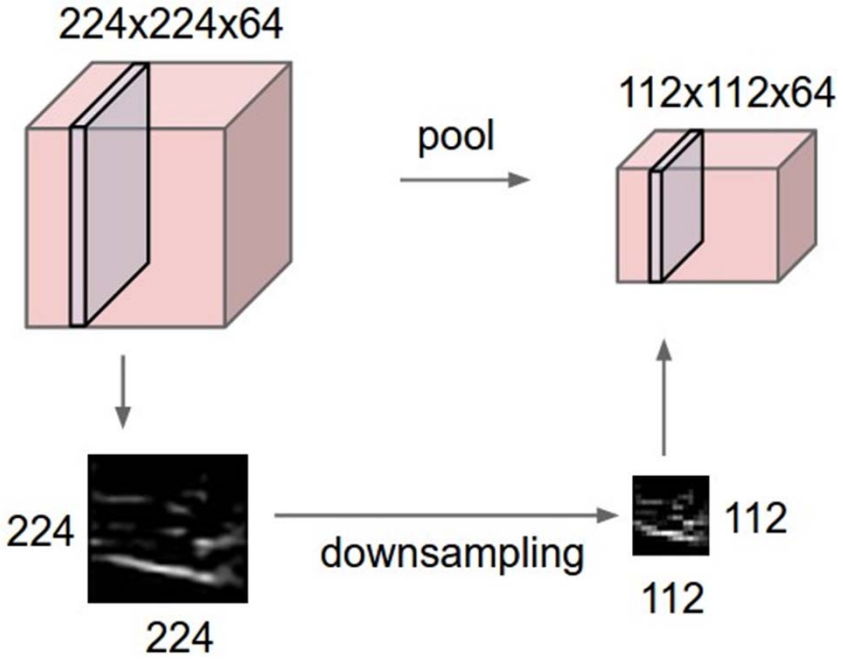
general architecture of ConvNets

Sometimes, there is no pooling layer or FC layer.

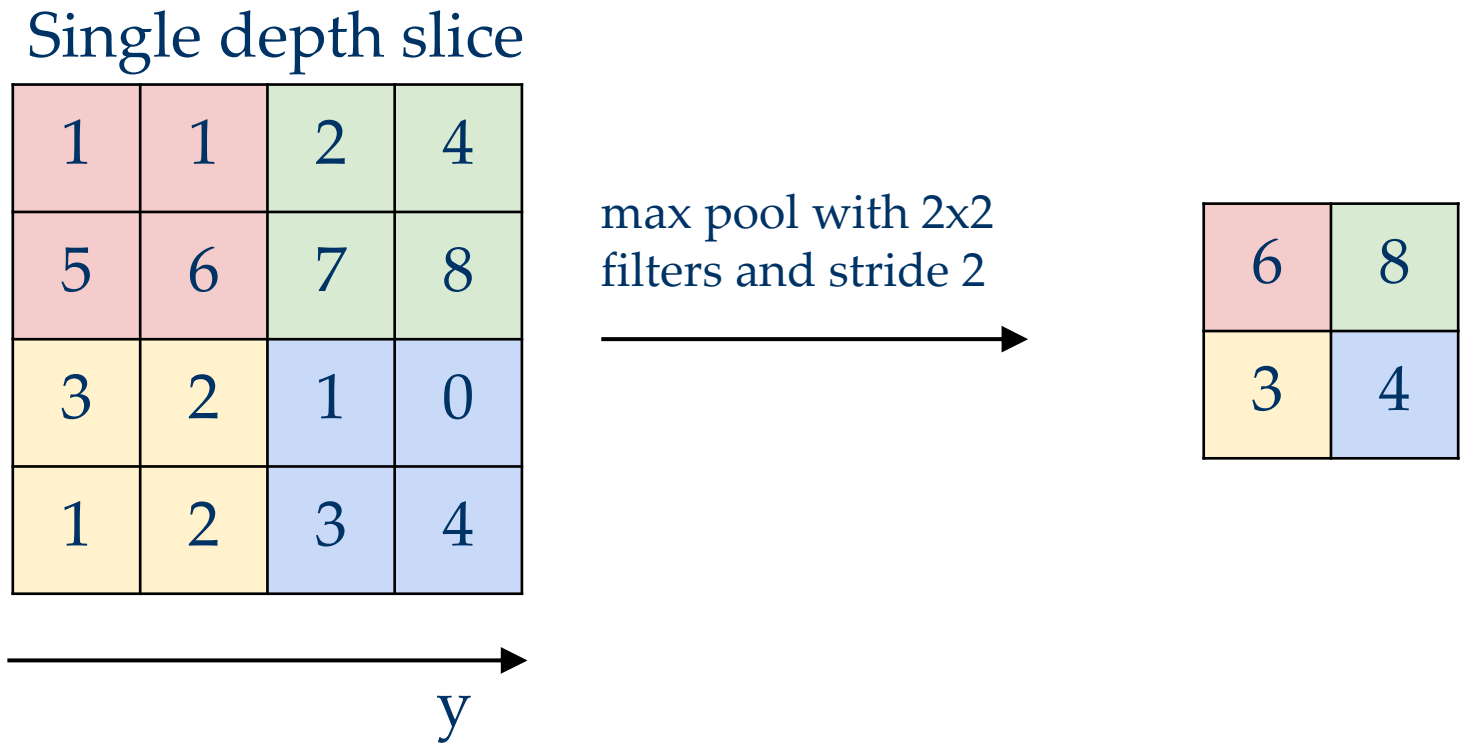
> *"There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers."*

Pooling layer

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

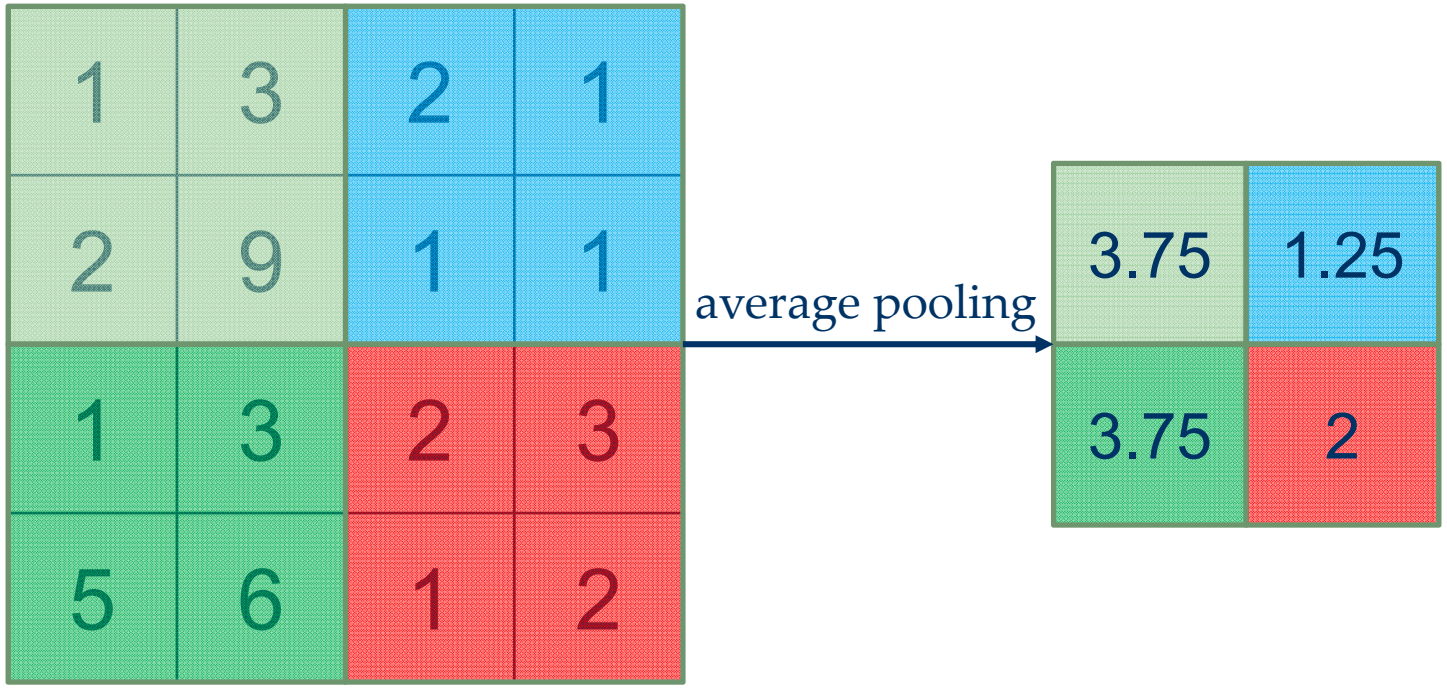


MAX POOLING



Pooling Layer

- **average pooling**



input : 4 × 4, pooling window : 2 × 2, stride : 2, output : 2 × 2

Average pooling caculate the average value of the units in the pooling window.

Case studies

- **LeNet-5**

The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's.

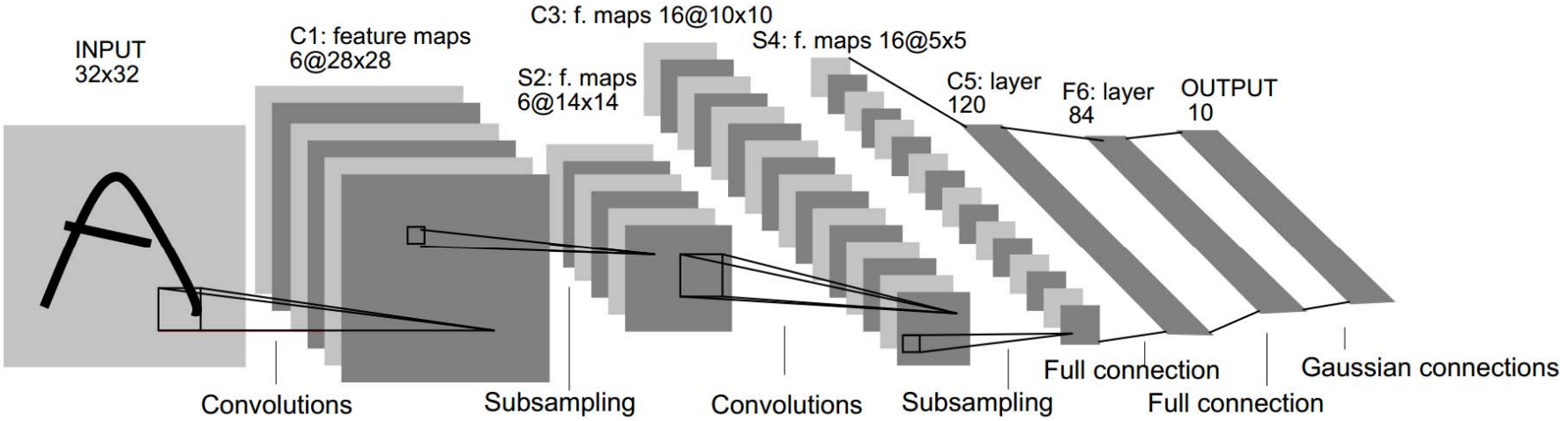


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

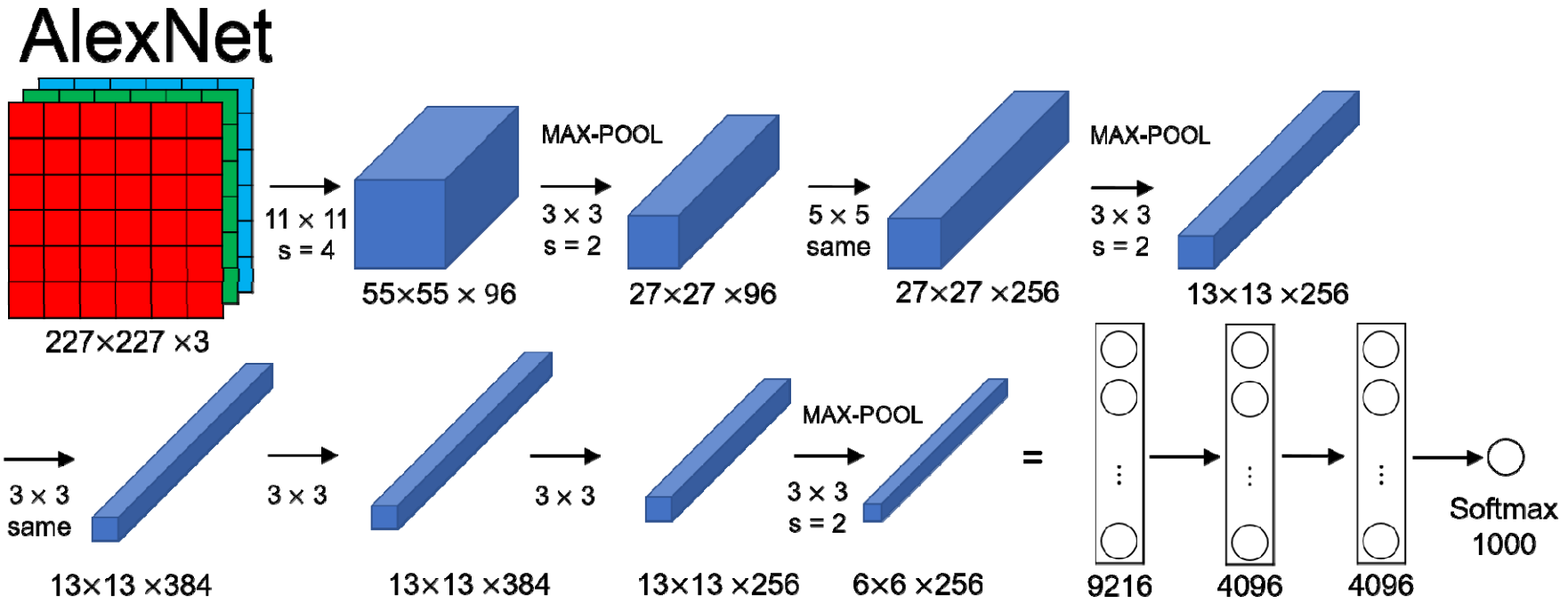
LeNet-5 comprises 7 layers, not counting the input, all of which contain trainable parameters.
Convolutional layers are labeled Cx, sub-sampling layers are labeled Sx, and FC layers are labeled Fx.

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

Case studies

- **AlexNet**

  The ILSVRC 2012 winner.



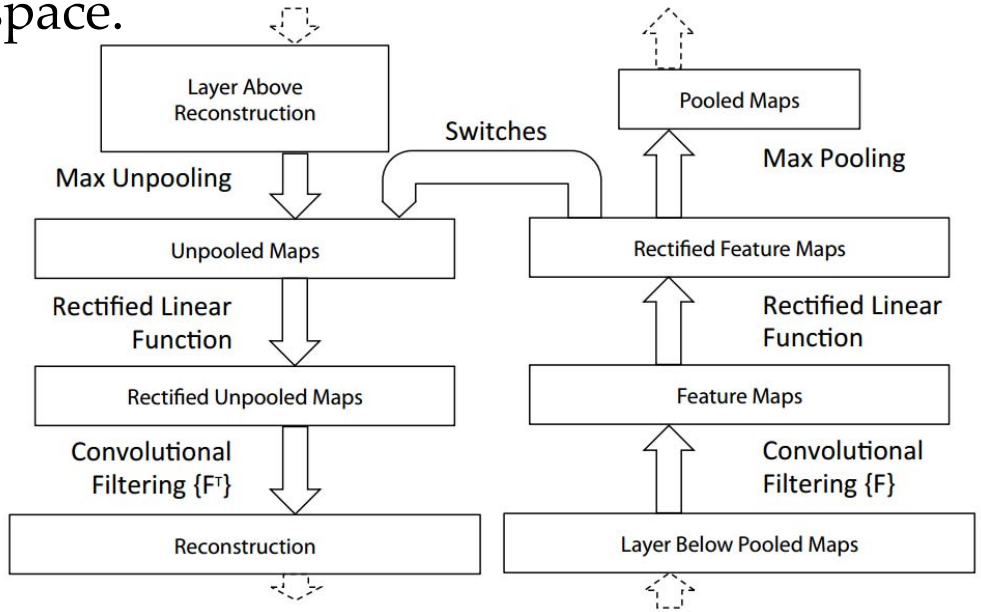
  The first work that popularized Convolutional Networks in Computer Vision.
  There are about 60M trainable parameters in the AlexNet.
  It's named after the author Alex Krizhevsky.

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Case studies

- **ZFNet**

The ILSVRC 2013 winner.
The paper present a novel way to map feature activities back to the input pixel space.



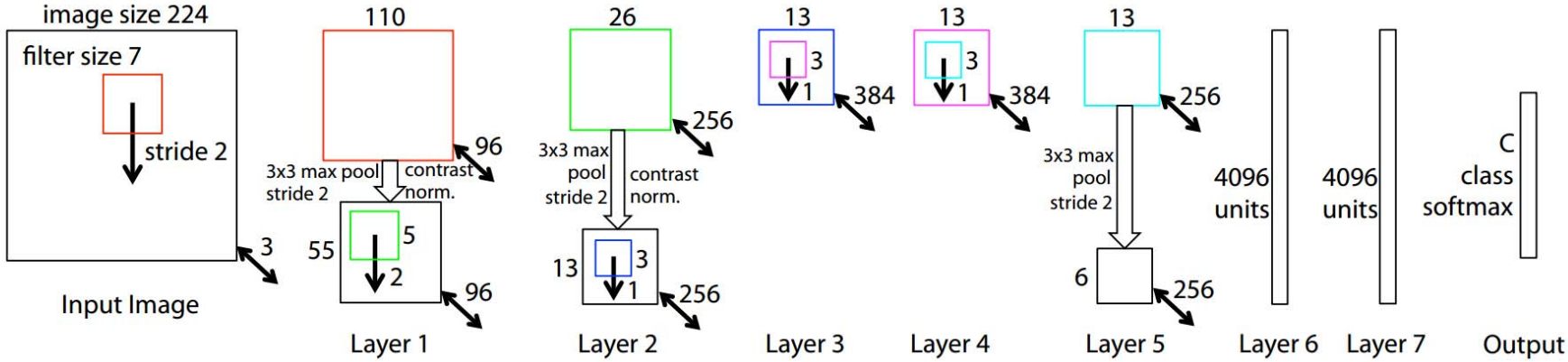
A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath.
The deconvnet use **transposed convolution**. The learned filter is flipped vertically and horizontally.

[Zeiler et al., 2014. Visualizing and Understanding Convolutional Networks]

Case studies

- **ZFNet**

The ZFNet model.



A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2.

The model use ReLU and max pooling.
The final layer is a C-way softmax function, C being the number of classes.
All filters and feature maps are square in shape.

[Zeiler et al., 2014. Visualizing and Understanding Convolutional Networks]

Case studies

- **VGGNet**

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG-16 VGG-19

[Simonyan et al., 2015. Very deep convolutional networks for large-scale image recognition]

Case studies

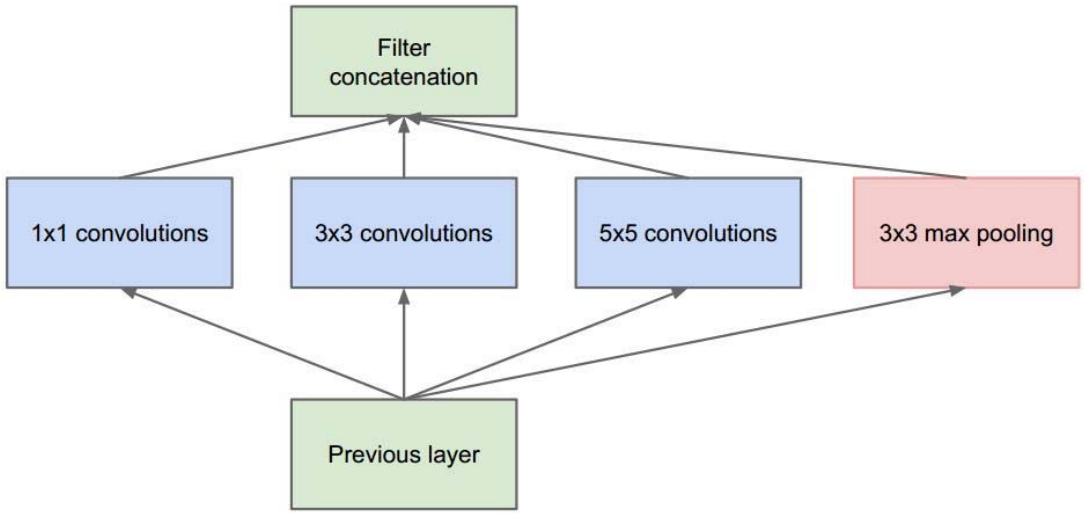
- **Inception**

  The ILSVRC 2014 winner.

  - motivation

    When we design a layer for a ConvNet, we may consider about the 1×1 conv filter, 3×3 conv filter , 5×5 conv filter, or we want a pooling layer. Why not do them all?

  - inception module

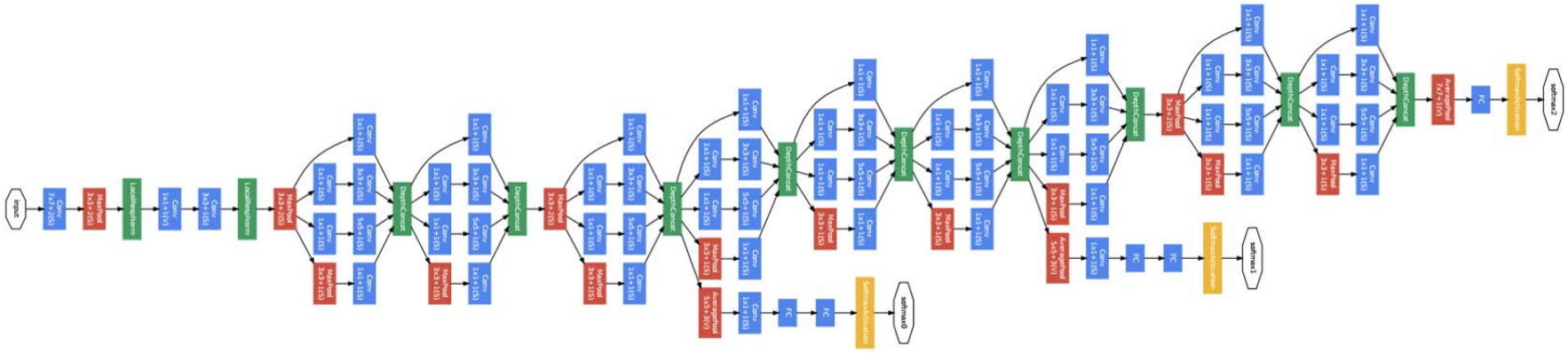


(a) Inception module, naïve version

[Szegedy et al., 2015. Going deeper with convolutions]

Case studies

- **GoogLeNet**
  - The "GoogLeNet" name refer to the particular incarnation of the Inception architecture used in the ILSVRC 2014 competition. It is a stack of many inception modules.



The picture is fuzzy. You can find the original one in the paper.

There are also Inception V2, Inception V3 and Inception V4. You can search them by yourself.

[Szegedy et al., 2015. Going deeper with convolutions]

Case studies

- **ResNet**
  - **residual block**

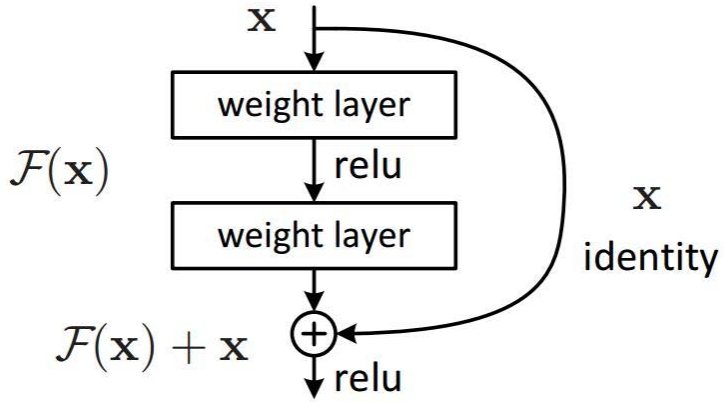


Figure 2. Residual learning: a building block.

  - Be careful that the dimensions of $\mathbf{x}$ and $\mathcal{F}$ must be equal.
  - ResNet is a stack of many residual blocks.
  - This really helps with the vanishing and exploding gradient problems and allows us to train much deeper neural networks without really appreciable loss in performance.
  - There are ResNet-34, ResNet-50, ResNet-101 and ResNet-152 in the paper.

[He et al., 2016. Deep residual learning for image recognition]

Natural Language Processing

美国的空气很好，尽管美国没有小龙虾吃。

输入7个字　　输入17个字

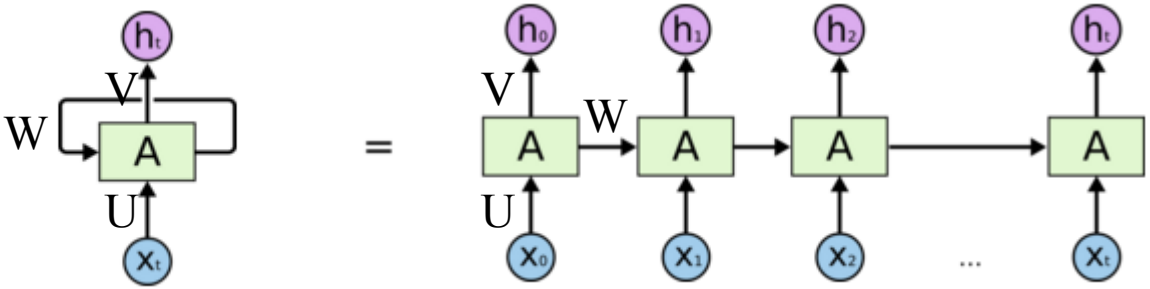
褒义

美国没有小龙虾吃，尽管美国的空气很好。

贬义

- Varying size inputs
- The order matters

Recurrent Neural Networks

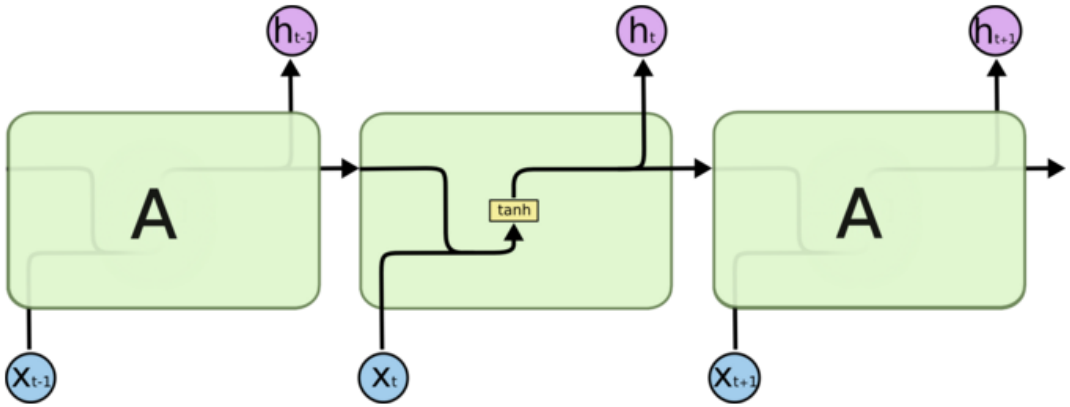


$$A_t = \tanh(\mathrm{U}x_t + \mathrm{W}A_{t-1})$$

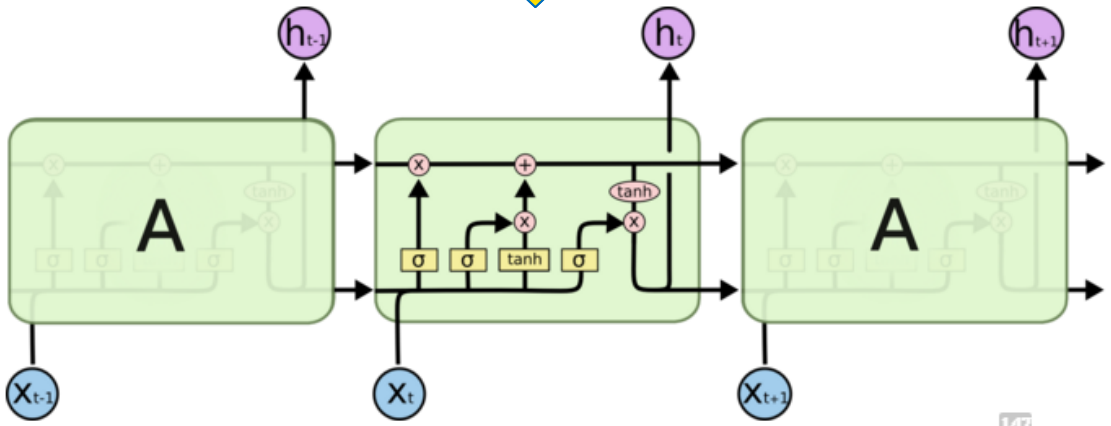
$$h_t = softmax(VA_t)$$

- Backpropagation Through Time (BPTT)
- BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem.
- For more details about BPTT: Alex Graves: Supervised Sequence Labelling with Recurrent Neural Networks.

Long Short Term Memory (LSTM)



The repeating module in a standard RNN contains a single layer.

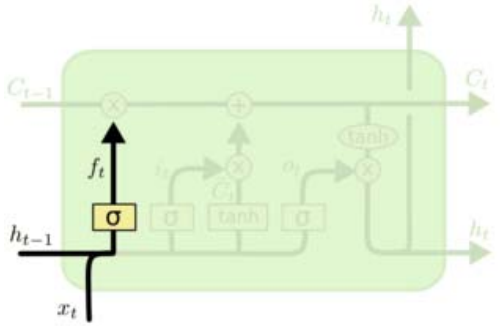


The repeating module in an LSTM contains four interacting layers.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, 1997.

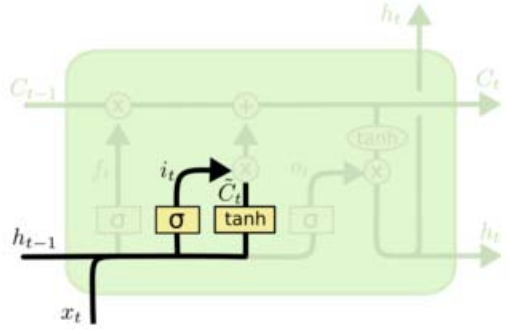
Long Short Term Memory (LSTM)

Forget gate



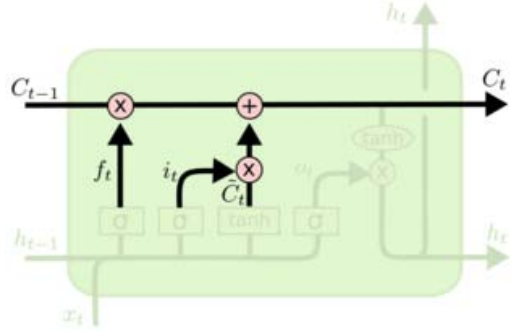
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Input gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

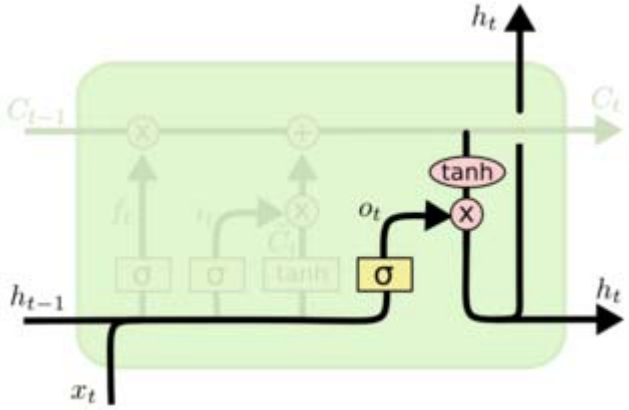


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Integration

Long Short Term Memory (LSTM)

Output gate

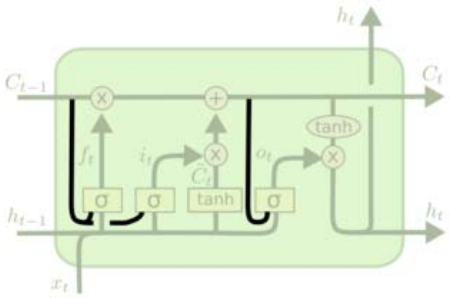


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

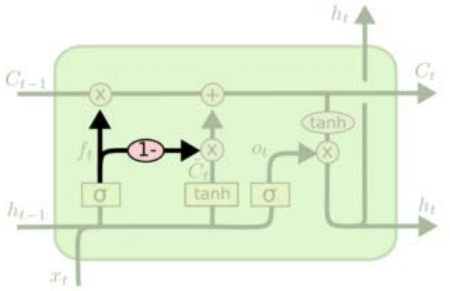
$$h_t = o_t * \tanh\left(C_t\right)$$

Long Short Term Memory (LSTM)

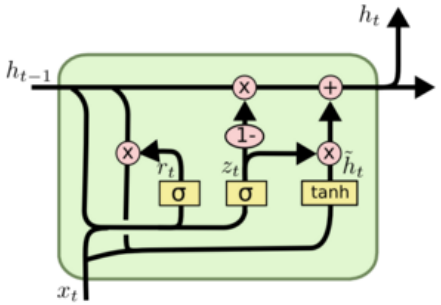
Variants



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$



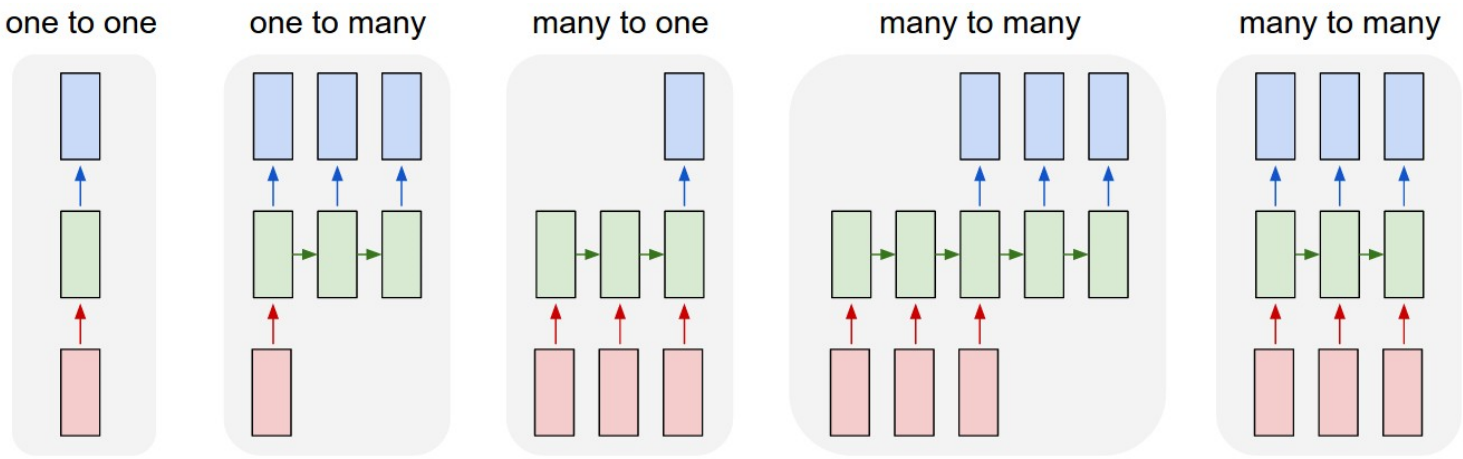
$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

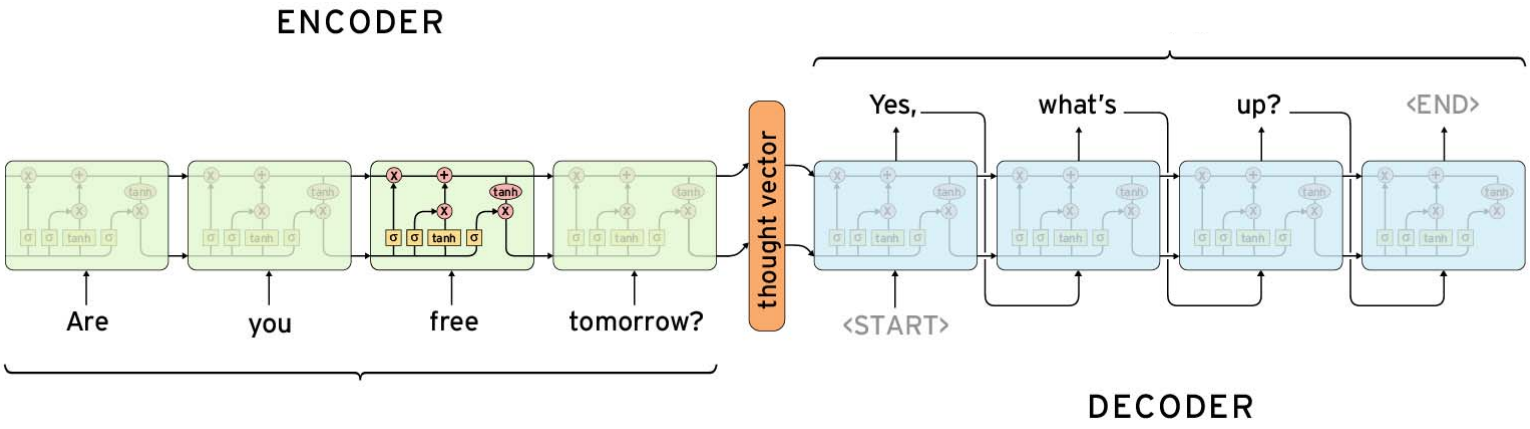
[1] Gers, F.; Schraudolph, N.; Schmidhuber, J. (2002). i"Learning precise timing with LSTM recurrent networks". Journal of Machine Learning Research. 3: 115–143.
[2] Gers, F. A.; Schmidhuber, J. (2001). "LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages". IEEE Transactions on Neural Networks. 12 (6): 1333–1340. doi:10.1109/72.963769.
[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Emprical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.," *arXiv*, vol. cs.NE, 2014.

RNN Inputs & Outputs



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Encoder-Decoder RNN Model



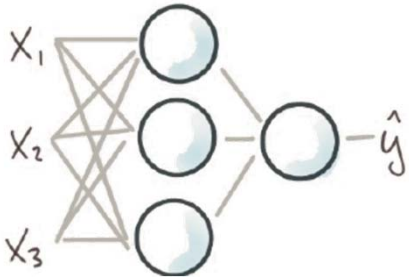
Variable-length Input

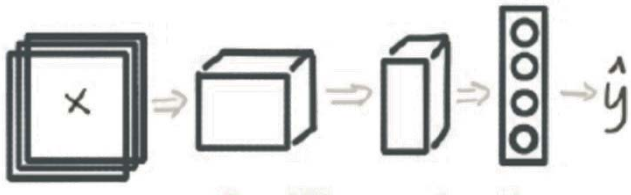
Variable-length Output

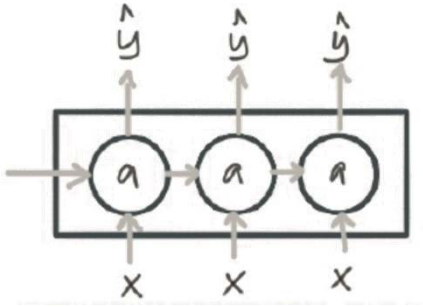
[1]D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv.org*, vol. cs.CL. 02-Sep-2014.
[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv.org*, vol. cs.CL. 11-Sep-2014.

Network Architecture (Topology)

Input: x	Output: y	NN Type
Home features AD + User INFO	Price Will click on AD(0/1)	 Standard NN
Image, Video	Object(1…1000) …	Convolutional NN (CNN)
Speech English	Text transcript Chinese	Recurrent NN (RNN)

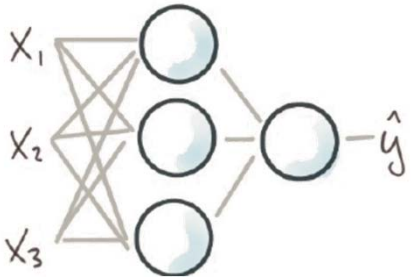


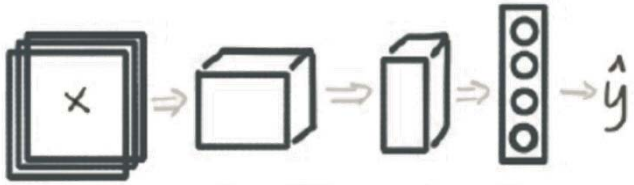


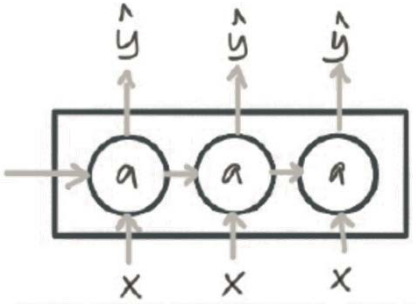


Summary

- Artificial Neural Networks (Deep Learning)
  - Net topology







  - Node (processor) characteristics
    - ReLU
  - Training/learning rules
    - Backpropagation Algorithm