ImageNet Classification with Deep Convolutional Neural Networks

Speaker: 吴良超

Members: 何君婷，吕慧雅，张豫，罗彭婷，巢娅， 谢澈澈，
邹敏艳，黄婷，李超然， 吴良超，孙伯毅

Outline

- Basis of CNN
- Design of the Network
- Training of the Network
- Performance of the Network
- Why it works

Outline

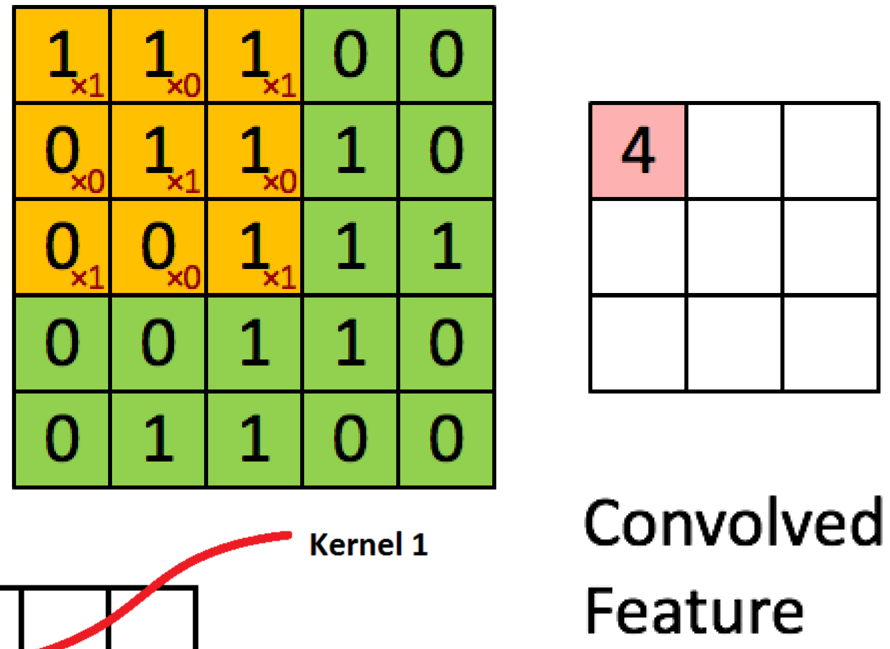
- ➢ Basis of CNN
- ➢ Design of the Network
- ➢ Training of the Network
- ➢ Performance of the Network
- ➢ Why it works

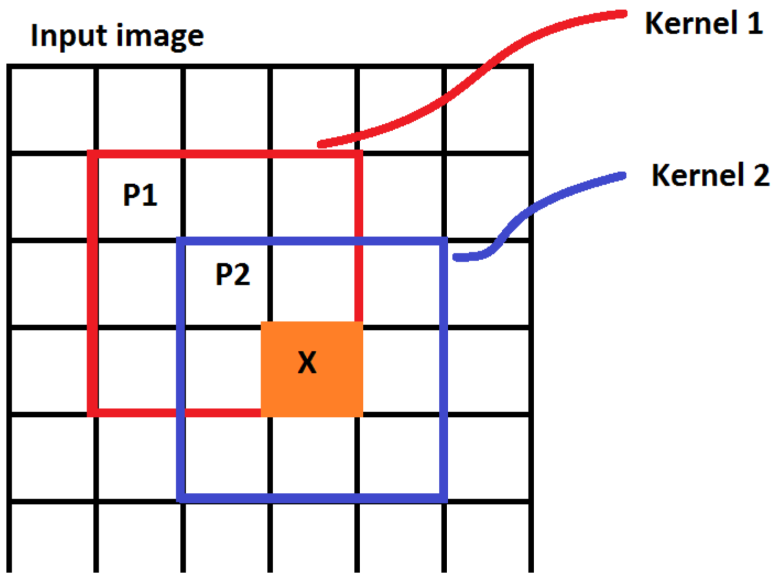
Basis of CNN

- CNN: abbreviation of Convolutional Neural Network
- feed-forward network, layer by layer
- special connectivity pattern
  - convolutional layer
  - pooling layer
  - ReLU layer
  - fully connected layer

Basis of CNN – Convolutional Layer

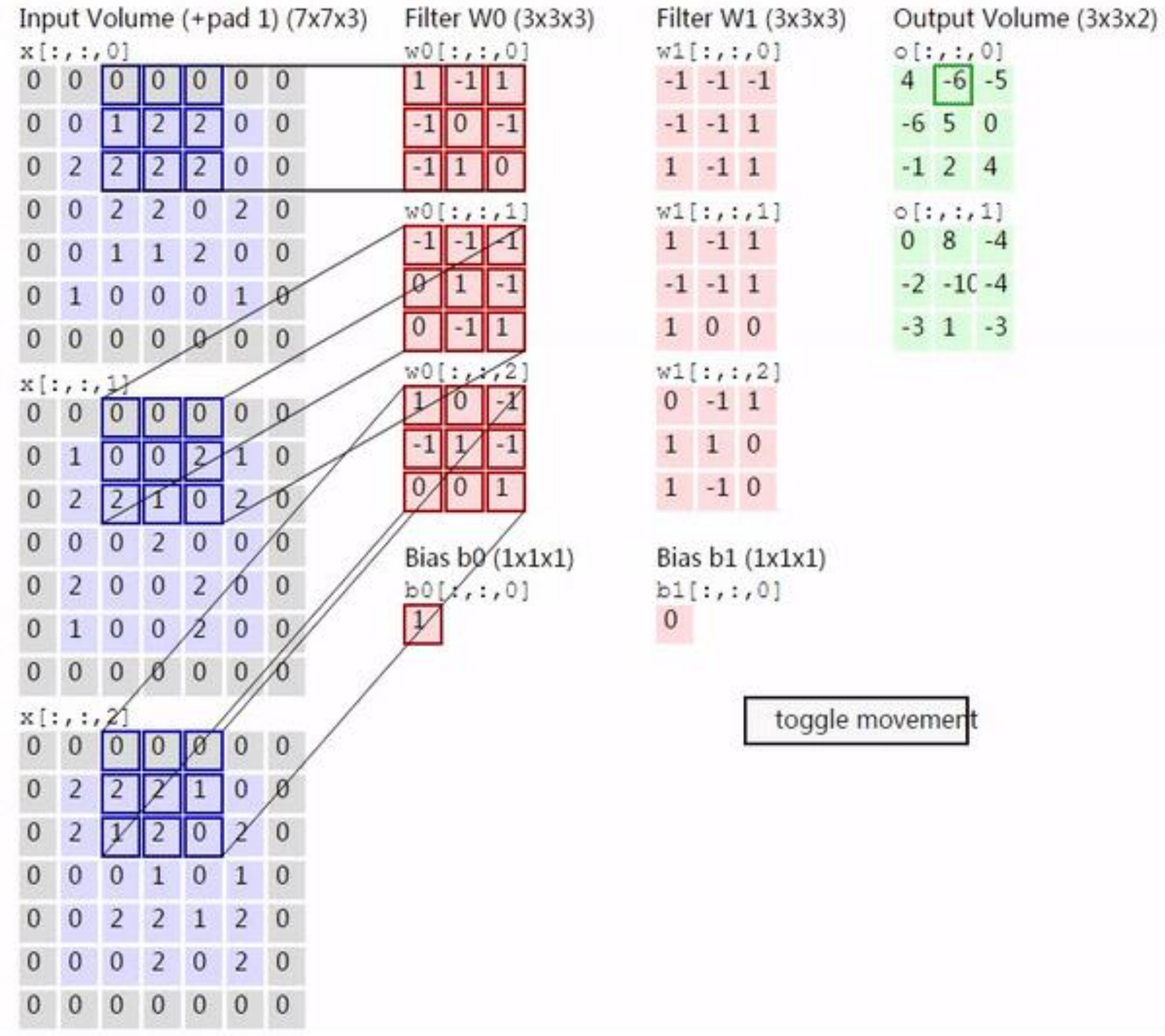
- kernel/filter
- convolution: dot product between kernel and input
- activation map: result of convolution
- feature extraction
- multiple kernels





Basis of CNN – Convolutional Layer

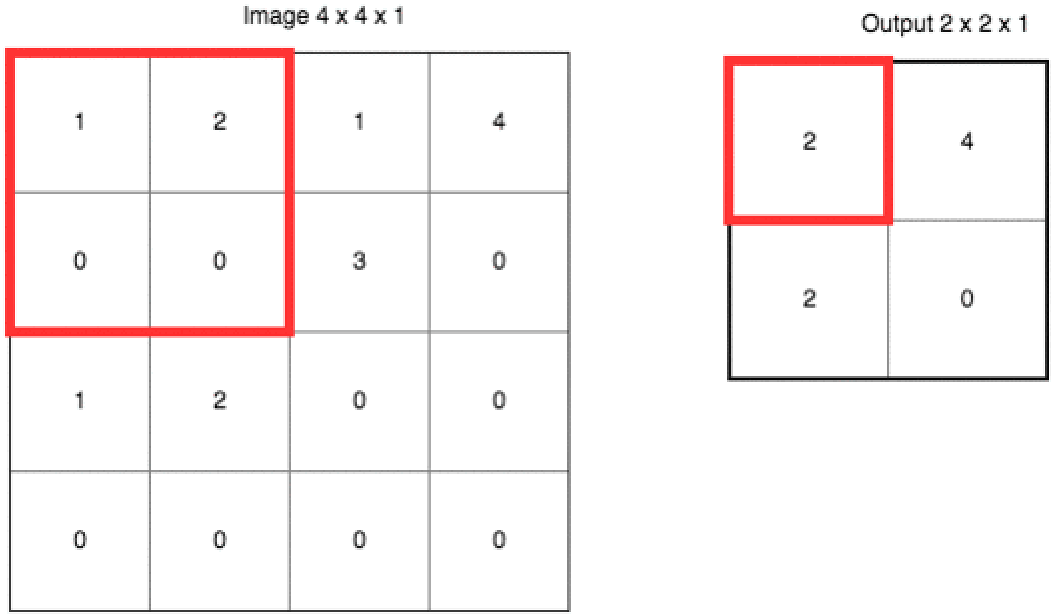
- ◆ channels of RBG image
- ◆3 channels, 2 kernels
- ◆one kernel -> one activation map



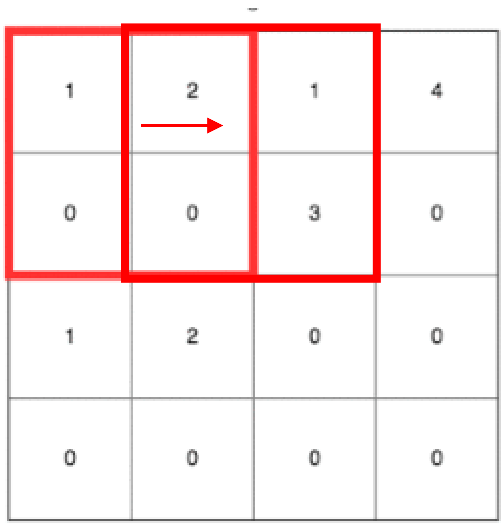
Source: http://cs231n.github.io/convolutional-networks/#conv

Basis of CNN – Pooling Layer

- pooling : a form of non-linear down-sampling
- different ways: max pooling, average pooling....
- reduce the number of parameters and amount of computation
- control overfitting
- overlapping pooling



max pooling with a 2x2 kernel and stride = 2



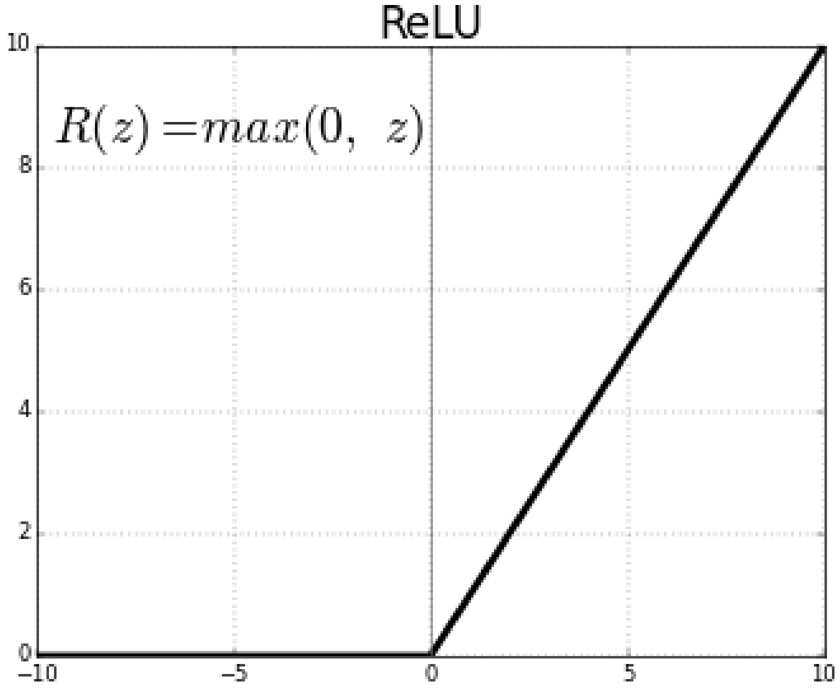
max pooling with a 2x2 kernel and stride = 1

Basis of CNN – ReLU Layer

◆ReLU : abbreviation of Rectified Linear Units

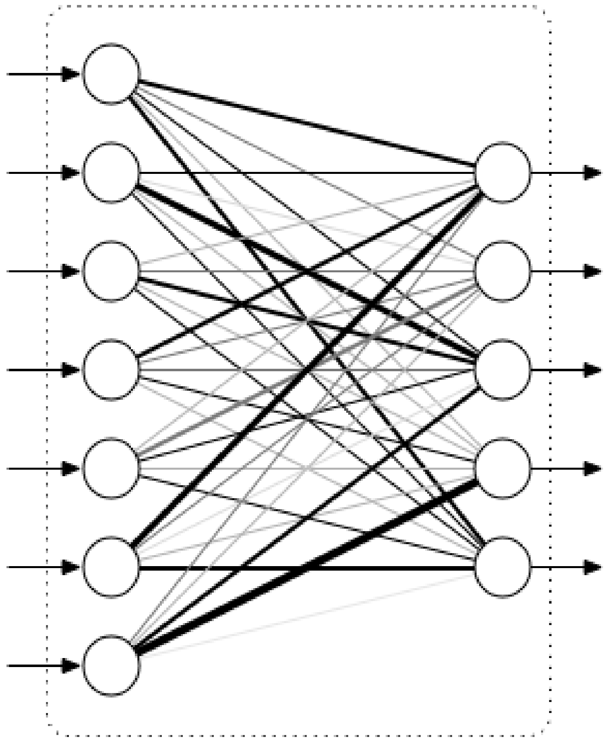
◆one kind of activation function

◆increases the nonlinear properties of the decision function



Basis of CNN – Fully Connected Layer

- regular neural network
- full connections to all neurons in the previous layer
- usually at the last layer of the network as output

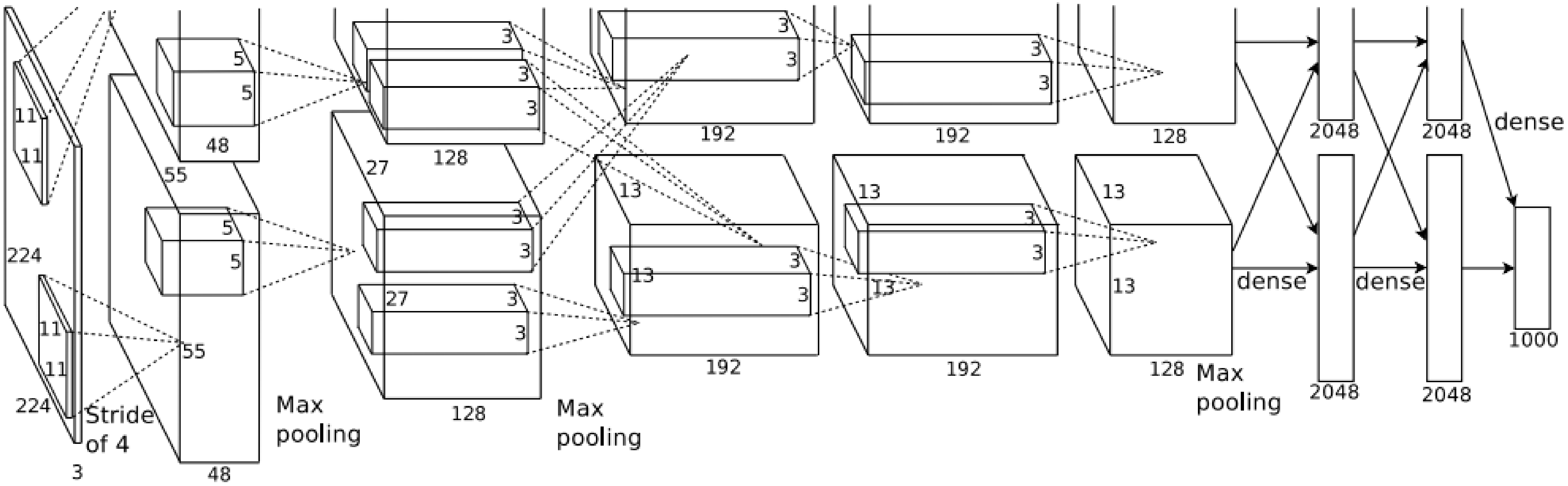


Outline

- Basis of CNN
- Design of the Network
- Training of the Network
- Performance of the Network
- Why it works

Overview of the Network

- ◆5 convolutional layers + 3 fully connected layers, train on two GPUs
- ◆max polling after the 1^{st}, 2^{nd},5^{th} convolutional layer
- ◆ReLU activation after each convolutional layer and fully connected layer
- ◆also called Alexnet

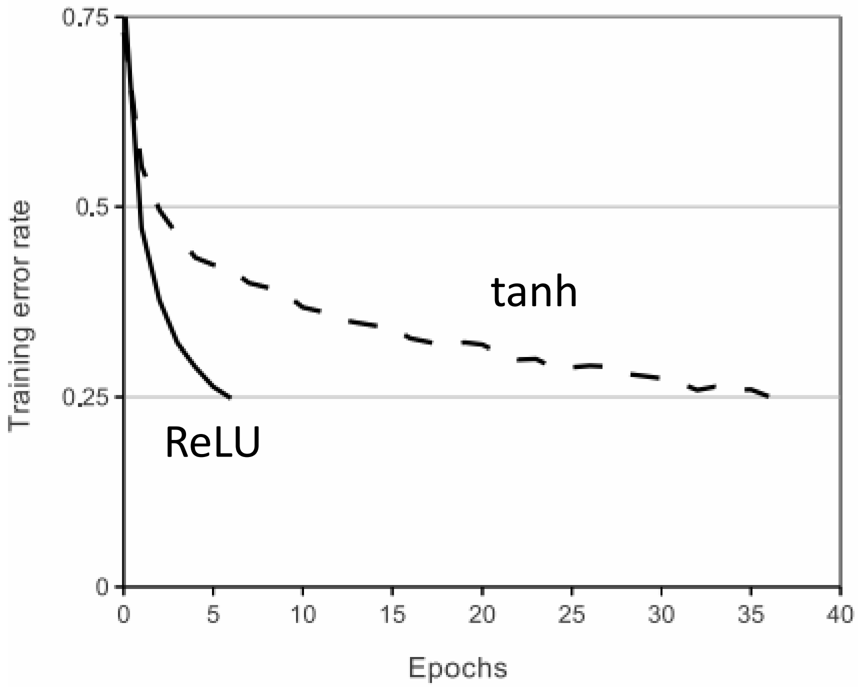


Important Features of the Network

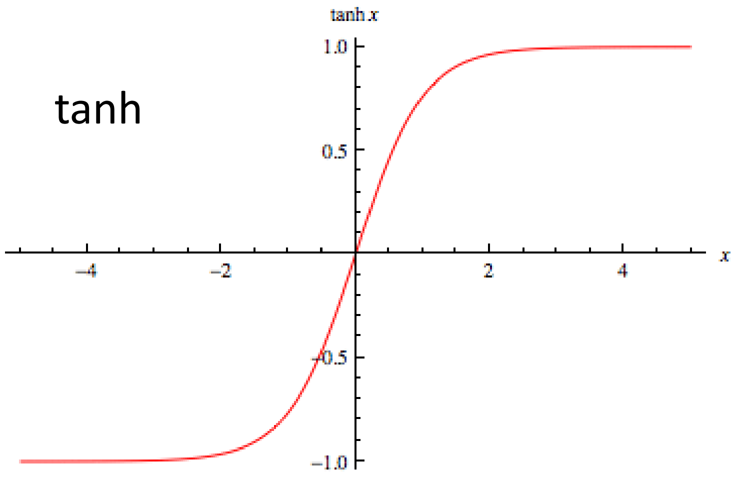
- ReLU Nonlinearity
- Local Response Normalization
- Overlapping Pooling

ReLU Nonlinearity

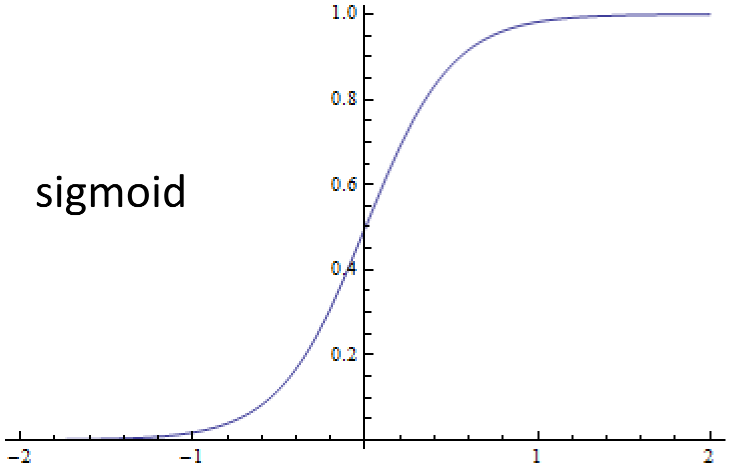
- ◆comparision of activation function: tanh, sigmoid, ReLU
- ◆ReLU converges faster than the other two
- ◆sigmoid, tanh suffers from vanishing gradient



tanh



sigmoid



Local Response Normalization

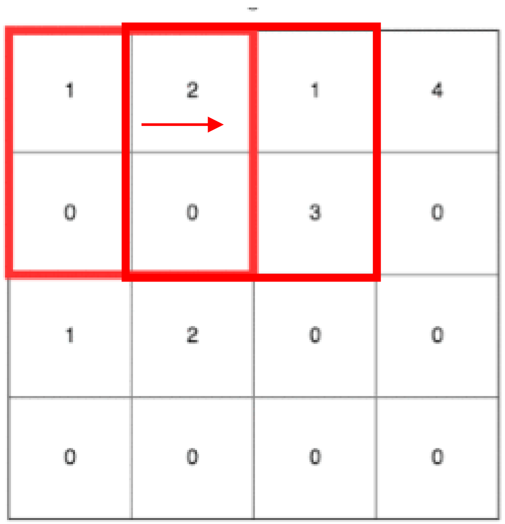
- normalize the output after ReLU
- $a_{x,y}^{i}$: output of kernel i at position (x, y) after ReLU

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta}$$

- k, n, α, β are hyper-parameters determined by validation
- applied after the first two convolutional layers
- reduces top-1 and top-5 error rates by 1.4% and 1.2%,respectively

Overlapping Pooling

◆reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively;



Outline

- ➢ Basis of CNN
- ➢ Design of the Network
- ➢ Training of the Network
- ➢ Performance of the Network
- ➢ Why it works

Training of the Network

◆objective function

◆training algorithm

◆training on multiple GPUs

◆reducing overfitting

Objective Function – Softmax

◆ target of the task : classification of 1000 categories

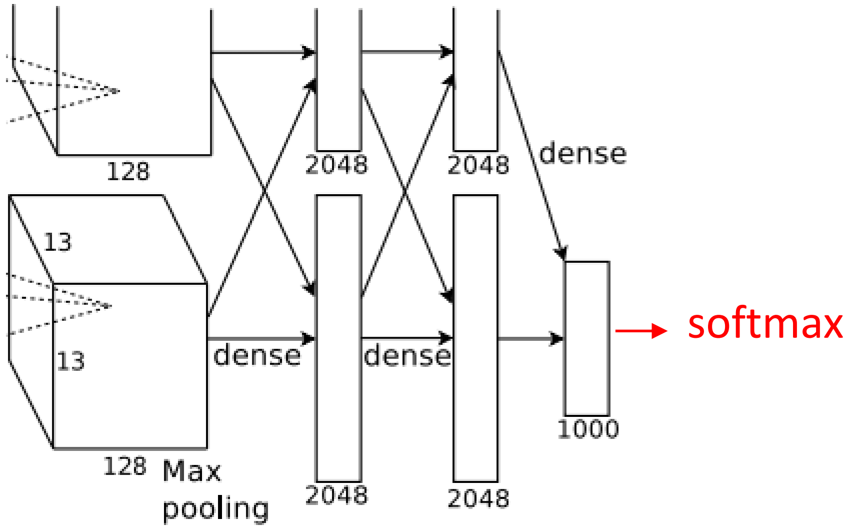
◆ multiple classification with softmax

- z : K-dimensional vector of arbitrary real values

softmax $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.

- $\sigma(z)$: K-dimensional vector of real values in the range (0, 1) that add up to 1

◆ $\sigma(z)$ can be used to represent a probability distribution of categories



Objective Function – Maximum Likelihood

- $\sigma(z)$: probability distribution of categories
- maximum log likelihood(m samples, k categories)

$$\max \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log \frac{e^{z_j}}{\sum_{l=1}^{k} e^{z_l}}$$

1{True} = 1, 1{False} =0

e.g. 1{2=3} = 0, 1{3=3} = 1

$$\min - \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log \frac{e^{z_j}}{\sum_{l=1}^{k} e^{z_l}}$$

- cross-entropy error $$\min - \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log \frac{e^{z_j}}{\sum_{l=1}^{k} e^{z_l}}$$ unconstrained optimization problem

Training Algorithm – Stochastic Gradient Descent

- SGD with momentum, weight decay and batch size
- momentum: update the current total gradient with previous one
- weight decay: L2 regularization
- batch size: more than one sample at each update
- update rule

$$
\begin{aligned}
v_{i+1} &:= 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i} \\
w_{i+1} &:= w_i + v_{i+1}
\end{aligned}
$$

i - iteration index

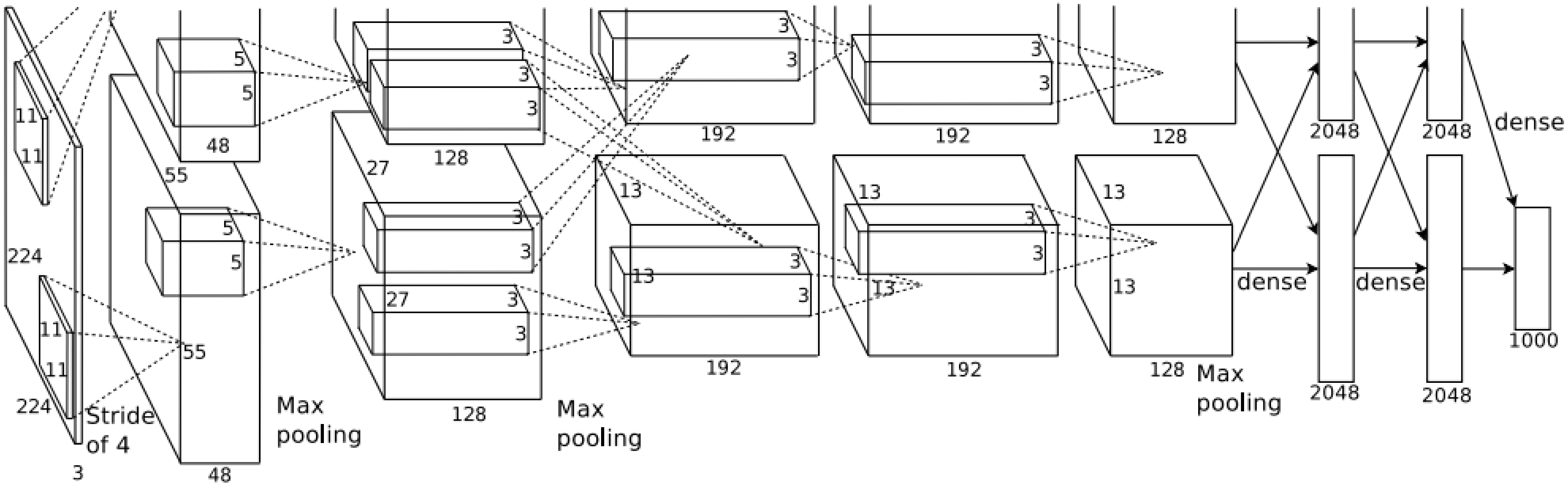
v_i - momentum variable/total gradient

ϵ- learning rate, step size

$\left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$ average gradient over the batch D_i

Training on Multiple GPUs

- ◆ read and write on GPU's memory , independent of host machine memory
- ◆ single GPU limits the maximum size of the networks can be trained
- ◆communicate only in certain layers for cross-validation
- ◆compared to one GPU, reduces top-1 and top-5 error rates by 1.7% and 1.2%, respectively



Reducing Overfitting

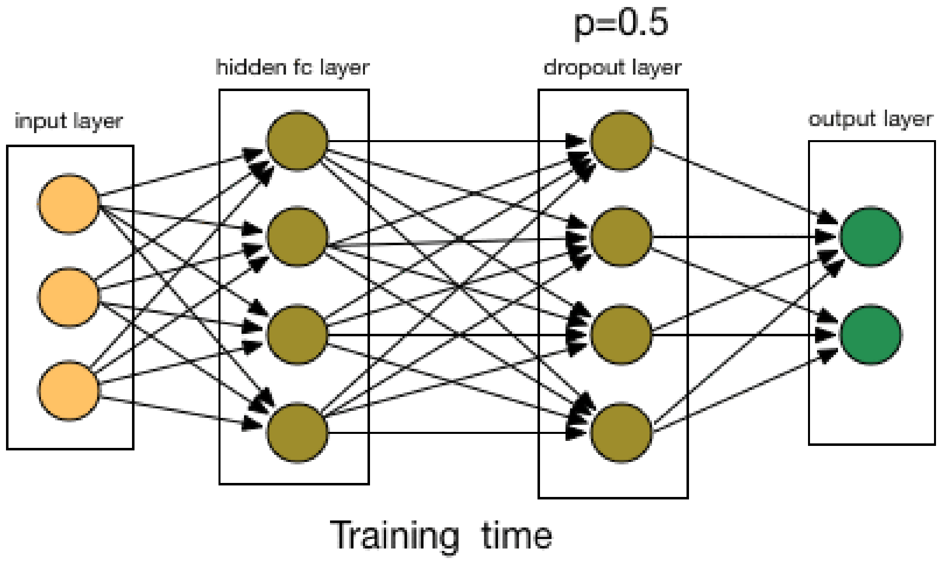
- data augmentation
- dropout

Reducing Overfitting – Data Augmentation

- extract multiple random 224×224 patches from the raw 256×256 images
- at each pixel, add extra information obtained from PCA on the image
- reduce the top-1 error rate by over 1%

Reducing Overfitting – Dropout

- setting the output of each hidden neuron to zero with probability 0.5
- reduces complex co-adaptations of neurons, more robust
- apply dropout in the first two fully-connected layers



Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Outline

- Basis of CNN
- Design of the Network
- Training of the Network
- Performance of the Network
- Why it works

Dataset

- ImageNet (http://image-net.org/)
- competition: ImageNet Large-Scale Visual Recognition Challenge(ILSVRC)
- offer 1000 images in each of 1000 categories
- roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images

Result

- ILSVRC-2010

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

- ILSVRC-2012

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Outline

- Basis of CNN
- Design of the Network
- Training of the Network
- Performance of the Network
- Why it works

Why it works

- it works -> small generalization error
- theoretical guarantee about the error
- VC dimension (statistical machine learning)

good generation performance requires the ratio of the number of samples to the number of parameters should be roughly greater than 10

$$\frac{\#samples}{\#parameters} \geq 10$$

- 60 million parameters, 1.25 million samples(training + validation)
- VC dimension doesn’t seem to work on the network

Why it works

- ***Understanding Deep Learning Requires Rethinking Generalization*[1]**, best paper in ICLR 2017
- bigger model, smaller $\frac{\#sample}{\#parameter}$, but smaller generalization error
- new theory need to be proposed to explain this problem

[1]: Zhang C, Bengio S, Hardt M, et al. Understanding deep learning requires rethinking generalization[J]. arXiv preprint arXiv:1611.03530, 2016.

Thank you for listening