南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Machine Learning and NeuroEngineering

机器学习与神经工程

Lecture 18 – More about CNN

Quanying Liu （刘泉影）

SUSTech, BME department

Email: liuqy@sustech.edu.cn

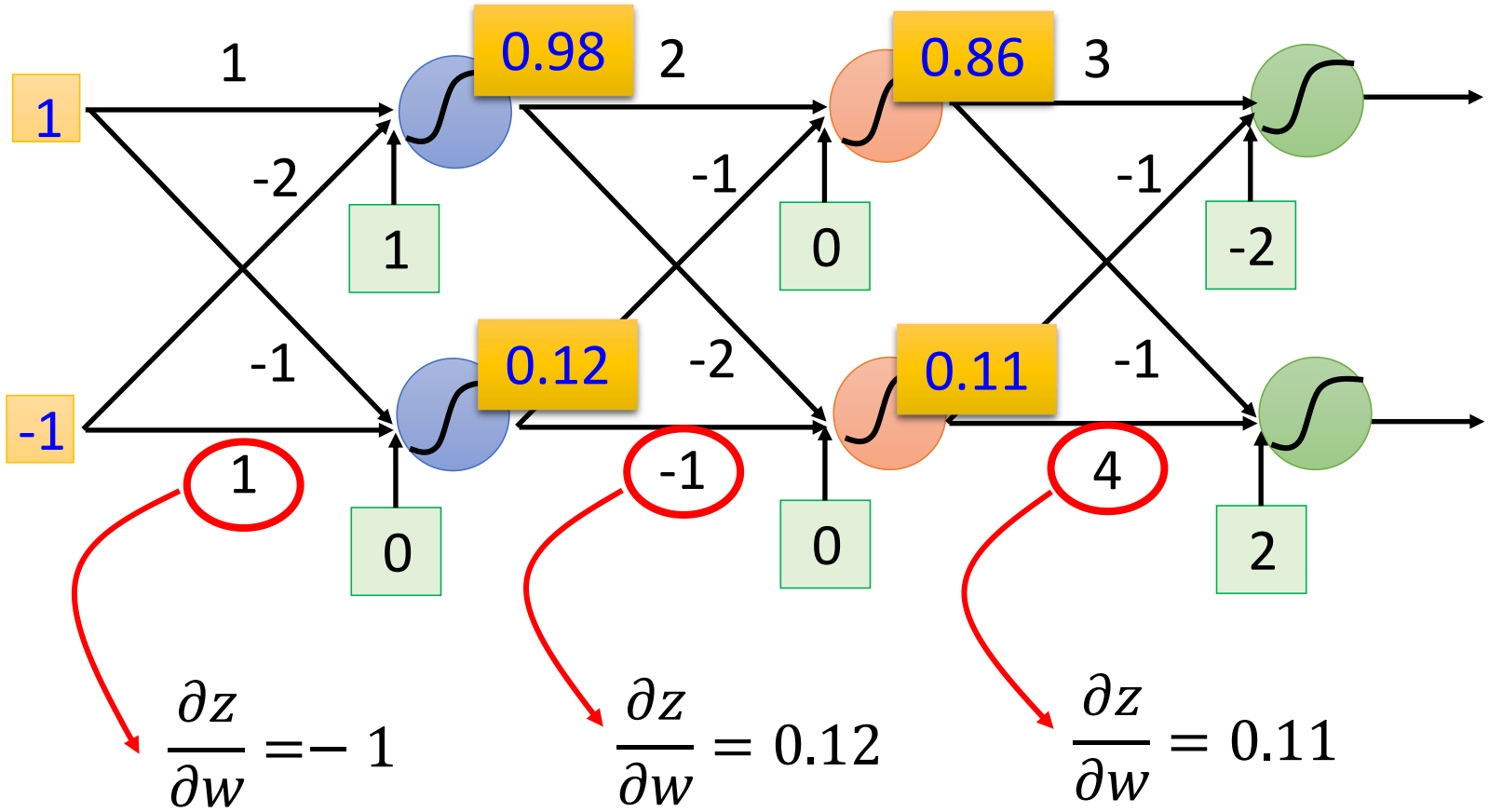
Lecture 15/16 - Recap

- Gradient Descent (GD)
  - What is Gradient Descent?
  - Gradient Descent to train deep NNs → Error Backpropagation
- Error Back-propagation (BP)
  - Backpropagation
  - Backpropagation – forward pass
  - Backpropagation – backward pass
- The Architecture of CNN
  - Convolution
  - Activation function
  - Pooling
  - Flatten
  - FC
- CNN Hands-on (tensorflow), thanks to 曲由之--> next lecture

Some slides are adjusted from 李宏毅

Backpropagation – Forward pass

Compute $\partial z / \partial w$ for all parameters

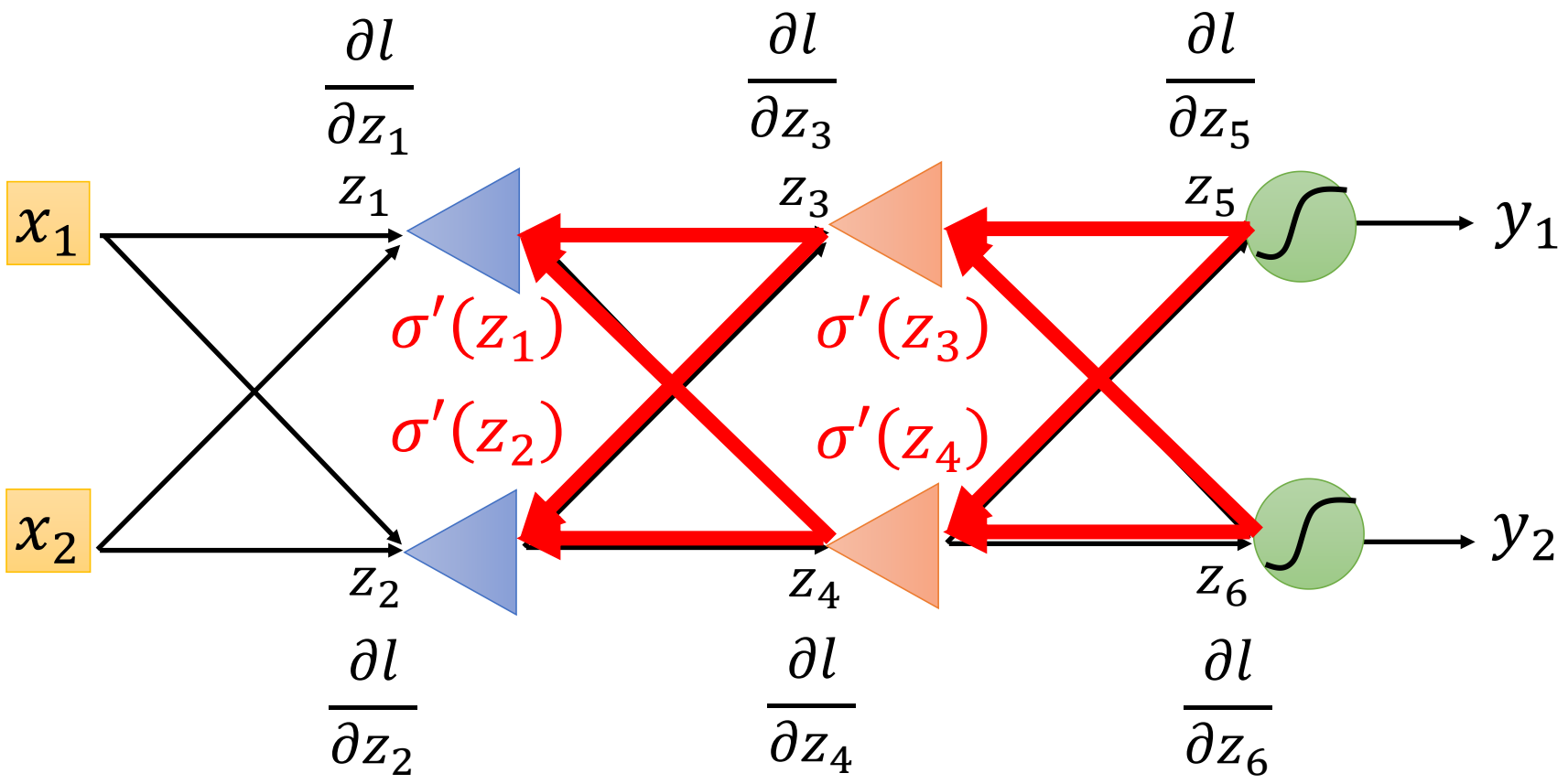


Backpropagation – Backward pass

Compute $\partial l/\partial z$ for all *activation function* inputs z

Compute $\partial l/\partial z$ from the output layer

$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial l}{\partial z_1} + w_4 \frac{\partial l}{\partial z_2}\right]$$



Suppose we have a NN (see Figure 1)

input layer x_i,

hidden layer $z_k = \arctan\left(\sum_{i=1}^{3} w_{ki} x_i\right)$,

output layer $\hat{y}_j = \sum_{k=1}^{4} v_{jk} z_k$,

loss function $L(y, \hat{y}) = \sqrt{\frac{1}{2}((\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2)}$. $\hat{y}_j$ is prediction, y_j is ground truth

Write down $\frac{\partial L}{\partial v_{jk}}$ and $\frac{\partial L}{\partial w_{ki}}$ in terms of only x_i, z_k, y_j, $\hat{y}_j$, w_{ki}, and/or v_{jk}.

Hint: $\frac{\partial \arctan(\alpha)}{\partial \alpha} = \frac{1}{\alpha^2 + 1}$.

$$\frac{\partial L}{\partial v_{jk}} = \frac{\partial L}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial v_{jk}}$$

$$\beta = \sqrt{\frac{1}{2}((\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2)}$$

$$\frac{\partial L}{\partial \hat{y}_1} = \frac{\partial}{\partial \hat{y}_1}\beta(\hat{y}_1) = \frac{1}{2\beta}(\hat{y}_1 - y_1)$$

$$\frac{\partial \hat{y}_j}{\partial v_{jk}} = \sum_{k=1}^{4} v_{jk} z_k = z_k$$

$$\rightarrow \frac{\partial L}{\partial v_{jk}} = \frac{1}{2\beta}(\hat{y}_i - y_i) z_k$$

$$\frac{\partial L}{\partial w_{ki}} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial w_{ki}}$$

$$\frac{\partial L}{\partial z_k} = \frac{\partial L}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial z_k} + \frac{\partial L}{\partial \hat{y}_2}\frac{\partial \hat{y}_2}{\partial z_k} = \sum_{j=1}^{2} \frac{1}{2\beta}(\hat{y}_j - y_j) v_{jk}$$

$$\frac{\partial z_k}{\partial w_{ki}} = \frac{\partial}{\partial w_{ki}} \arctan\left(\sum_{i=1}^{3} w_{ki} x_i\right) = \frac{x_i}{\tan^2(z_k) + 1}$$

$$\rightarrow \frac{\partial L}{\partial w_{ki}} = \left(\sum_{j=1}^{2} \frac{1}{2\beta}(\hat{y}_j - y_j) v_{jk}\right) \frac{x_i}{\tan^2(z_k) + 1}$$

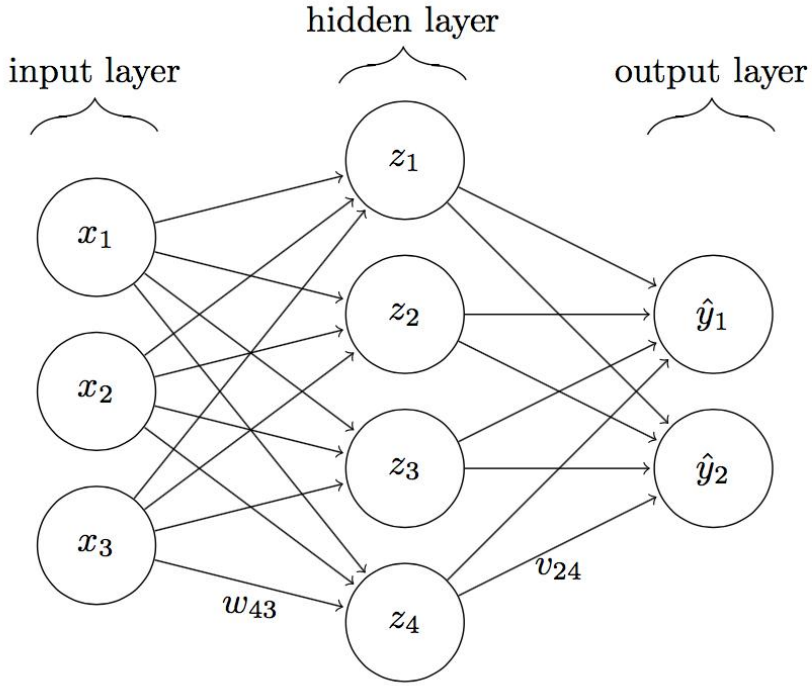


Figure 1: A neural network with one hidden layer.

Recall CNN architecture

Convolution

Activation function

Pooling

These blocks can repeat many times

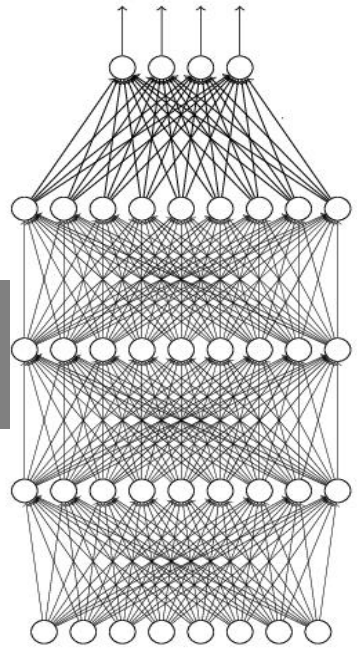
Convolution

Activation function

Pooling

Flatten

Fully Connected Feedforward network



cat / dog ……

Some Activation Functions

1. Binary Step Function
2. Sigmoid / Logistic
3. TanH / Hyperbolic Tangent
4. Softmax
5. ReLU (Rectified Linear Unit)
6. Leaky ReLU
7. Parametric ReLU
8. …

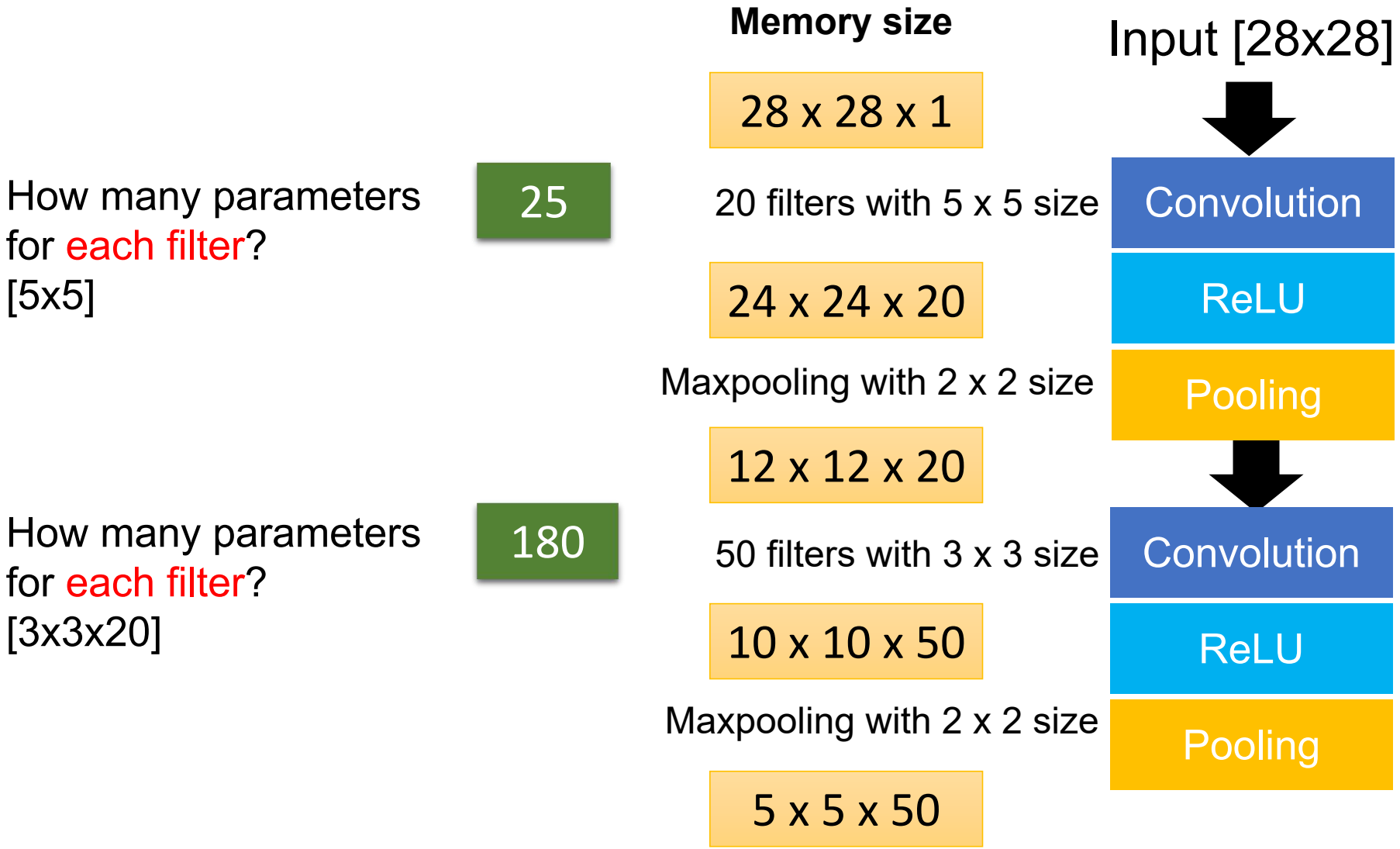
Lecture 17 – Recap

0. Marr's 3 levels of explanation
1. Evolution of the eye
2. Function of the visual system
3. Structure of the eye
4. Photoreceptors
5. Information integration by ganglion cell
6. **Visual pathways**: photoreceptors, interneurons, ganglion cells, LGN, V1, ventral/dorsal pathways
7. **Image process: brain vs. CNN**

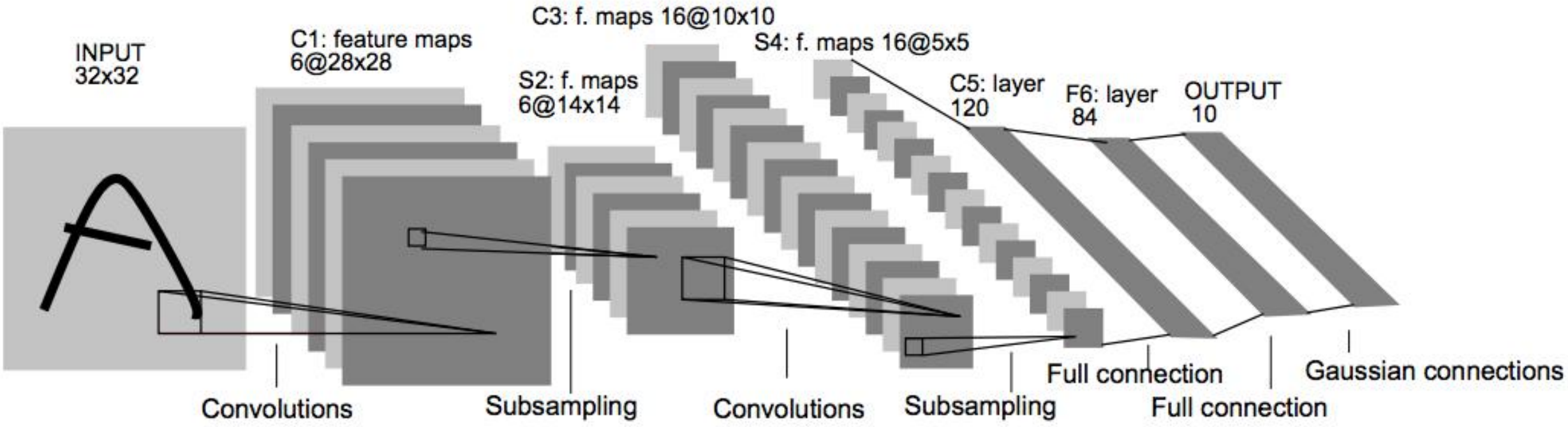
Lecture 18 – More about CNN

1. Parameter number and memory size in CNN
   - Examples with some well-known CNNs (LeNet, AlexNet, VGG-16)
2. Three components of CNN
   - Objective function / loss function (cross-entropy loss, mse loss…)
   - Learning rule (Adagrad, Momentum, Adam…)
   - Network architecture (VGG, ResNet, U-net…)
3. Course projects

Memory size & parameter size of CNN

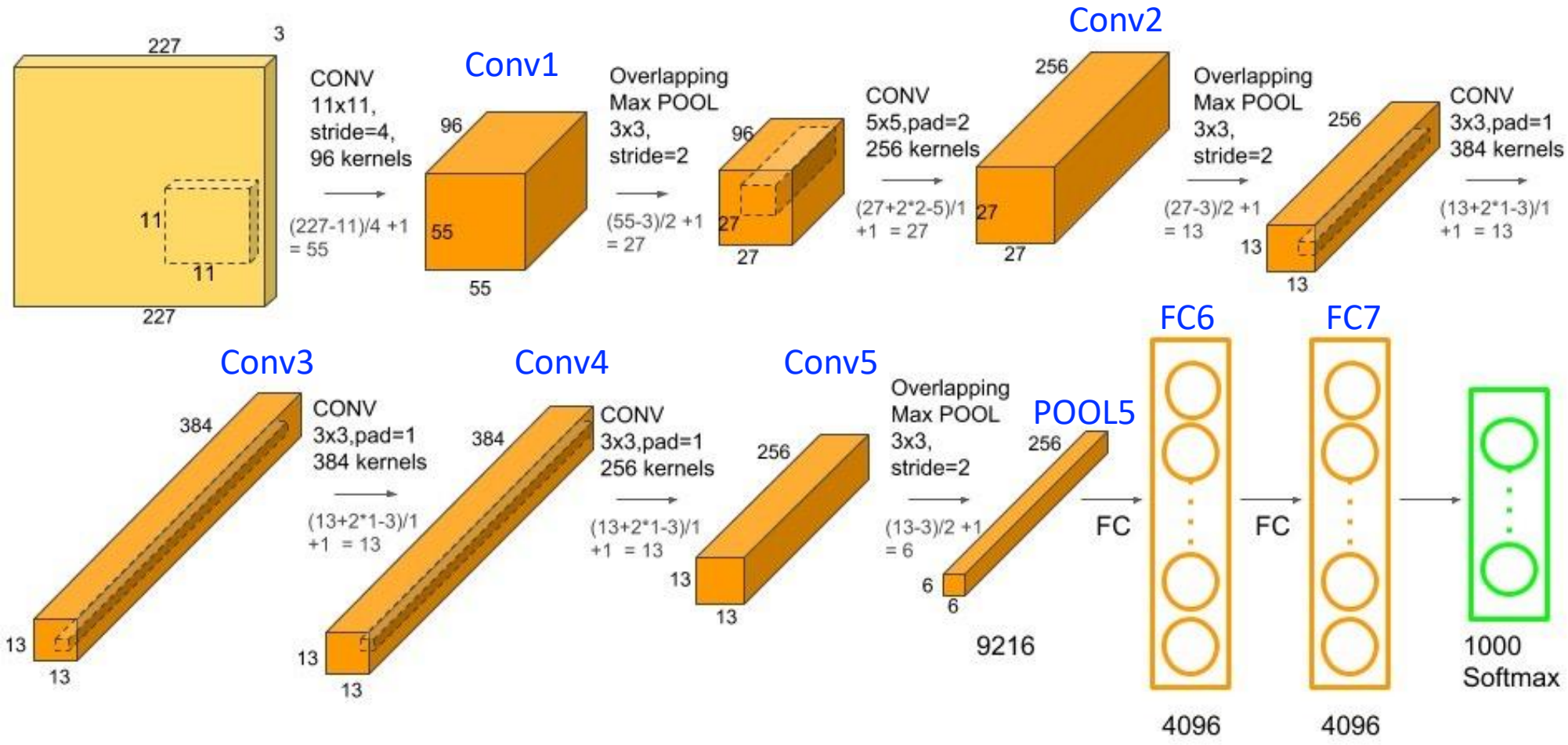


Memory size & parameter size of LeNet

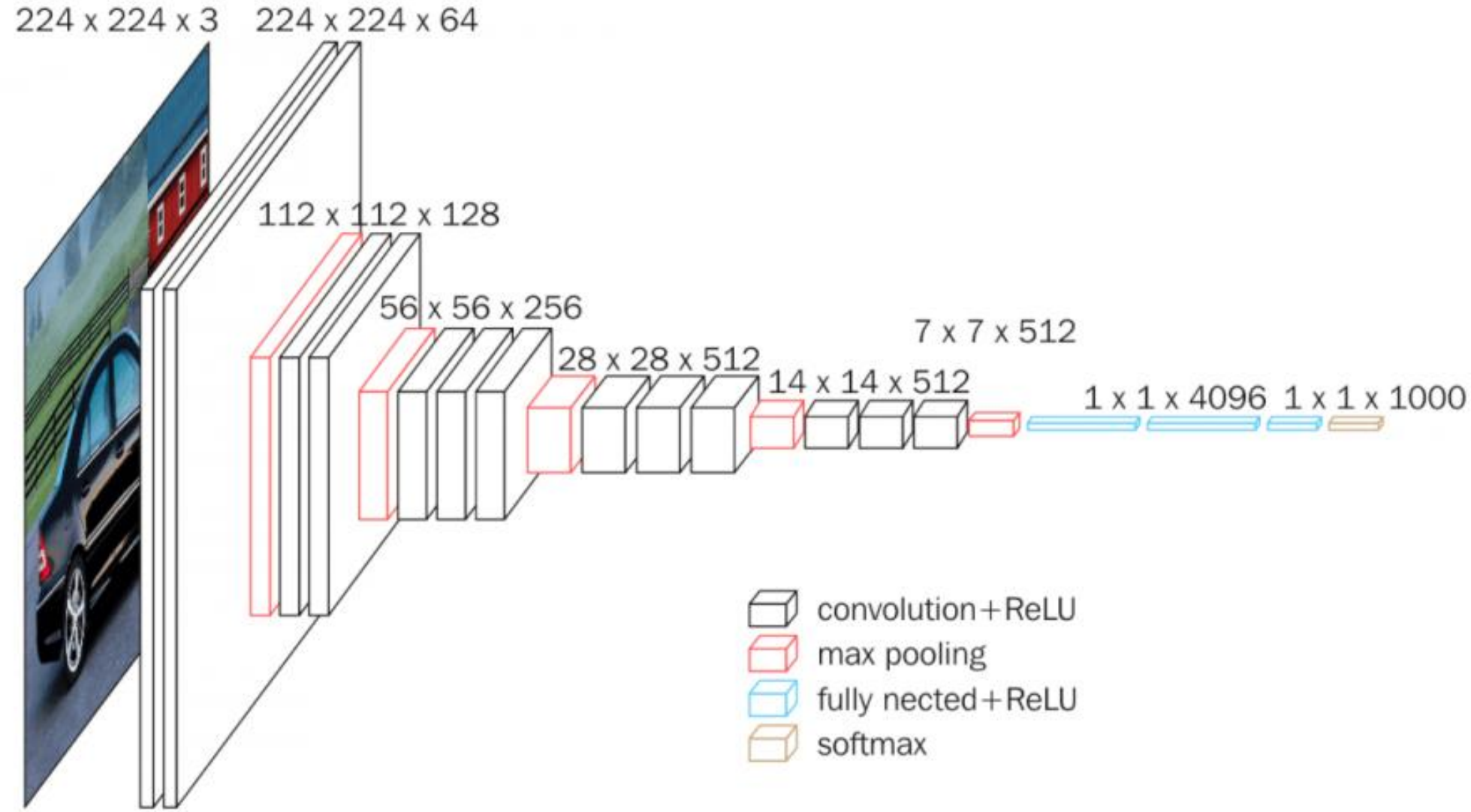


[5x5]x6 [5x5x6]x16 120x84 84x10

Memory size & parameter size of AlexNet



Memory size & parameter size of VGG-16



INPUT: [224x224x3]	memory: 224*224*3=150K	weights: 0
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	weights: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	weights: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	weights: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	weights: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	weights: 0
FC: [1x1x4096]	memory: 4096	weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]	memory: 4096	weights: 4096*4096 = 16,777,216
FC: [1x1x1000]	memory: 1000	weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= **93MB** / image (only forward! ~*2 for backward)

TOTAL parameters: **138M** parameters

Three components of CNN

- **Objective function / Loss function**

The learning goal, expressed as an objective function (or loss function) to be maximized or minimized

- **Learning rule**

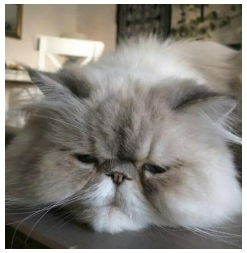
A set of learning rules, expressed as synaptic weight updates

- **Network architecture**

Expressed as the pathways and connections for information flow

Classification Task: Detect whether there is a cat in an image

Data:
labeled image dataset $\{(x_i, y_i)\}_{i=1}^{N}$



1

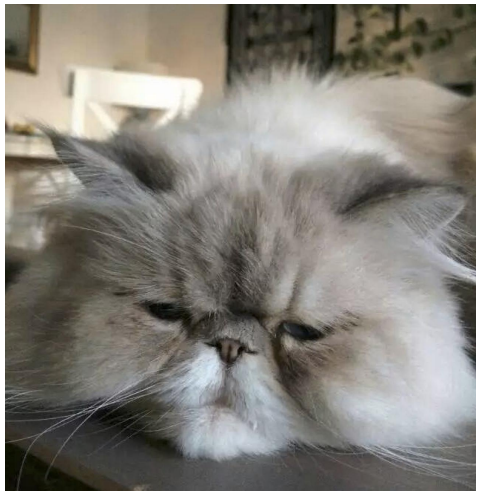
1

0

Classification Task: Detect whether there is a cat in an image

Data:
labeled image dataset $\{(x_i, y_i)\}_{i=1}^{N}$

INPUT x_i



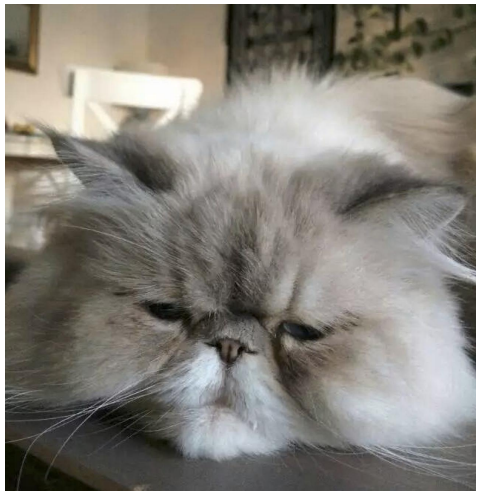
OUTPUT
$y_i = 1$

Classification Task: Detect whether there is a cat in an image

Data:
labeled image dataset

$$L(y_i, y_i)$$

INPUT x_i



$$S(x) = \frac{1}{1 + e^{-x}}$$

OUTPUT
$y_i = 1$

Neural Network

$$\hat{y}_i = f(x_i; \boldsymbol{\theta})$$

sigmoid

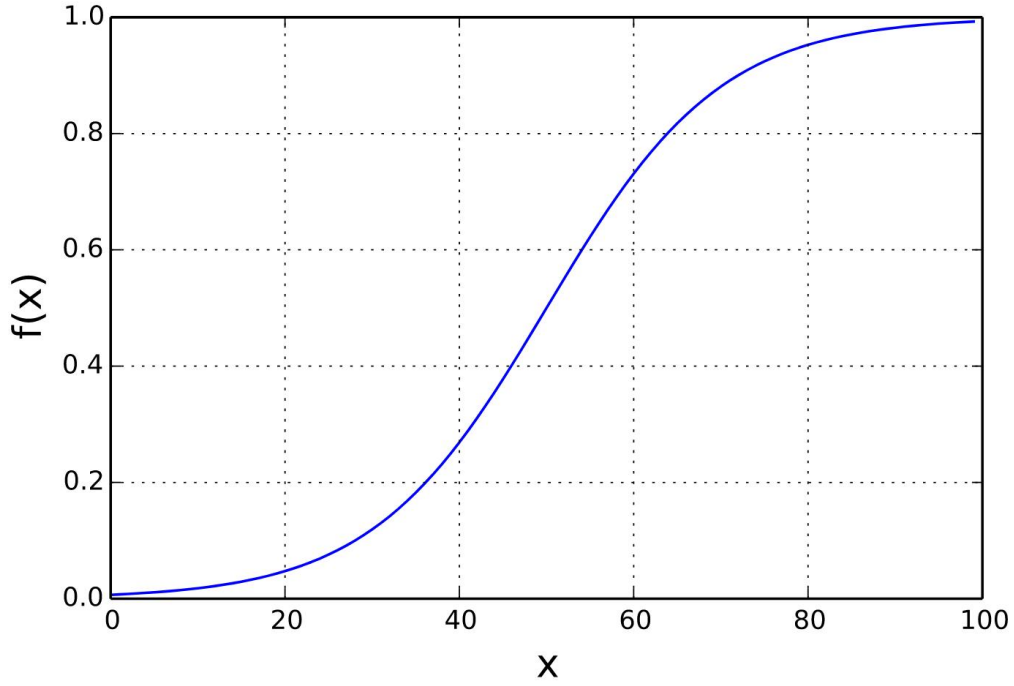
0.9

$\hat{y}_i$

$L(\hat{y}_i, y_i)$

GD

Recall: Activation function – Sigmoid / logistic



$$a_j^i = f(x_j^i) = \frac{1}{1 + \exp(-x_j^i)}$$

The **sigmoid** or **logistic** activation function maps the input values in the range (0,1), which is essentially their *probability* of belonging to a class.

It is mostly used for binary-class classification.
However, it suffers from the vanishing gradient problem. Also, its output is not zero-centered, which causes difficulties during the optimization step. It also has a low convergence rate.

Loss function for binary classification

Binary Cross Entropy Loss

$$L(\hat{y}_i, y_i) = -[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

$$L(\hat{y}, y) = -\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

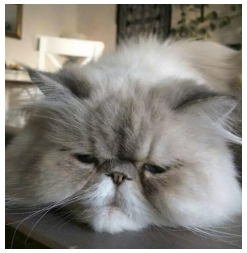
- If $\hat{y}_i$ is far away from y_i, we have large loss.
- If $\hat{y}_i$ is close to y_i, we have small loss.
- If $\hat{y}_i$ is just as y_i, we have zero loss.

- Minimize *L* is maximize log-likelihood *P(y|x)*

Proofs see Textbook - Murphy, Machine Learning: A Probabilistic Perspective (PP249)

Classification Task: Detect what is in an image

Data:
labeled image dataset $\{(x_i, y_i)\}_{i=1}^{N}$



One-hot vector 1000000 1000000 0100000

ImageNet

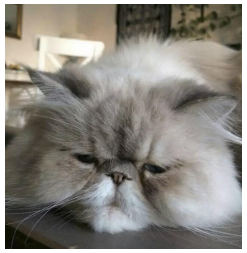
ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset".

There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+).

In ImageNet, we aim to provide on average 1000 images to illustrate each synset.

Classification Task: Detect what is in an image

Data:
labeled image dataset $\{(x_i, y_i)\}_{i=1}^{N}$


1000000


1000000


0100000

ImageNet

CIFAR-10: 60,000 32x32 colour images in 10 classes

CIFAR-100: 100 classes containing 600 images each

MNIST: 70,000 28x28 handwritten digits

Fashion MNIST: 28x28 fashion items (10 classes)

29954 citations for MNIST
21282 citations for ImageNet
8847 citations for CIFAR

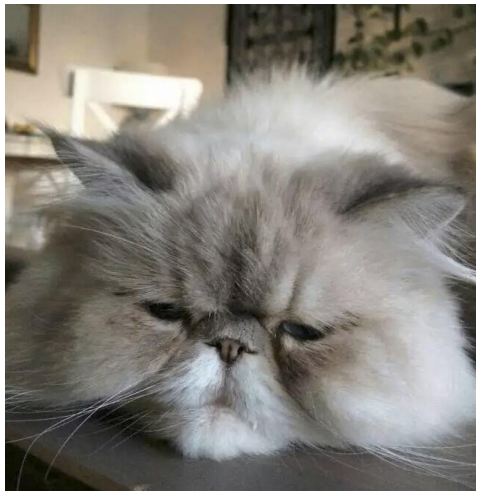
(Labeled) Data is extremely important for DL!!!

Classification Task: Detect what is in an image

Data:
labeled image dataset

$$L(y_i, y_i)$$

INPUT x_i



$$\sigma(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

OUTPUT

$$y_i = 1000000$$

Neural Network

softmax → 0.7 0.1 0.2 ...

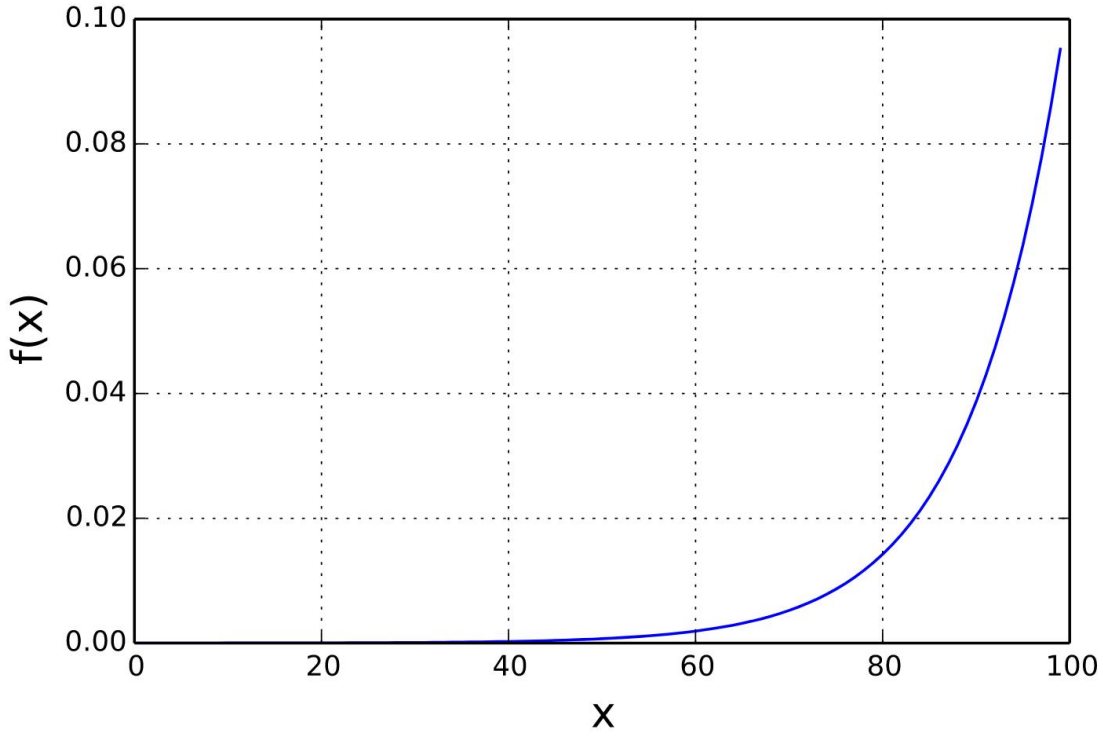
$$\hat{y}_i = f(x_i; \boldsymbol{\theta})$$

$\hat{y}_i$

$L(\hat{y}_i, y_i)$

GD

Recall: Activation function – softmax



$$a_j^i = f(x_j^i) = \frac{\exp(z_j^i)}{\sum_k \exp(z_k^i)}$$

The **softmax** function gives us the *probabilities* that any of the classes are true.
It produces values in the range (0,1).
Its values always sum to 1.

It highlights the largest value and tries to suppress values which are below the maximum value.
This function is widely used in multiple classification / logistic regression models.

Loss function for multiclass classification

Categorical Cross Entropy Loss

$$L(\hat{y}_i, y_i) = -\sum_{k=1}^{K} [y_{ik} \log(\widehat{y_{ik}})]$$

$$L(\hat{y}, y) = -\sum_{i=1}^{N} \sum_{k=1}^{K} [y_{ik} \log(\widehat{y_{ik}})]$$

- If $\widehat{y_{ik}}$ is far away from y_{ik}, we have large loss.
- If $\widehat{y_{ik}}$ is close to y_{ik}, we have small loss.
- If $\widehat{y_{ik}}$ is just as y_{ik}, we have zero loss.

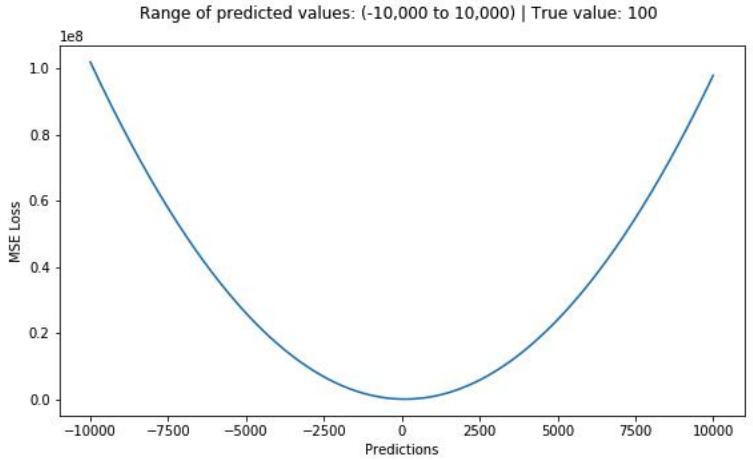
- Minimize L is maximize log-likelihood

y_i	$\hat{y}_i$
1	0.7
0	0.1
0	0.2
0	0
0	0

Loss function for linear regression

- Mean Squared Error (MSE, L2 loss, or Quadratic loss)

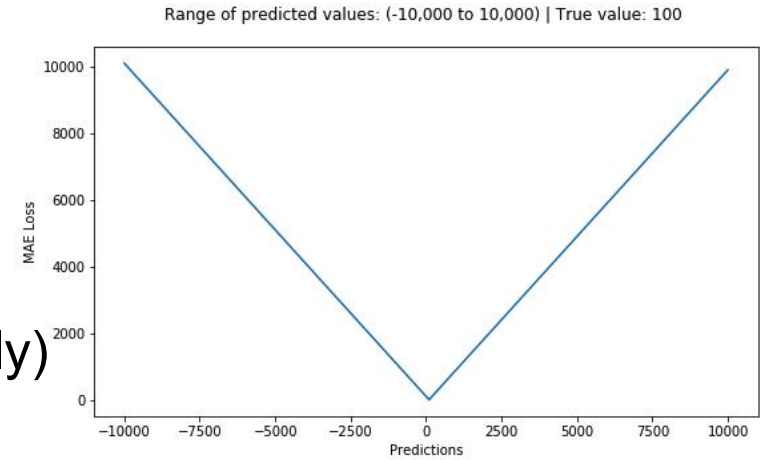
$$L(\hat{y}, y) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 / N$$



Plot of MSE Loss (Y-axis) vs. Predictions (X-axis)

- Mean Absolute Error (MAE, L1 loss)

$$L(\hat{y}, y) = \sum_{i=1}^{N} |\hat{y}_i - y_i| / N$$



Plot of MAE Loss (Y-axis) vs. Predictions (X-axis)

One big problem in using MAE loss (for neural nets especially) is that its gradient is the same throughout, which means the gradient will be large even for small loss values.
This isn't good for learning.

Lecture 18 – summary

1. Parameter number and memory size in CNN
   - Examples with some well-known CNNs (LeNet, AlexNet, VGG-16)
2. Three components of CNN
   - Objective function / loss function (cross-entropy loss, mse loss…)
   - Learning rule (Adagrad, Momentum, Adam…)
   - Network architecture (VGG, ResNet, U-net…)
3. Course projects

Course Project

- Find your teammates, **form a team**, discuss about your course project, and read 4 papers relevant to your course project (until May 10)
- **Do** the course project (until May 24)
- **Finalize** the course project (until May 26)
- The final presentation will be a **poster** presentation next to 大榕树 (on June 1)