



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

DDA4220 Deep Learning and Applications

Lecture 4 Computer Vision Basic --- Part I

Ruimao Zhang

zhangruimao@cuhk.edu.cn

School of Data Science

The Chinese University of Hong Kong (Shenzhen)

Thanks to the openmmlab team to provide the material

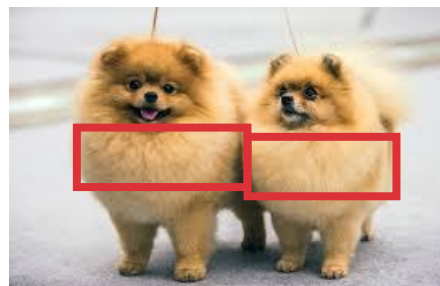


What is computer vision

- Computer Vision is the discipline in which computers learn to "see", the study of how to automatically understand what is in images and videos.



What is the animal in the picture?



Where is the dog located?



What is the dog doing?

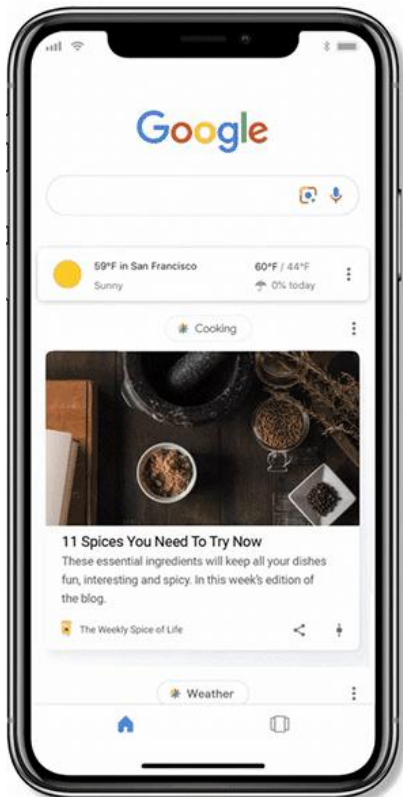


Generate a picture of a dog

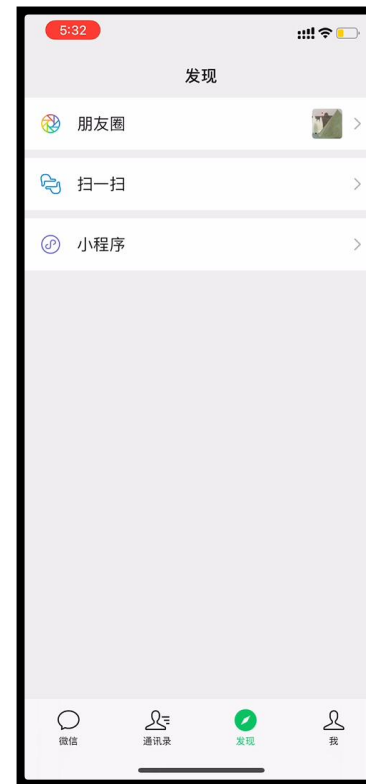


Computer vision in our everyday life

- **Image Recognition:** Recognizing what an object is in an image



Dog

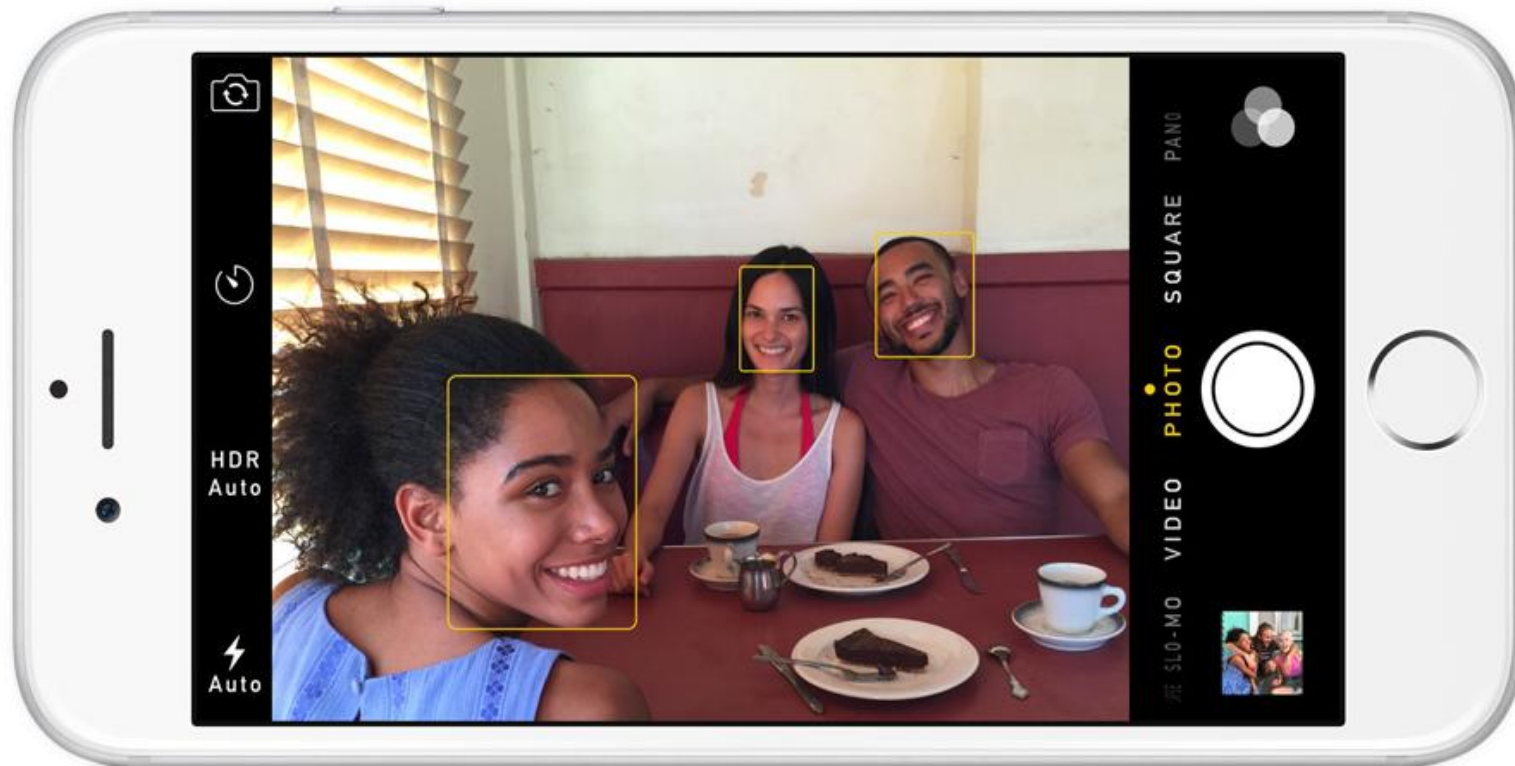


Cola



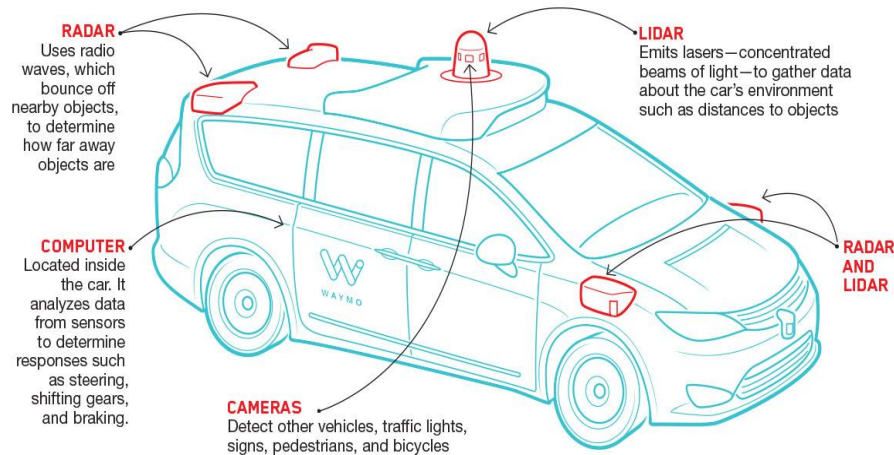
Computer vision in our everyday life

- **Face Recognition:** Identify the faces in the photo and locate them

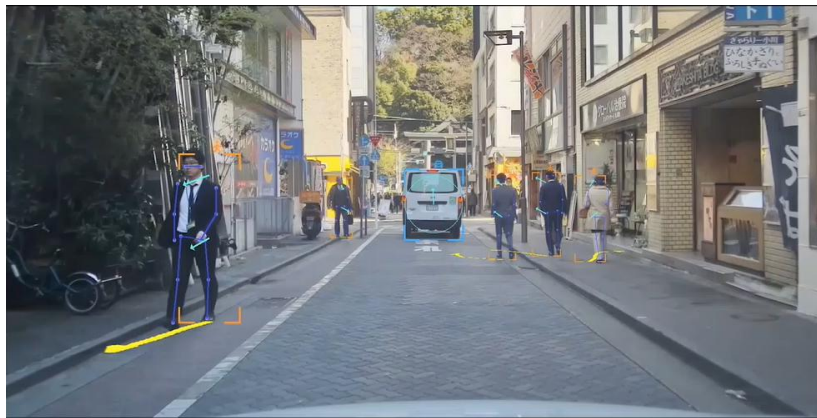




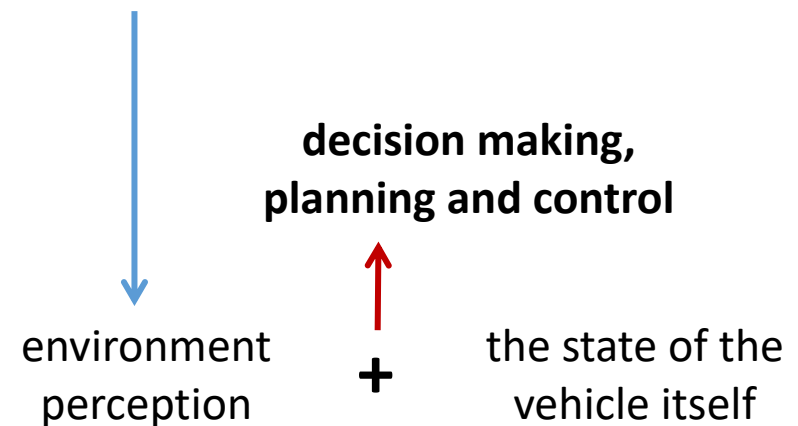
Computer vision in our everyday life



lidar based analysis



camera based analysis





Computer vision in our everyday life

- **Image Style Transfer:** Generate an image with similar content but different style from an image

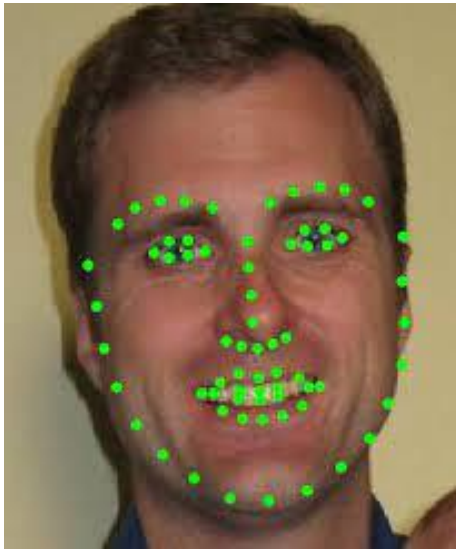


Anime Character Generation

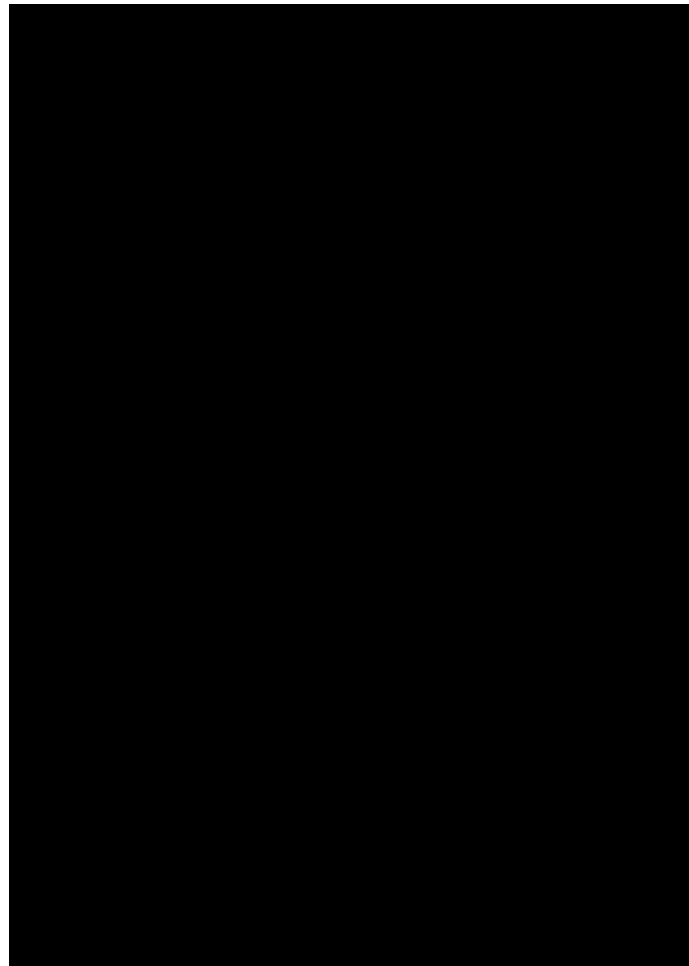


Computer vision in our everyday life

- **Landmark detection:** Aim to detect and track keypoints from a human face.



Face Key Point Recognition
+ Computer Graphics





Computer vision in our everyday life

- Video Understanding and Automatic Editing



Rio Olympics 2016 – Man's 3M Spring Board Final



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

The Development of Computer Vision

----- From arduous exploration to massive application

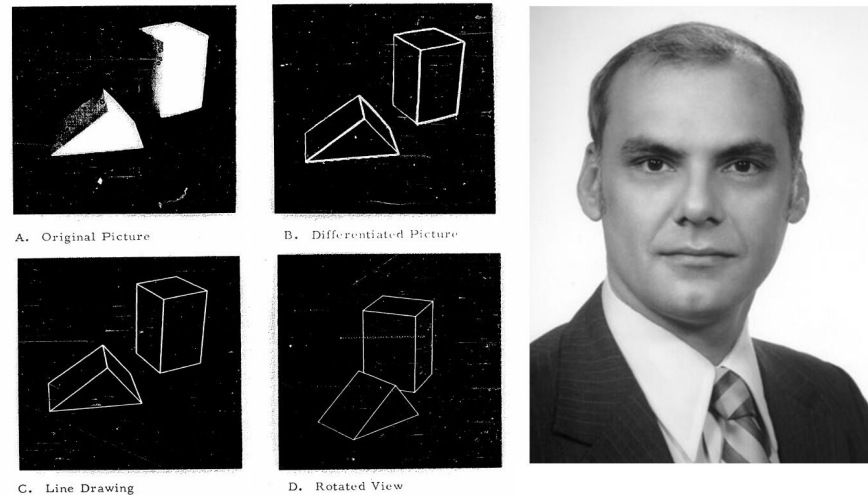




Early Germination (1960~1980)

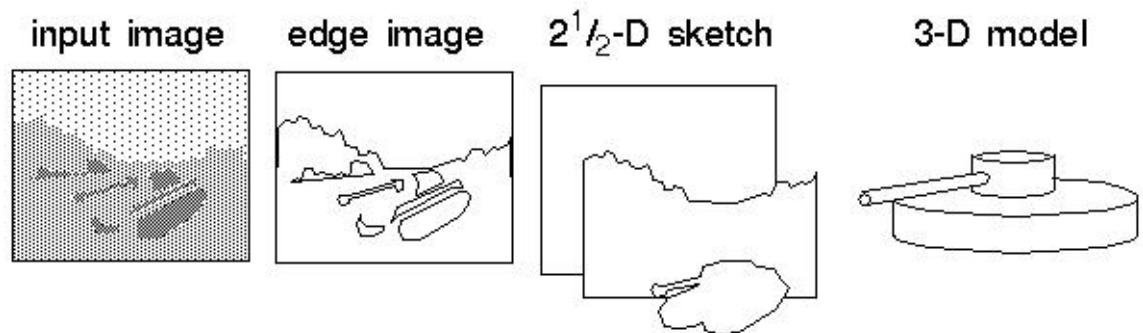
Machine perception of 3d solids

Larry Roberts 1964



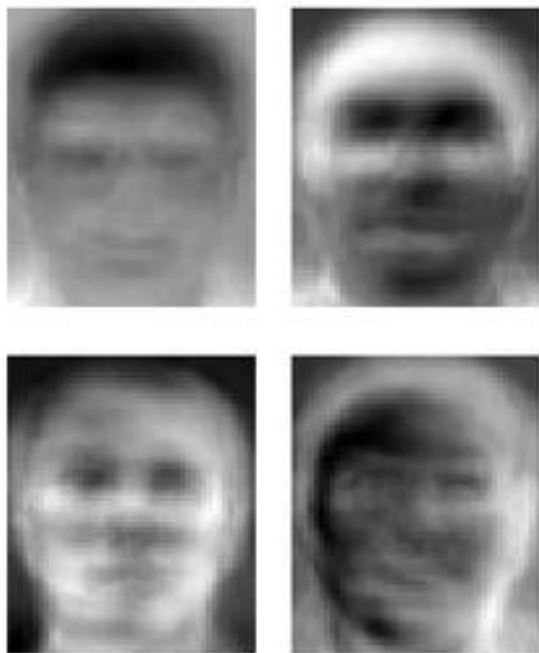
3D Visual Computing Theory

David Marr 1982





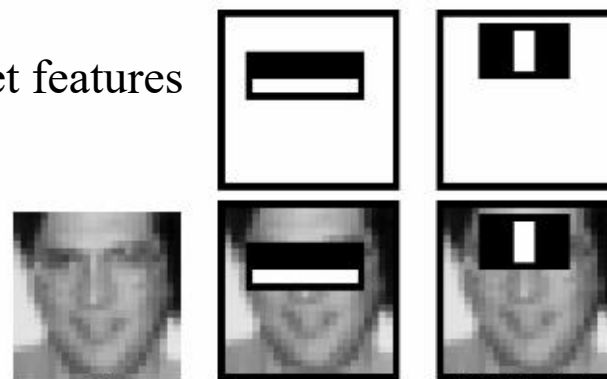
Statistical Machine Learning (1990~2000)



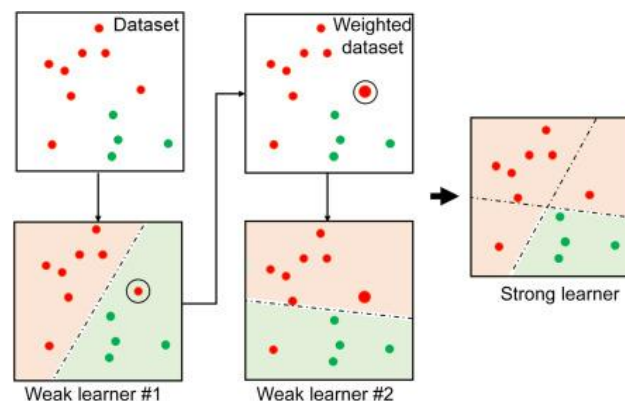
Eigen Face

Turk & Pentland 1991

Haar wavelet features



Adaboost Cascade Classifiers



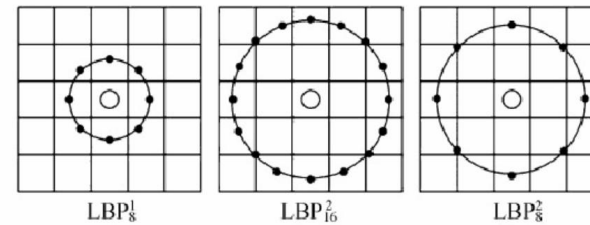
VJ face detection

Viola & Jones 2001

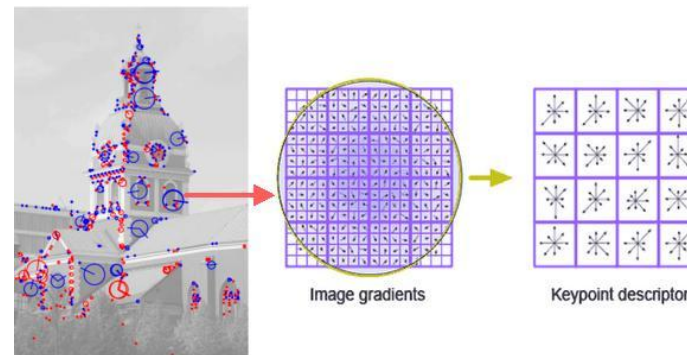


Visual feature representation

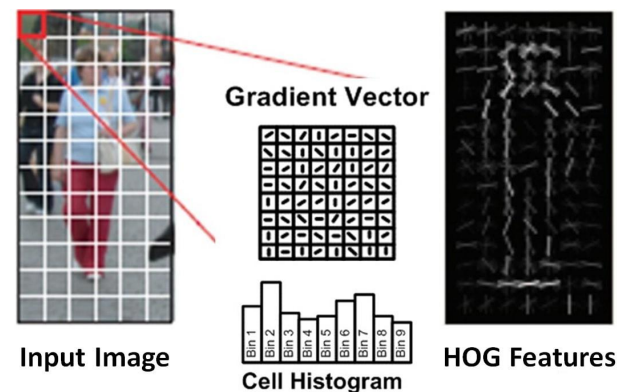
Local Binary Pattern



Scale Invariant Feature Transform



Histogram of Oriented Gradients



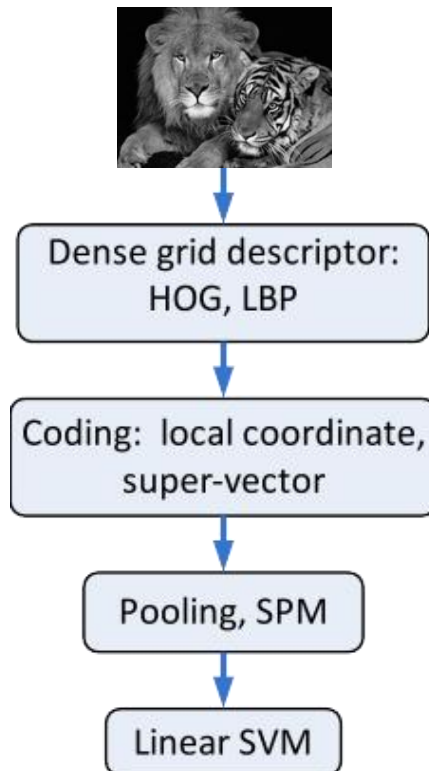
ImageNet Large Database (2006)



- Prof. Fei-Fei Li from Stanford University launched the ImageNet project in 2006, aiming to provide a large-scale, high-quality image database for computer vision research. ImageNet so far contains about **20,000 categories and a total of about 15 million images**.
- Since 2010, ImageNet has officially held the annual large-scale visual recognition challenge ILSVRC. The image classification track requires the participating teams to complete image classification tasks on a subset of ImageNet containing 1,000 categories and 1 million images.



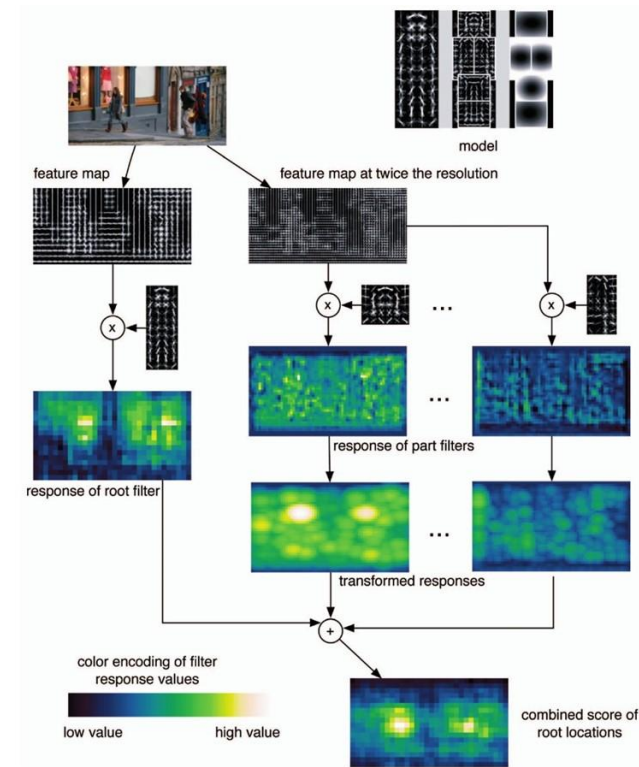
Vision system with initial success (~2010)



ImageNet Classification

72% Top-5 Accuracy

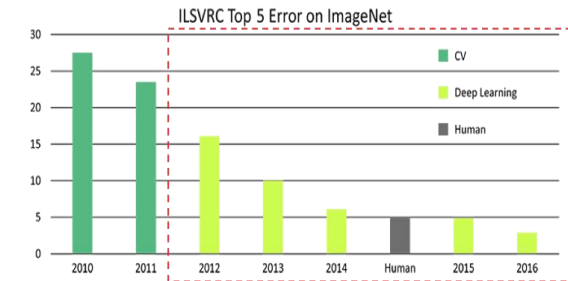
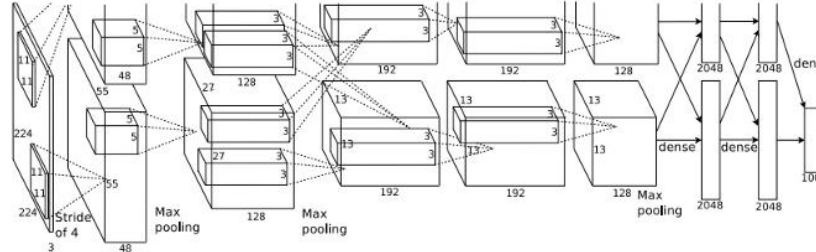
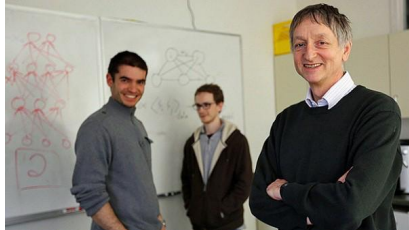
NEC-UIUC 2010



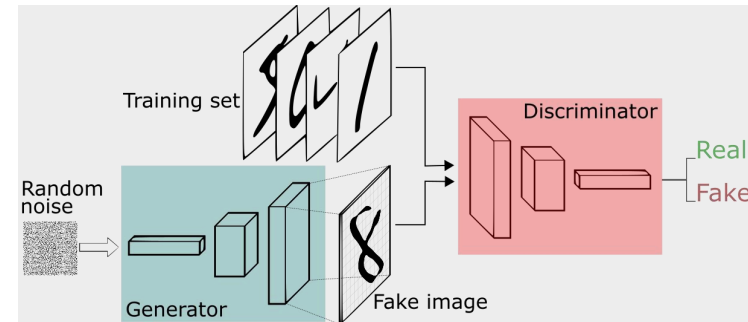
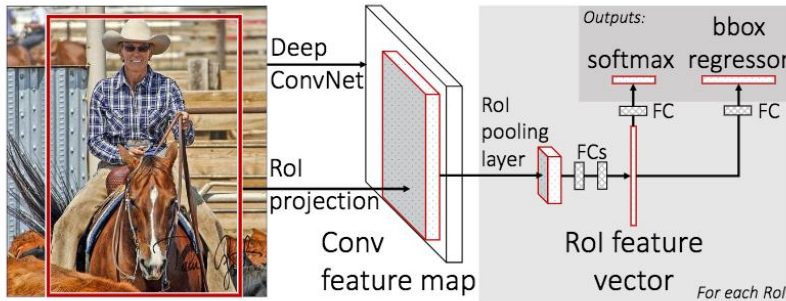
Deformable Part Model
for object detection



The era of deep learning (2012~)



AlexNet breaks through traditional vision system performance



Fast R-CNN: object detection enters the era of deep learning

Deep Generative Adversarial Networks achieves Image Generation



Nowadays

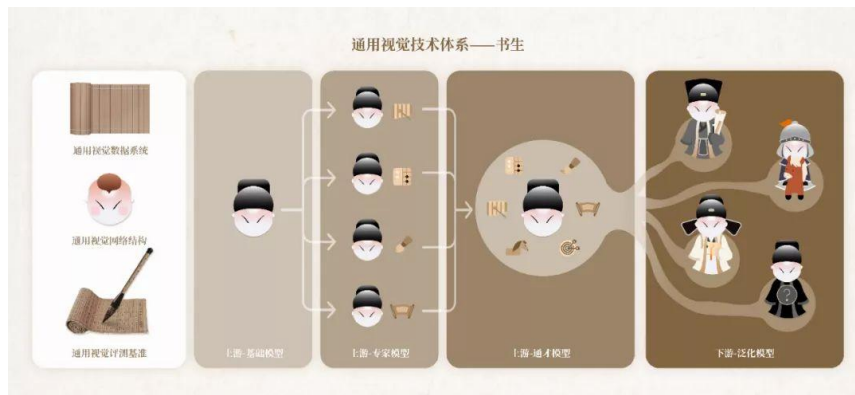
Image generation based on the text description



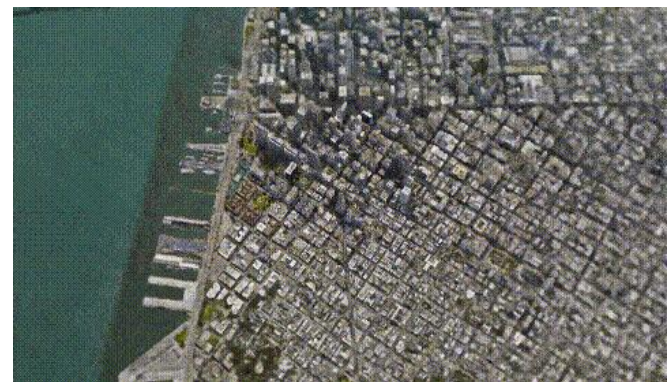
A glass whale lying among the ruins,
abstract painting



An astronaut riding a horse
in a photorealistic style



Large scale vision deep models

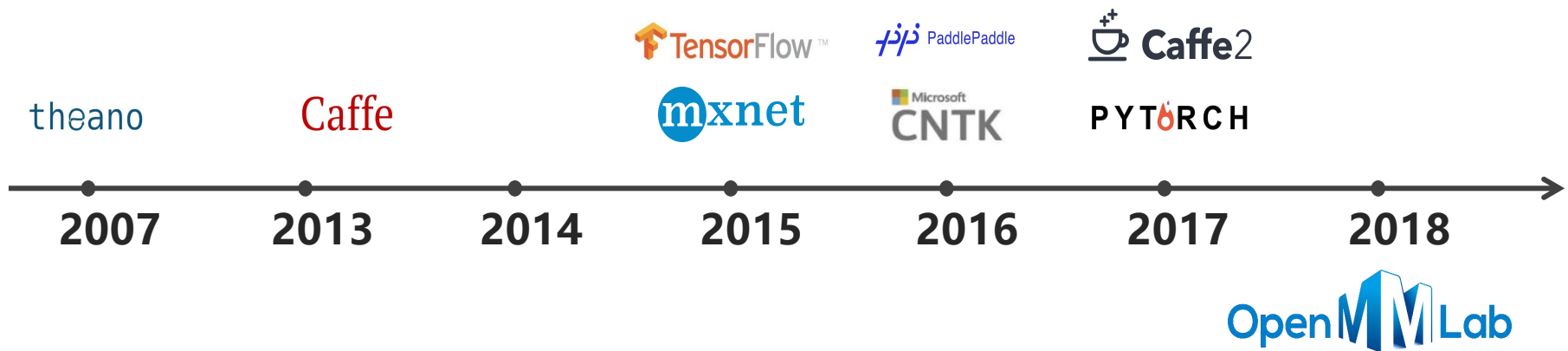


Neural Rendering CityNeRF



Open source becomes the new engine in AI

With the development of deep learning and computer vision, major research institutions and companies have successively open-sourced their own deep learning frameworks, and open-source code has become a new habit and consensus in the relevant research community when papers are published



Unified Deep Learning Framework



Unified Algorithm Framework and Ecology



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Image Classification and Training Strategies





What is image classification

Task Objective: Given a picture, identify what the object in the image is





Image classification

- An image is a matrix of pixels which can be presented as $X \in \mathbb{R}^{H \times W \times 3}$
- Give the class ID to each category: banana-->1, apple-->2, orange-->3, the class set $y \in \{1, \dots, K\}$, K is the total number of categories.
- Image classification: Construct a computationally implementable function F that map the input raw image to the class ID, and the predicted results are consistent with human perception



=

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 34 | 5 | 14 | ... | 75 | 57 | 56 |
| 35 | 55 | 88 | ... | 56 | 34 | 45 |
| 55 | 84 | 78 | ... | 75 | 76 | 80 |
| ... | ... | ... | ... | ... | ... | ... |
| 78 | 84 | 91 | ... | 142 | 142 | 187 |
| 88 | 58 | 159 | ... | 124 | 156 | 178 |
| 98 | 149 | 164 | ... | 154 | 156 | 226 |

$$y = F(X)$$



$$y = 1$$

Category: banana



Difficulties in visual tasks

The content of the image is the result of the presentation of all pixels and is not directly related to the value of individual pixels, making it difficult to follow specific rules for designing algorithms

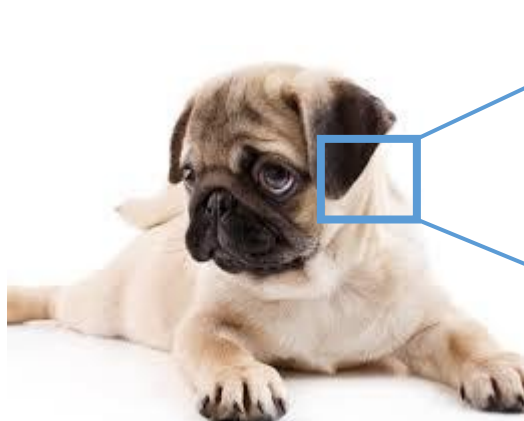


Image seen by
the human eye

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 135 | 135 | 129 | 133 | 130 | 134 | 134 | 137 |
| 133 | 133 | 132 | 132 | 135 | 127 | 123 | 119 |
| 132 | 127 | 129 | 115 | 121 | 87 | 96 | 110 |
| 110 | 104 | 115 | 109 | 120 | 103 | 129 | 160 |
| 105 | 112 | 136 | 162 | 173 | 201 | 219 | 231 |
| 167 | 187 | 202 | 223 | 216 | 231 | 240 | 238 |
| 221 | 231 | 240 | 223 | 214 | 216 | 218 | 219 |
| 224 | 217 | 222 | 214 | 215 | 217 | 219 | 220 |

The image seen
by the machine

?

→ Dog



Beyond the Rules: Machines Learn from Data

1. Collecting the data



dog



dog

.....



cat

X

y

2. Define the model

Functions with learnable parameters:

$$y = F_{\Theta}(X)$$

3. Model Training

Finding the best parameters Θ^* , making the model $y = F_{\Theta^*}(X)$ achieve the highest accuracy on the training set

4. Prediction

Given a new image \hat{X} , predicting its class labels as $\hat{y} = F_{\Theta^*}(\hat{X})$



Traditional method: Designing features (1990s~2000s)

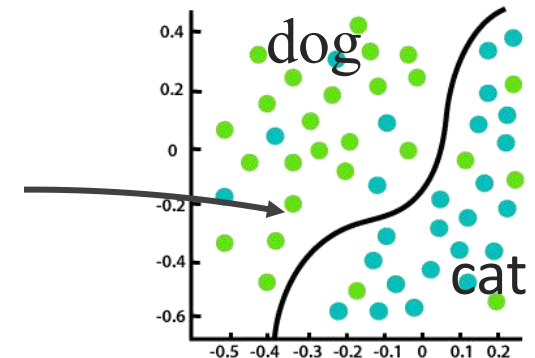
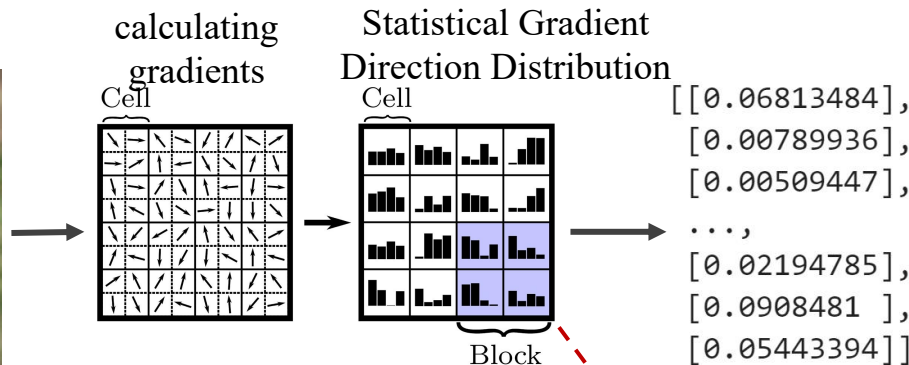
manually designed algorithms

machine learning

Image

Feature Vector

Classification



Hundreds of thousands of dimensions

visualization

HOG (Histogram of Oriented Gradients): the distribution of the direction of the pixel gradient is computed in the local area, and the image is mapped into a relatively low-dimensional feature vector while retaining enough information to identify the object.

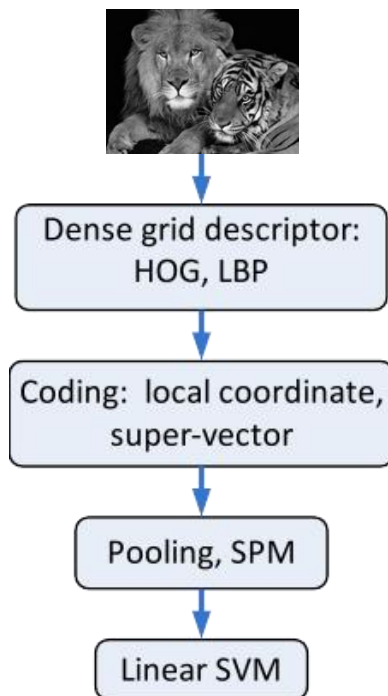


Good features:

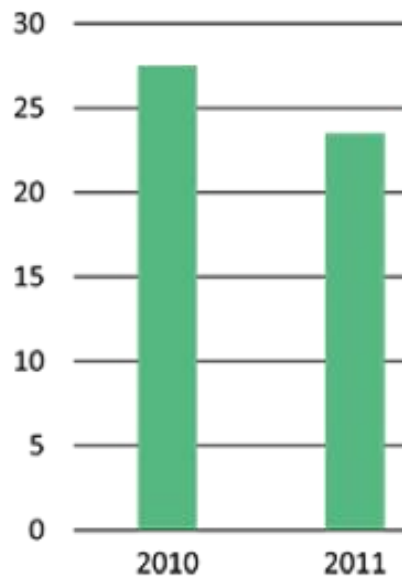
- Greatly simplifies data representation
- Keep information about content

The limitation of the feature engineering

In the ImageNet challenge, the champion teams of 2010 and 2011 adopted classic visual methods, based on hand-crafted features + machine learning algorithms to realize image classification, and the Top-5 error rate was around 25%.



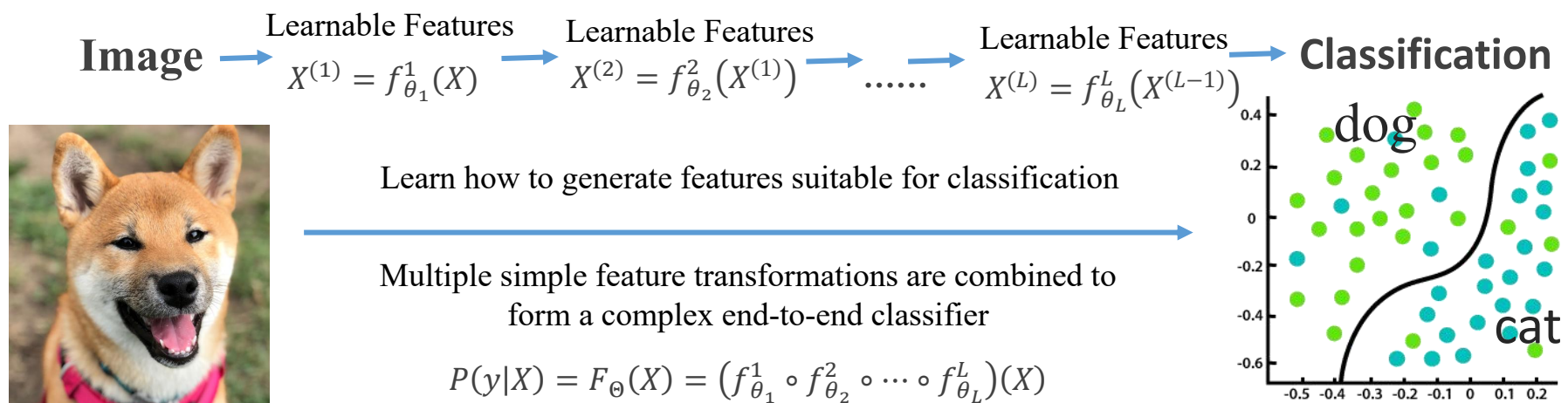
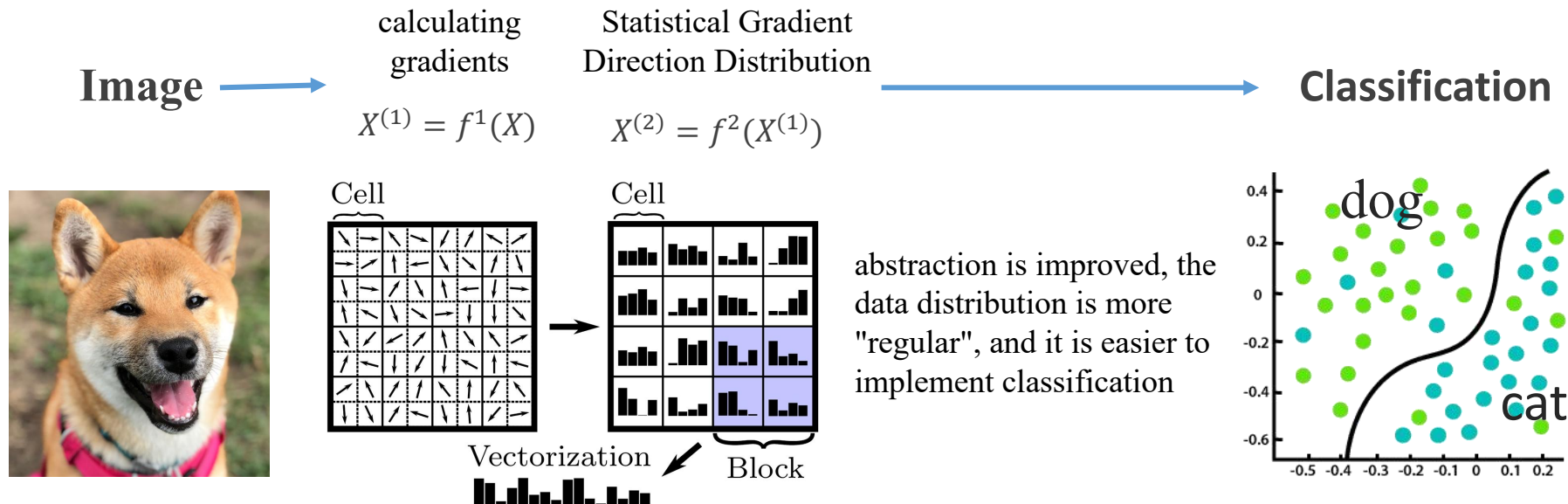
Top-5 Error Rate of Champion Method



Limited by human intelligence, the limitation of hand-designed features is that too much information is lost, and the performance on visual tasks reaches the bottleneck



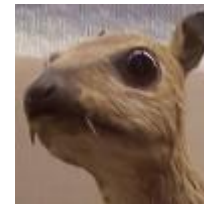
From feature engineering to feature learning



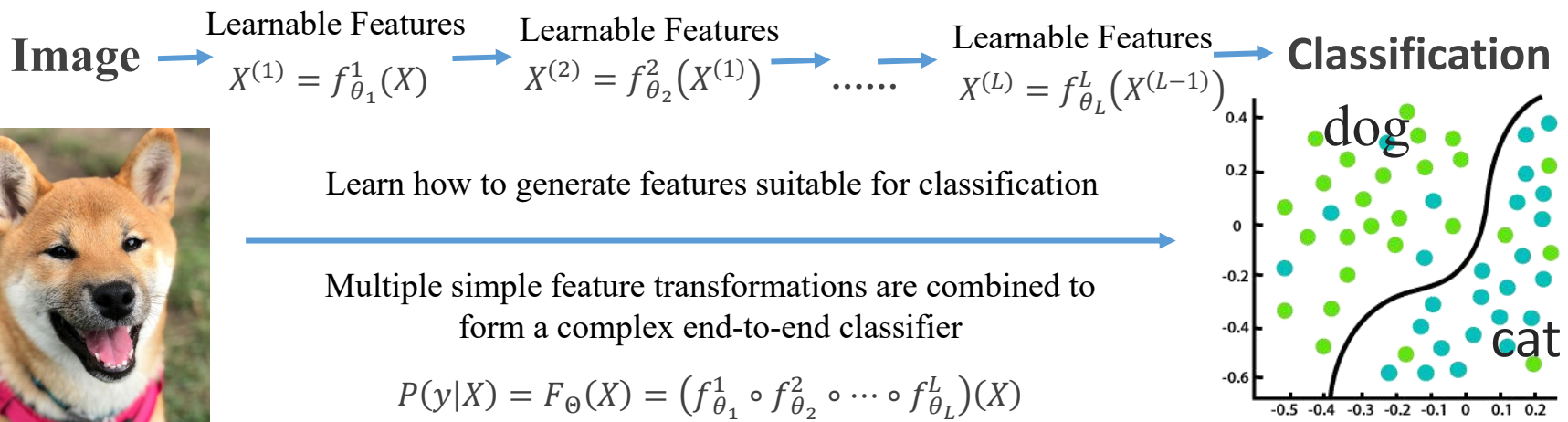
Implementation of hierarchical features

Convolution ----> Convolutional Neural Network

- Feature maps have the same two-dimensional spatial structure as images
- The feature of the last layer is the weighted summation of the features of the previous layer in the spatial



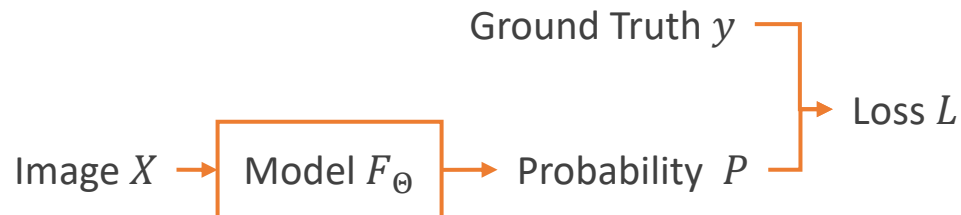
$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$





Model Training Overview

Supervised Learning



1. Label a dataset $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^N$
2. Define the loss function $L: [0,1]^K \times \mathbb{N} \rightarrow \mathbb{R}$, measuring the "good/bad" of the predictions
3. Solve an optimization problem and find the parameters Θ^* that minimize the overall loss

$$\Theta^* = \arg \min_{\Theta} \sum_{i=1}^N L(F_{\Theta}(X_i), y_i) \longrightarrow \text{Gradient Descent Methods + Tricks and Strategies for Vision Problems}$$



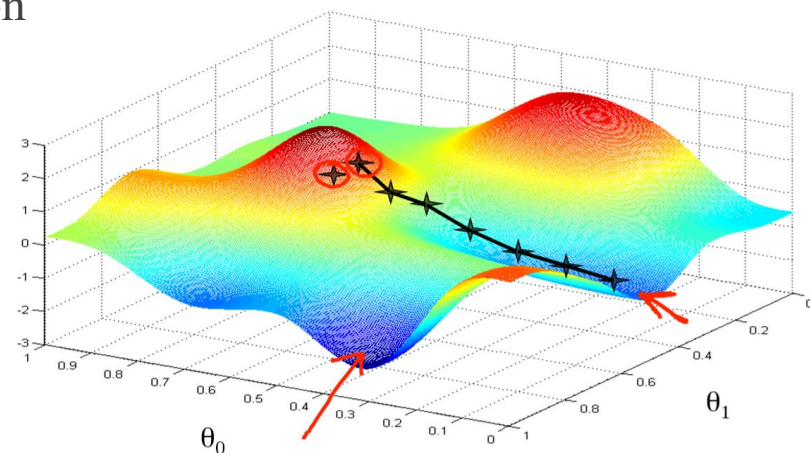
Gradient Descent Algorithm

Given the dataset $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^N$, the model $F_{\Theta}(X)$ and the loss function $L(P, y)$, the overall loss of the model on the given dataset is defined as::

$$L(\Theta|\mathcal{D}) = \frac{1}{N} \sum_{i=1}^N L(F_{\Theta}(X_i), y_i)$$

For the neural network, L is a non-convex function of Θ , the gradient descent algorithm (GD) is often adopted:

- Random initialization parameters $\Theta^{(0)}$
- Iterate until convergence:
 - Forward + backpropagation, calculate the gradient $\nabla_{\Theta} L(\Theta^{(t-1)})$
 - Update the parameters $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_{\Theta} L(\Theta^{(t-1)})$



Learning rate



Stochastic Gradient Descent

In standard gradient descent, each iteration needs to calculate the gradient for all samples, which is difficult on large data sets

$$\nabla_{\Theta} L(\Theta) = \frac{1}{N} \sum_{i=1}^N \nabla_{\Theta} L(F_{\Theta}(X_i), y_i)$$

N is on the order of tens of thousands to millions

Each iteration randomly selects a part of samples $\mathcal{B}^{(t)} = \{i_1, i_2, \dots, i_B\}$ to calculate the gradient as an approximation of the complete gradient

$$\nabla_{\Theta} L(\Theta) \approx \frac{1}{B} \sum_{i \in \mathcal{B}} \nabla_{\Theta} L(F_{\Theta}(X_i), y_i)$$

B take dozens to hundreds

The corresponding sample subset $\mathcal{D}_{\mathcal{B}} = \{(X_i, y_i)\}_{i \in \mathcal{B}} \subset \mathcal{D}$ is called the mini-batch, its size B is called the batch size



The momentum SGD

Core Idea: Continue the movement of the previous step to current step

Accumulation of gradients
from previous iterations

Calculate Gradient
Cumulative Value

$$\Delta^{(0)} := 0$$

Moving Average

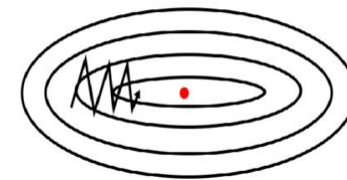
$$\Delta^{(t)} := \alpha \Delta^{(t-1)} + \nabla_w L^{(t)}$$

Moving according to
Accumulative Value

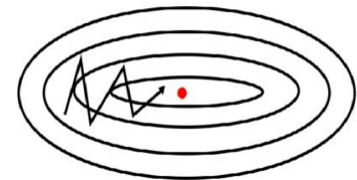
$$w^{(t)} := w^{(t-1)} - \eta \Delta^{(t)}$$

➤ Reduce fluctuations in
stochastic gradient descent

SGD without momentum

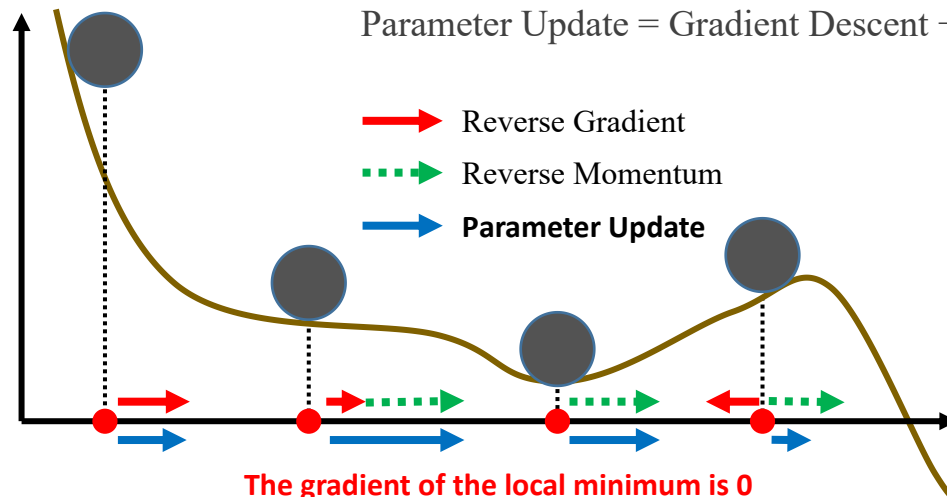


SGD with momentum



➤ Helps escape from local
minima and saddle points

Loss Curve



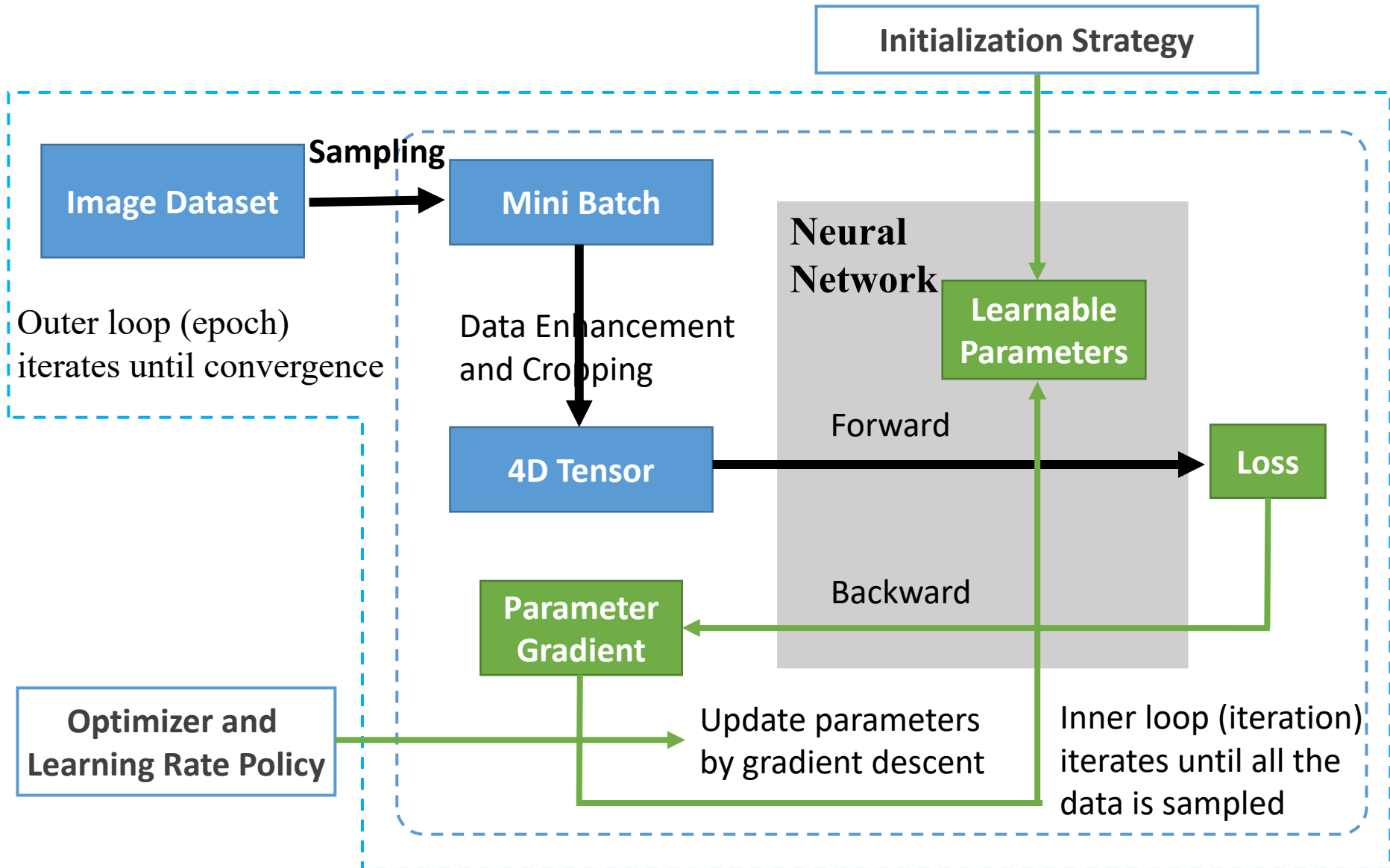
Parameter Update = Gradient Descent + Momentum

- ➔ Reverse Gradient
- ➔ Reverse Momentum
- ➔ Parameter Update

The gradient of the local minimum is 0

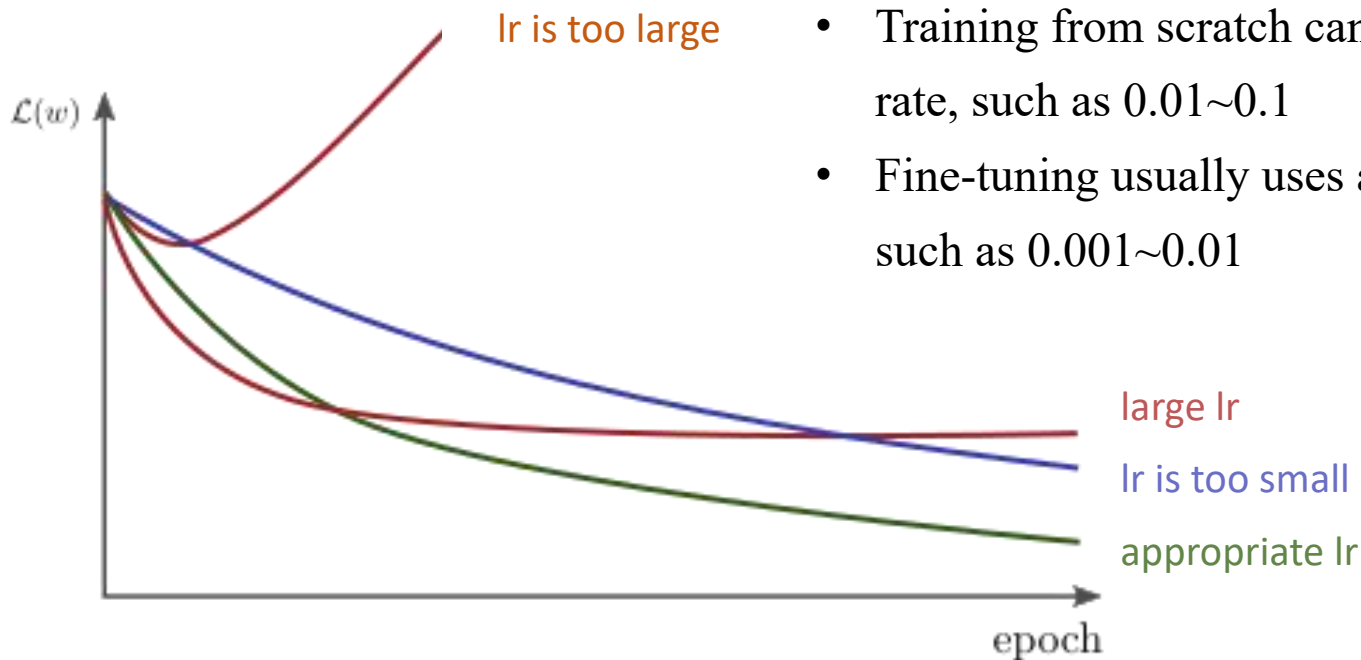
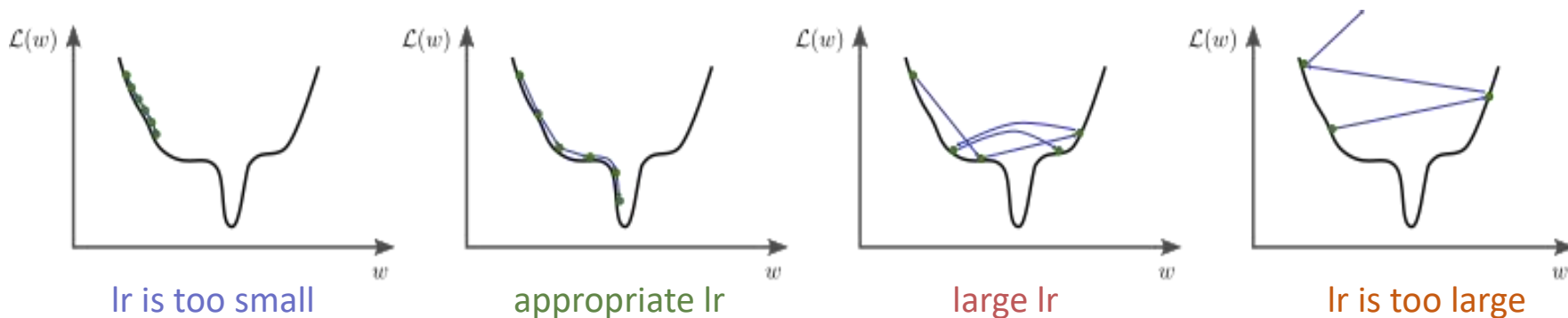


Model Training based on Stochastic Gradient Descent





Learning Rate and Optimizer



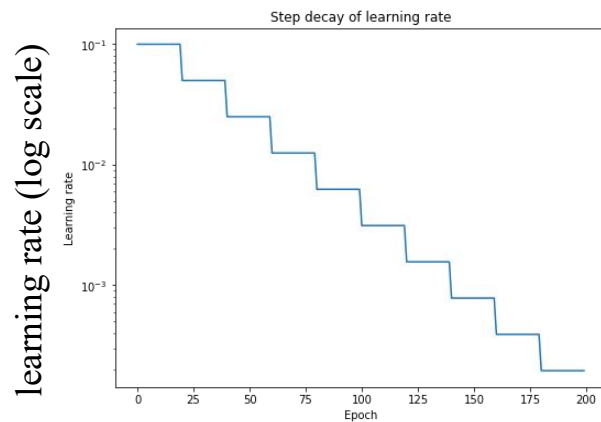
- Training from scratch can use a larger learning rate, such as 0.01~0.1
- Fine-tuning usually uses a small learning rate, such as 0.001~0.01



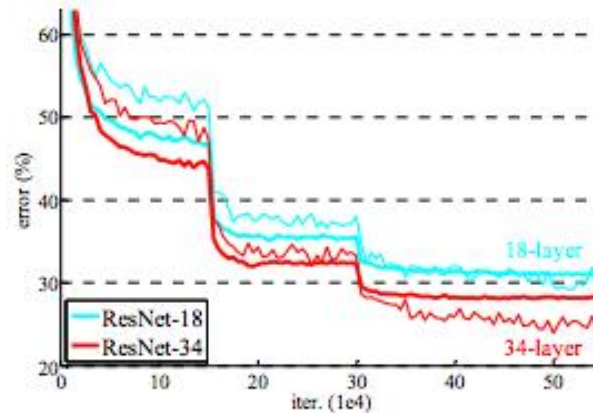
Learning Rate Annealing

Use a larger learning rate in the initial stage of training, and decrease the learning rate after the loss function stabilizes:

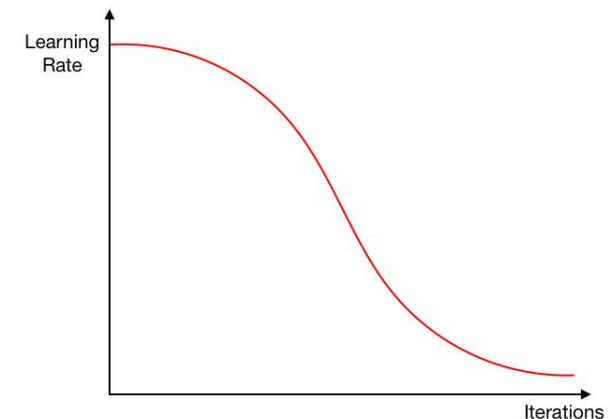
- Step down
- Proportionally down $\eta(t) = \eta_0 e^{-kt}$
- Down by reciprocal $\eta(t) = \frac{\eta_0}{\epsilon + t}$
- Down by cosine function
$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos \left(\frac{T_{\text{cur}}}{T_{\text{max}}} \pi \right) \right)$$



Reduce learning rate to 1/3 every 20 epochs



The loss function continues to decrease after reducing the learning rate

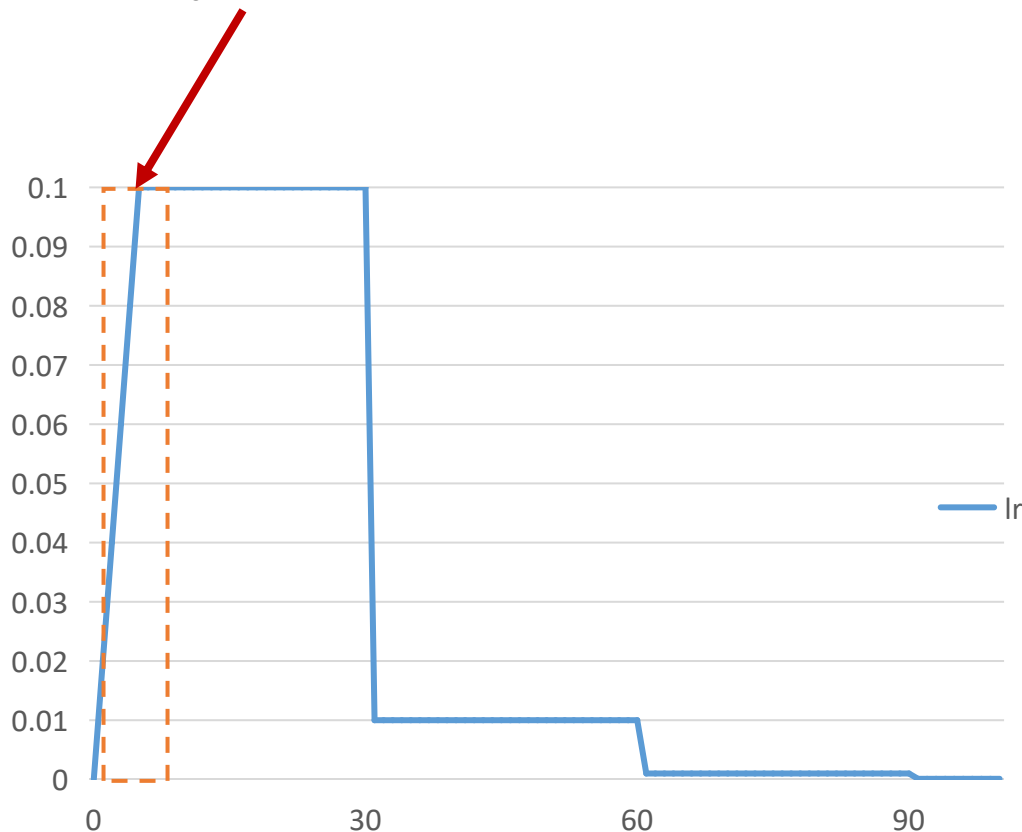


The learning rate decreases according to the cosine function

Learning Rate Warmup

The learning rate is gradually increased in the first few rounds of training until it reaches the preset learning rate to stabilize the initial stage of training

- linear increase $\eta(t) = \frac{t}{T_0} \eta_0$ ($t \leq T_0$)





Linear Scaling Rule

Empirical conclusion: For the same training task, when the batch size is expanded to k times, the learning rate should also be expanded by k times.

Intuitive understanding: Doing so can ensure that the average step size of the gradient descent brought by each sample is the same.

Assuming a total of kB samples,

1. Using batch size B , iterate k times:

$$w_{t+k} = w_t - \eta \sum_{i=0}^{k-1} \nabla L(w_{t+i} | \mathcal{B}_i)$$

2. Use batch size kB , iterate 1 time:

$$w_{t+k} = w_t - \eta \nabla L(w_t | \mathcal{B}_{\text{all}})$$

If it is assumed that all $\nabla L(w_{t+i} | \mathcal{B}_i)$ are approximately equal, the learning rate η in scheme 2 should be k times that in scheme 1 to ensure that the two w_{t+k} are the same

In practice, it is assumed that the pre-training model uses $\text{lr}=0.1$, and 8 GPUs are trained in parallel. If you want to use 1 GPUs to reproduce the experiment, lr should be set to 0.0125



Adaptive Gradient Algorithm

Core Idea: Different parameters require different learning rates, and the learning rate is automatically adjusted according to the historical magnitude of the gradient

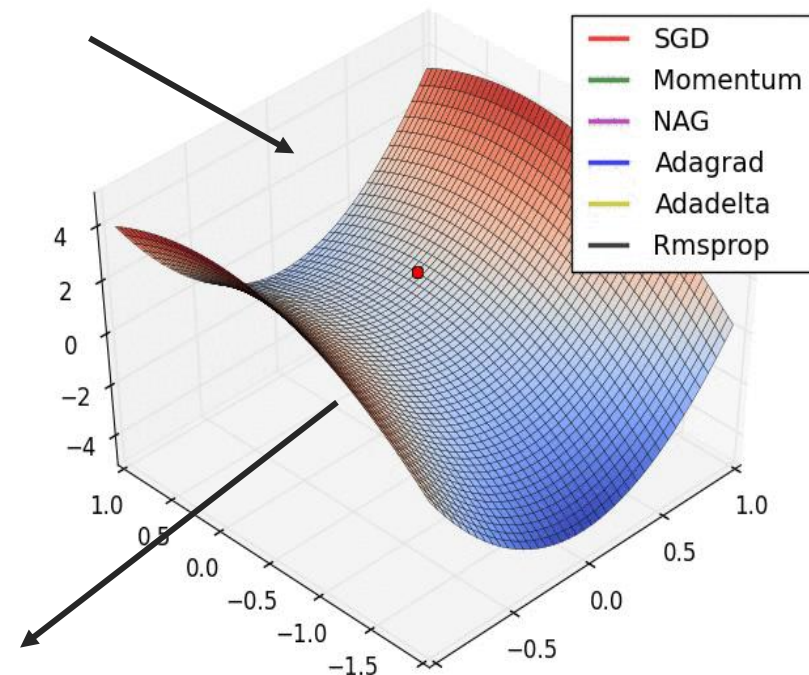
Adagrad

$$v_t^w = v_{t-1}^w + (\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w} + \epsilon} * \nabla w_t$$

In the X direction, the initial gradient is small and the learning rate is gradually increased during the training process

Adam / AdamW

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} * \hat{m}_t$$



In the Y direction, the initial gradient is large and the learning rate is gradually reduced during the training process



Adaptive Gradient Algorithm

Core Idea: Different parameters require different learning rates, and the learning rate is automatically adjusted according to the historical magnitude of the gradient

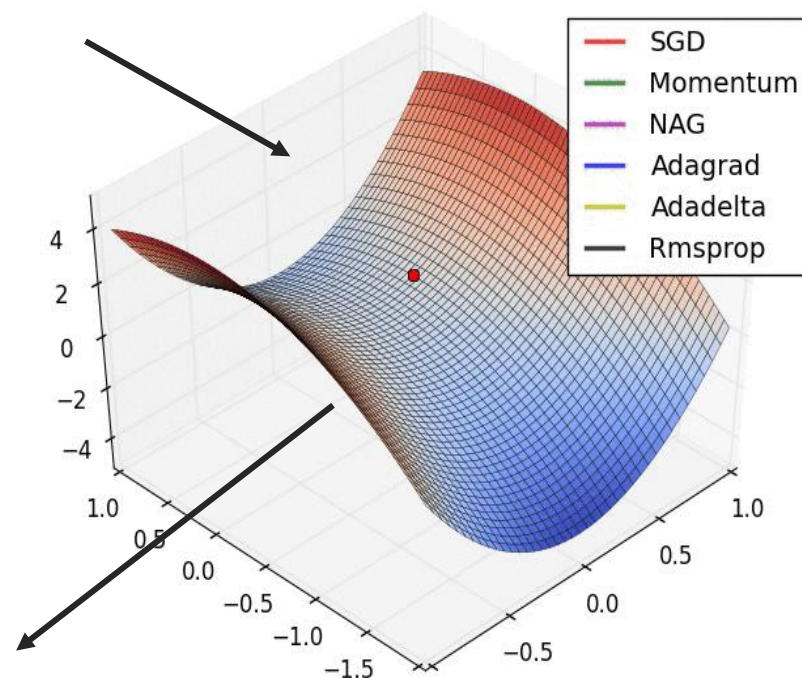
Adagrad

$$v_t^w = v_{t-1}^w + (\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w} + \epsilon} * \nabla w_t$$

In the X direction, the initial gradient is small and the learning rate is gradually increased during the training process

Adam / AdamW

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} * \hat{m}_t$$



In the Y direction, the initial gradient is large and the learning rate is gradually reduced during the training process



Regularization and Weight Decay

- Introduce a regularization term in the loss function to encourage training relatively simple models:

$$R(\Theta) = L(\Theta|\mathcal{D}) + \frac{1}{2} \lambda \|\Theta\|_2^2$$

- Gradient of structural risk:

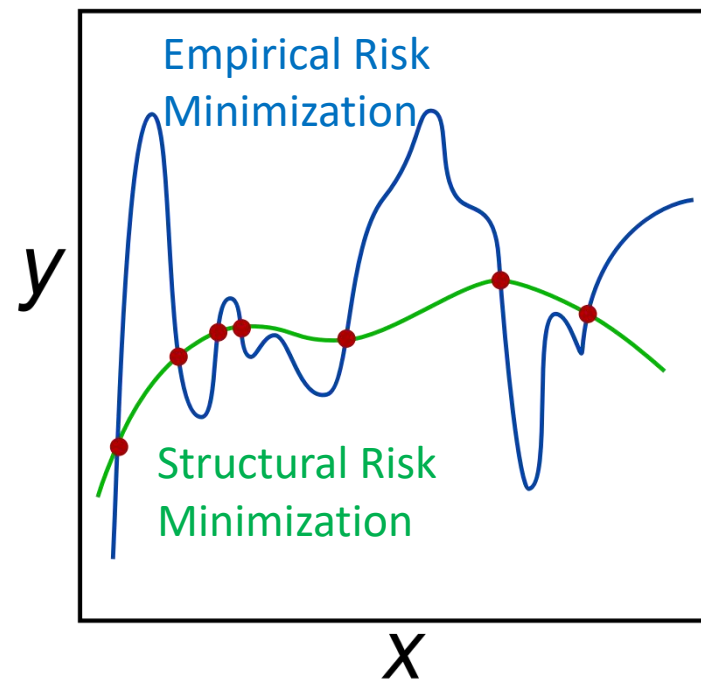
$$\nabla_{\Theta} R = \nabla_{\Theta} L + \lambda \Theta$$

- Gradient update strategy:

$$\begin{aligned} \Theta^{(t)} &= \Theta^{(t-1)} - \eta \nabla_{\Theta} R(\Theta^{(t-1)}) \\ &= \Theta^{(t-1)} - \eta \nabla_{\Theta} L(\Theta^{(t-1)}) - \lambda \eta \Theta^{(t-1)} \\ &= \underbrace{(1 - \lambda \eta)}_{\text{Weight Decay}} \Theta^{(t-1)} - \underbrace{\eta \nabla_{\Theta} L(\Theta^{(t-1)})}_{\text{Regular Gradient Descent}} \end{aligned}$$

Weight Decay

Regular Gradient Descent

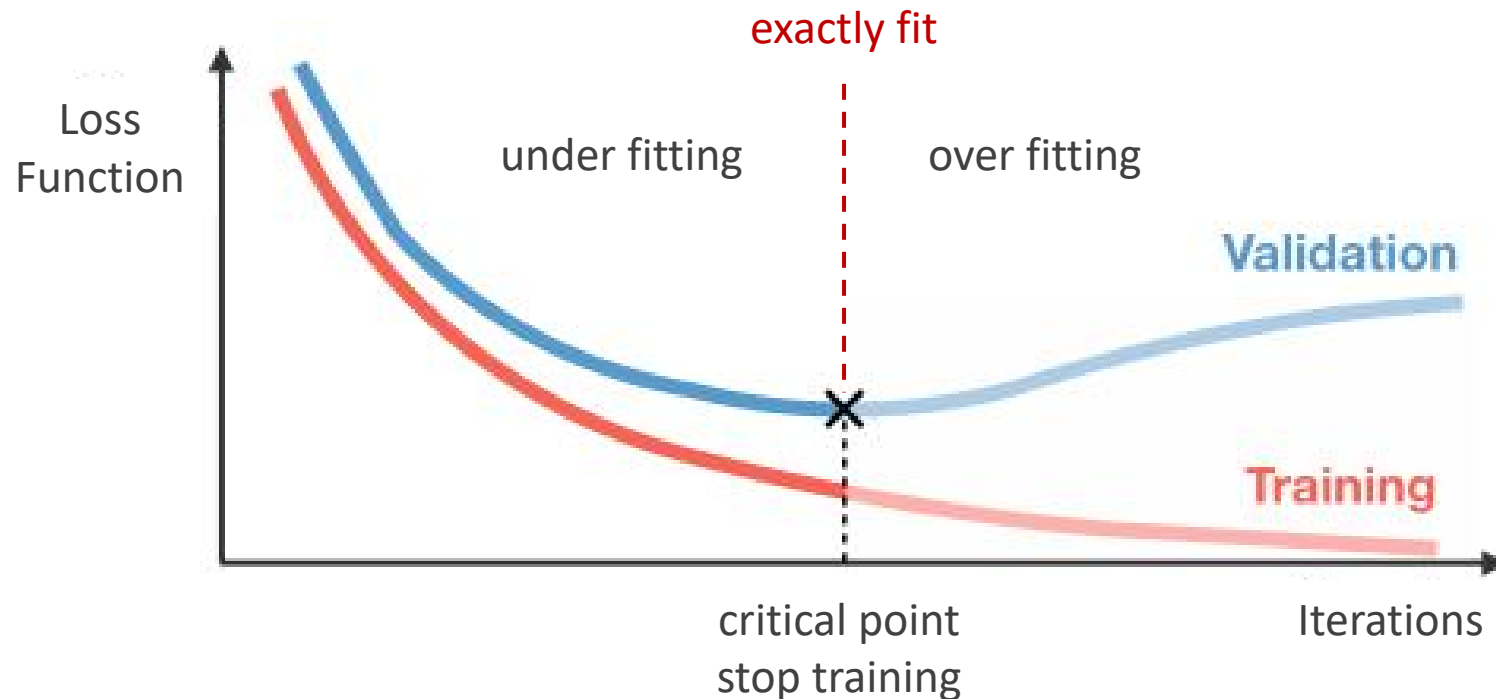


Note: The above equivalence is established for SGD / Momentum SGD, but not for adaptive algorithms such as Adam. AdamW proposes that weight decay should be performed independently of the normalization step, and the effect is better than Adam



Early Stopping

- Divide the training data set into a training set and a validation set, train on the training set, and periodically test the classification accuracy on the validation set
- When the classification accuracy of the verification set reaches the maximum value, stop training to prevent overfitting





Model Weight Averaging

- The queue of parameters produced by the gradient descent algorithm (Momentum SGD, Adam, etc.): $\Theta_0, \Theta_1, \dots, \Theta_t, \dots, \Theta_T$
- Exponential Moving Average

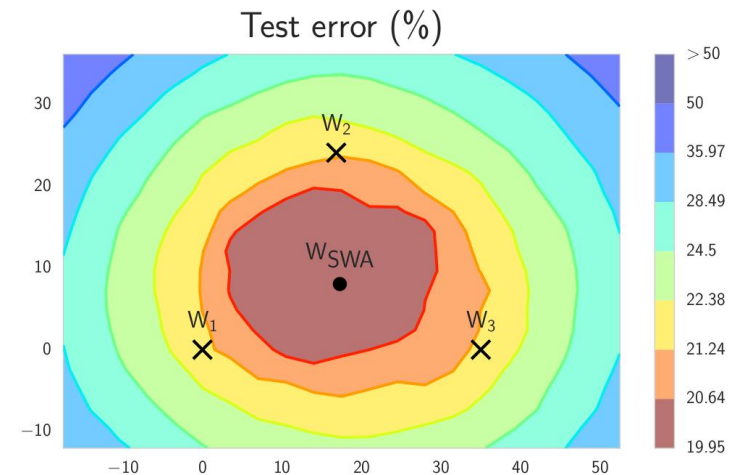
$$\bar{\Theta}_0 = \Theta_0$$

$$\bar{\Theta}_t = \beta \bar{\Theta}_{t-1} + (1 - \beta) \Theta_t$$

For example $\beta = 0.9998$,

You can also choose to update every few steps $\bar{\Theta}_t$

Basic assumption: the model will "rotate" around the minimum value at the end of optimization, and the average parameter is closer to the minimum value point





Data Augmentation

- Training a model with good generalization requires a large amount of diverse data, but the collection and labeling of data are costly
- Images can be simply transformed to generate a series of "replicas" to augment the training dataset.
- Data augmentation operations can be further combined to produce images with more complex variations

geometric
transformation



Original



Horizontal Flip



Pad & Crop



Rotate

color change



Original image



RGBShift



HueSaturationValue



ChannelShuffle

random
occlusion





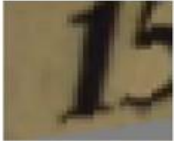















image classification



AutoAugment & RandAugment

AutoAugment

Searching for data-augmented combination strategies via reinforcement learning

| | Original | Sub-policy 1 | Sub-policy 2 | Sub-policy 3 | Sub-policy 4 | Sub-policy 5 |
|---------|---|---|---|---|---|---|
| Batch 1 |  |  |  |  |  |  |
| Batch 2 |  |  |  |  |  |  |
| Batch 3 |  |  |  |  |  |  |
| | | ShearX, 0.9, 7 Invert, 0.2, 3 | ShearY, 0.7, 6 Solarize, 0.4, 8 | ShearX, 0.9, 4 AutoContrast, 0.8, 3 | Invert, 0.9, 3 Equalize, 0.6, 3 | ShearY, 0.8, 5 AutoContrast, 0.7, 3 |

Search results on the SVHN dataset

Five groups of strategies, randomly select a group for each batch

Each group of policies contains two operations, and the corresponding operation is applied with a certain probability

AutoAugment & RandAugment

Rand Augment

Simpler search space: randomly select N combinations of data

augmentation operations with magnitude M

```
transforms = [  
    'Identity', 'AutoContrast', 'Equalize', 'Rotate', 'Solarize', 'Color', '  
    Posterize', 'Contrast', 'Brightness', 'Sharpness', 'ShearX', 'ShearY', '  
    TranslateX', 'TranslateY']  
  
def randaugment(N, M):  
    """Generate a set of distortions.  
  
    Args:  
    N: Number of augmentation transformations to apply sequentially.  
    M: Magnitude for all the transformations.  
    """  
  
    sampled_ops = np.random.choice(transforms, N)  
    return [(op, M) for op in sampled_ops]
```

Magnitude: 9

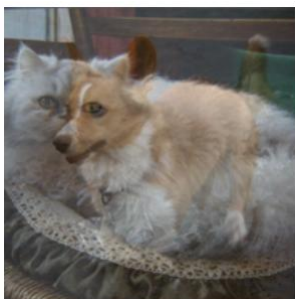




Mixup & CutMix

Mixup

Blend images pixel by pixel



$$\text{Image} = 0.5 * \text{DogImage} + 0.5 * \text{CatImage}$$

$$\text{Label} = \{\text{Dog: } 0.5, \text{Cat: } 0.5\}$$

CutMix

Cover the original image and fill it
with another image



$$\text{Image} = \text{Mask} * \text{DogImage} + (1 - \text{Mask}) * \text{CatImage}$$

$$\text{Label} = \{\text{Dog: } 0.6, \text{Cat: } 0.4\}$$



Label Smoothing

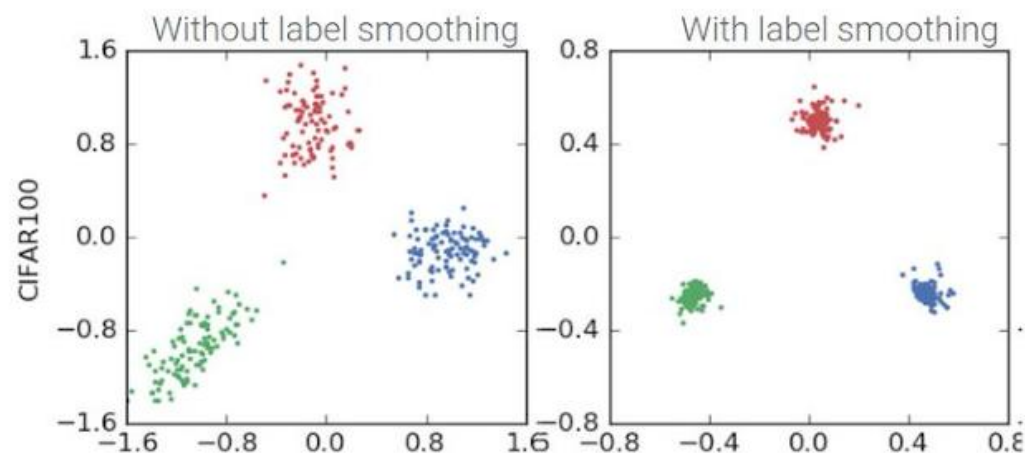
Motivation: Class labels may be wrong or inaccurate, and maximizing the model to fit the labeled categories may hinder generalization

Approach: Introduce the smoothing parameter ε to reduce the "confidence" of the label

Traditional one-hot labeling: $P_i = \begin{cases} 1, & i = y \\ 0, & i \neq y \end{cases}$

label smoothing: $P_i = \begin{cases} 1 - \varepsilon, & i = y \\ \frac{\varepsilon}{K-1}, & i \neq y \end{cases}$

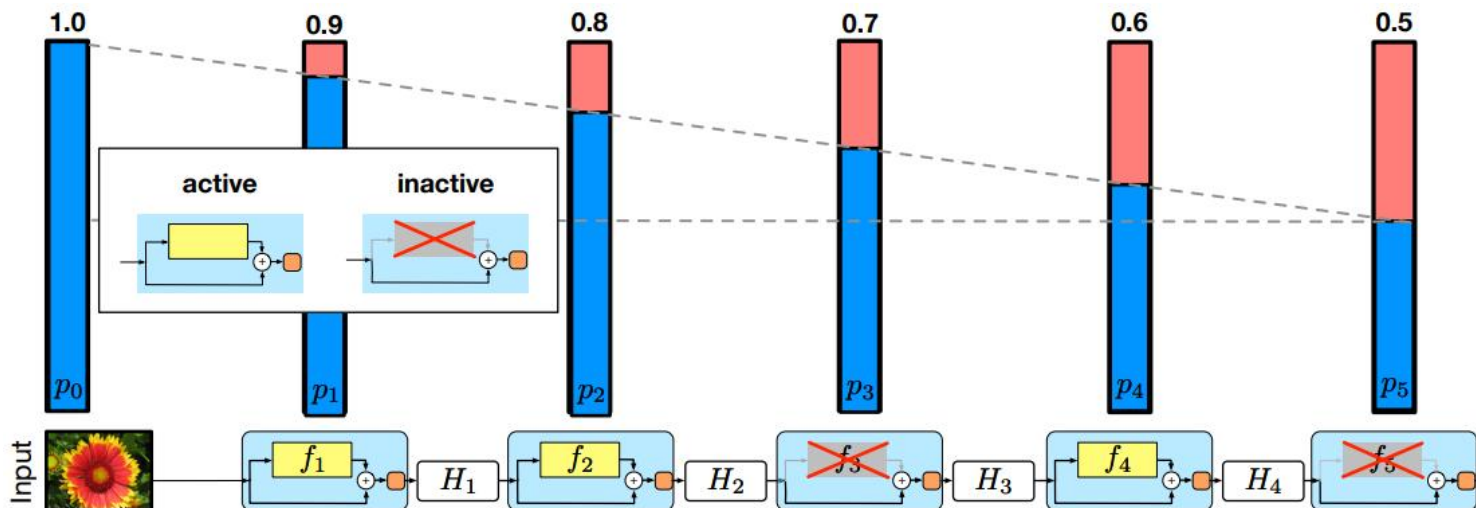
where k is the total number of classes



When Does Label Smoothing Help?

Stochastic Depth

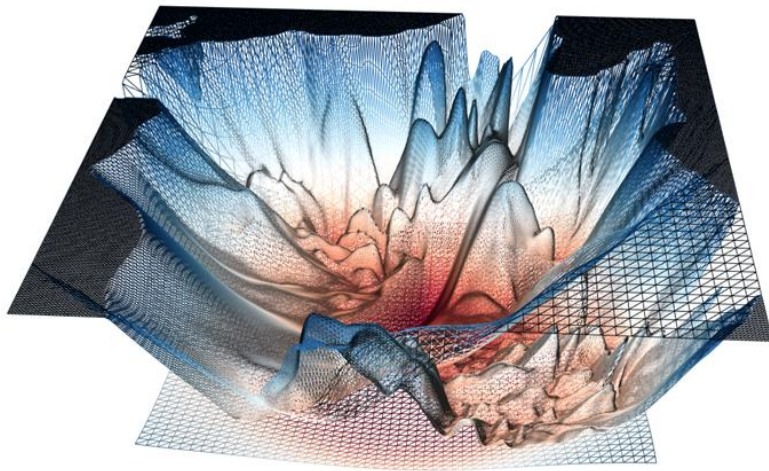
- When training the ResNet network, a random variable b (Bernoulli random variable, which can only take 0/1, the probability of taking 0 is $1-p$, and the probability of taking 1 is p) is added, and the entire ResBlock convolution part is randomly discarded.
 - If $b = 1$, it is simplified to the original ResNet structure;
 - If $b = 0$, this ResBlock is not activated, reducing to the identity function.
- Using different depth training, it is equivalent to potentially merging multiple networks of different depths to improve performance.



Summary

Model Learning Principles:

- Stochastic gradient descent
- Various experience strategies



<https://www.telesens.co/2019/01/16/neural-network-loss-visualization/>

Summary of empirical strategies:

- The loss function is highly irregular and non-convex
 - Weight initialization: pre-trained model
 - Optimizer Improvements: Momentum SGD, Adaptive Gradient Algorithm
 - Learning rate strategy: learning rate annealing, learning rate warmup
- The model is complex and prone to overfitting
 - Early Stopping,
 - Data Augmentation, Label Smoothing
 - Dropout, Random Depth
- The model converges slowly
 - Batch Normalization



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Thanks for Listening !

Ruimao Zhang

Room 517, Daoyuan Building, The Chinese Univeristy of Hong Kong, Shenzhen

zhangruimao@cuhk.edu.cn

ruimao.zhang@ieee.org
