

# AI Overview



# Foreword

---

- Mankind is welcoming the fourth industrial revolution represented by intelligent technology. New technologies such as AI, IoT, 5G and bioengineering are integrated into all aspects of human society; driving changes in global macro trends, such as sustainable social development and economic growth. New kinetic energy, smart city upgrading, industrial digital transformation, consumer experience, etc.
- As the world's leading provider of ICT (information and communications) infrastructure and smart terminals, Huawei actively participates in the transformation of artificial intelligence and proposes Huawei's full-stack full-scenario AI strategy. This chapter will mainly introduce AI Overview, Technical Fields and Application Fields of AI, Huawei's AI Development Strategy, AI Disputes, Future Prospects of AI.

# Objectives

Upon completion of this course, you will be able to:

- Understand basic concepts of AI.
- Understand AI technologies and their development history.
- Understand the application technologies and application fields of AI.
- Know Huawei's AI development strategy.
- Know the development trends of AI.

# Contents

---

- 1. AI Overview**
2. Technical Fields and Application Fields of AI
3. Huawei's AI Development Strategy
4. AI Disputes
5. Future Prospects of AI

# AI in the Eyes of the Society

- People get to know AI through news, movies, and actual applications in daily life. What is AI in the eyes **of the public?**

Haidian Park: First AI-themed Park in the World  
StarCraft II: AlphaStar Beat Professional Players  
AI-created Edmond de Belamy Sold at US\$430,000  
Demand for AI Programmers:↑ 35 Times! Salary:  
Top 1!  
50% Jobs Will be Replaced by AI in the future  
Winter is Coming? AI Faces Challenges  
...

The Terminator  
2001: A Space Odyssey  
The Matrix  
I, Robot  
Blade Runner  
Elle  
Bicentennial Man  
...

Self-service security check  
Spoken language evaluation  
Music/Movie recommendation  
Smart speaker  
AI facial fortune-telling  
Vacuum cleaning robot  
Self-service bank terminal  
Intelligent customer service  
Siri  
...

## News

AI Applications  
AI industry outlook  
Challenges faced by AI  
...

## Movies

AI Control over human beings  
Fall in love with AI  
Self-awareness of AI  
...

## Applications in daily life

Security protection  
Entertainment  
Smart Home  
Finance  
...

# AI in the Eyes of Researchers

"I propose to consider the question, 'Can machines think?'"

— Alan Turing 1950

The branch of computer science concerned with making computers behave like humans.

— John McCarthy 1956

The science of making machines do things that would require intelligence if done by men.

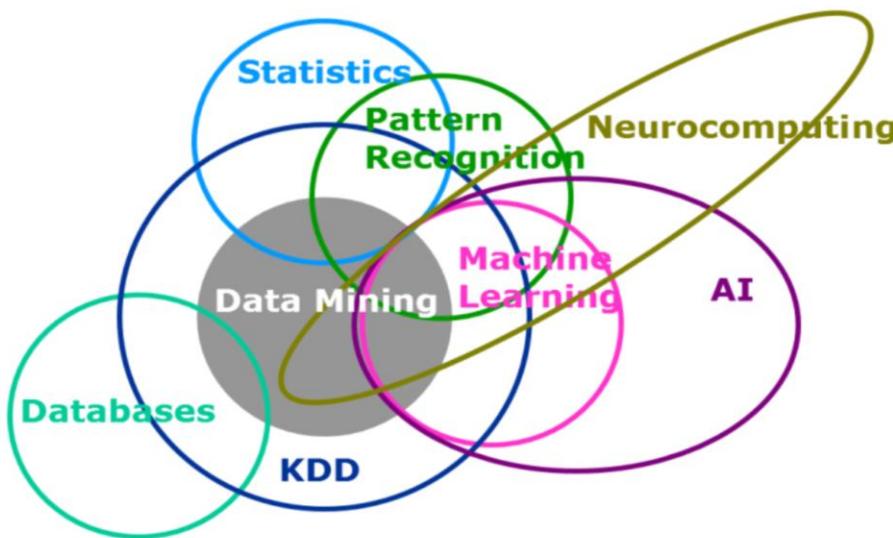
— Marvin Minsky

# What Are Intelligences?

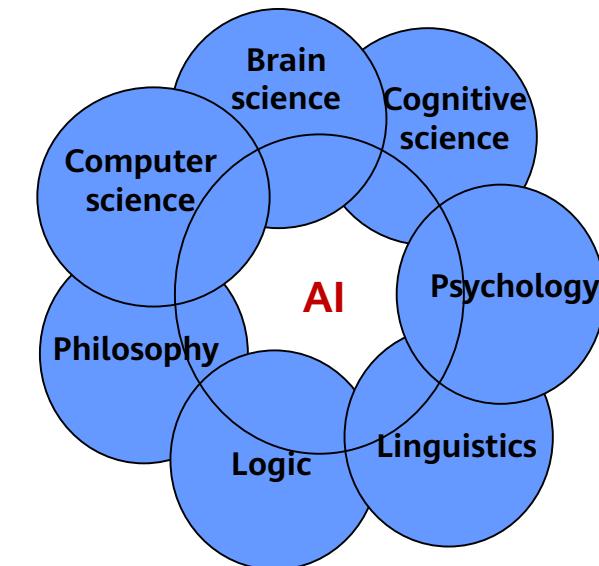
- Howard Gardner's Multiple Intelligences
- Human intelligences can be divided into seven categories:
  - Verbal/Linguistic
  - Logical/Mathematical
  - Visual/Spatial
  - Bodily/Kinesthetic
  - Musical/Rhythmic
  - Inter-personal/Social
  - Intra-personal/Introspective

# What Is AI?

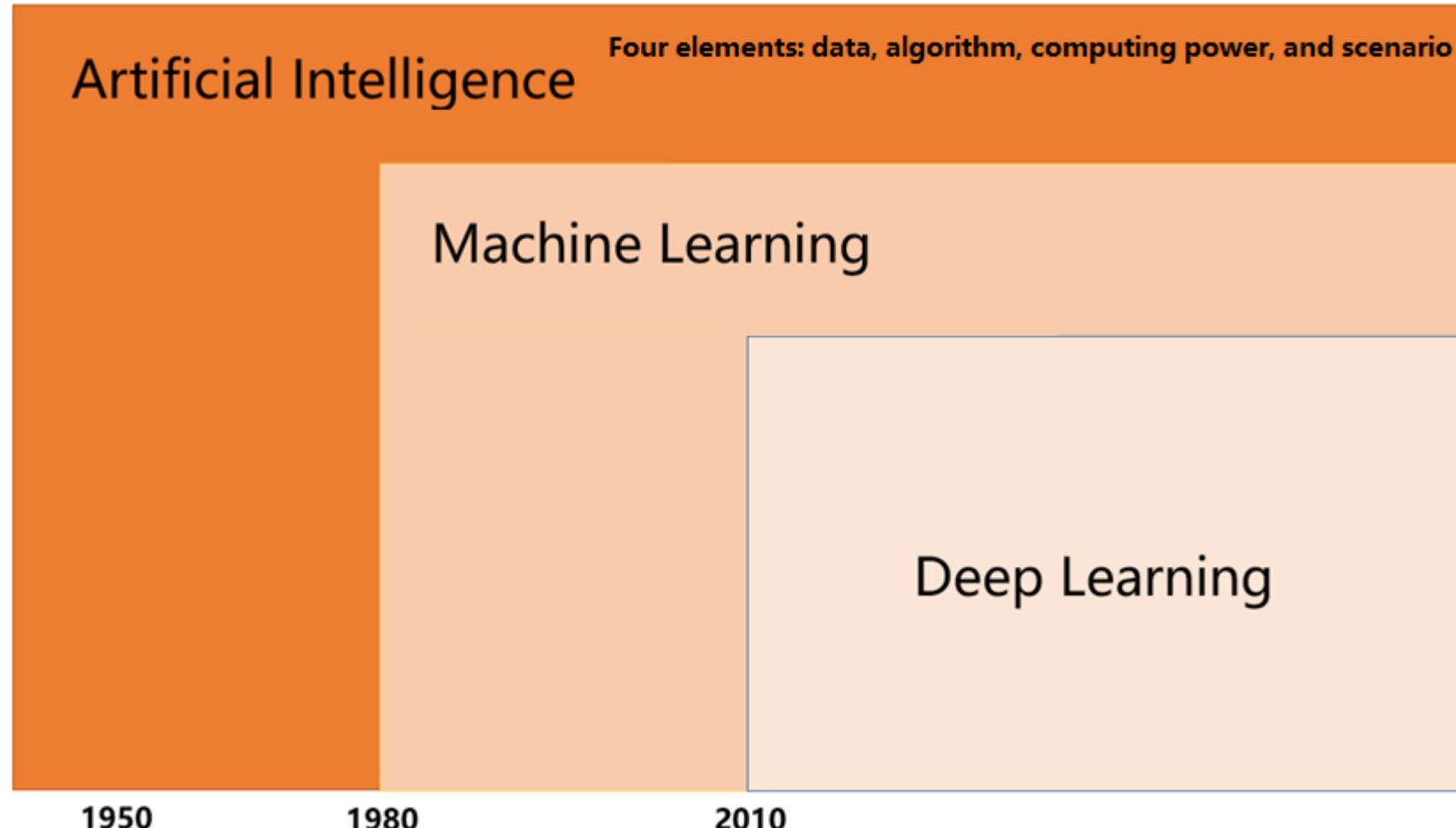
- Artificial Intelligence (AI) is a new technical science that studies and develops theories, methods, techniques, and application systems for simulating and extending human intelligence. In 1956, the concept of AI was first proposed by John McCarthy, who defined the subject as "science and engineering of making intelligent machines, especially intelligent computer program". AI is concerned with making machines work in an intelligent way, similar to the way that the human mind works. At present, AI has become an interdisciplinary course that involves various fields.



Identification of concepts related to AI and machine learning  
AI Development Report 2020



# Relationship of AI, Machine Learning, and Deep Learning



# Relationship of AI, Machine Learning and Deep Learning

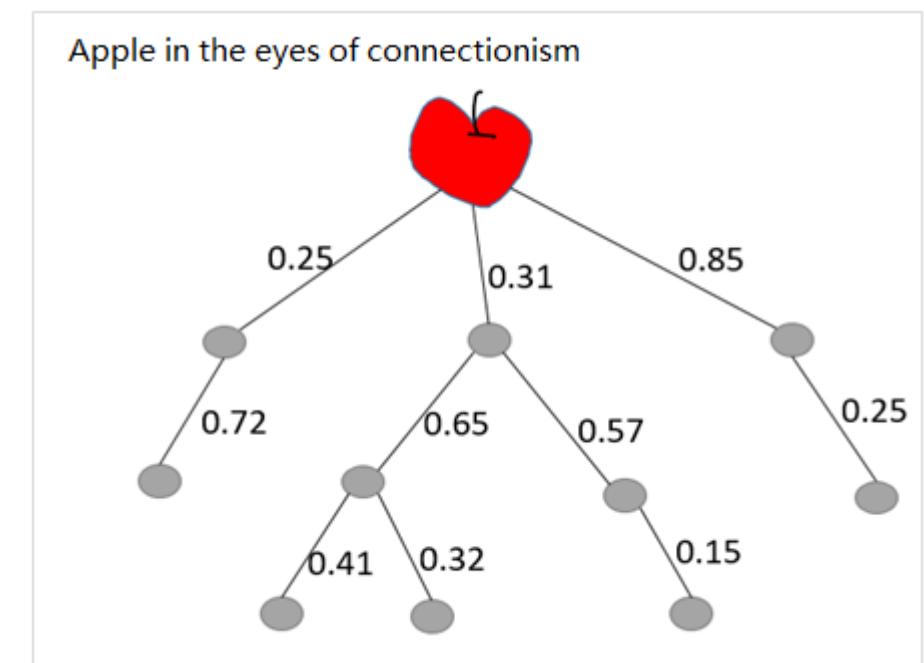
- AI: A new technical science that focuses on the research and development of theories, methods, techniques, and application systems for simulating and extending human intelligence.
- Machine learning: A core research field of AI. It focuses on the study of how computers can obtain new knowledge or skills by simulating or performing learning behavior of human beings, and reorganize existing knowledge architecture to improve its performance. It is one of the core research fields of AI.
- Deep learning: A new field of machine learning. The concept of deep learning originates from the research on artificial neural networks. The multi-layer perceptron (MLP) is a type a deep learning architecture. Deep learning aims to simulate the human brain to interpret data such as images, sounds, and texts.

# Three Major Schools of Thought: Symbolism

- Basic thoughts
  - The cognitive process of human beings is the process of inference and operation of various symbols.
  - A human being is a physical symbol system, and so is a computer. Computers, therefore, can be used to simulate intelligent behavior of human beings.
  - The core of AI lies in knowledge representation, knowledge inference, and knowledge application. Knowledge and concepts can be represented with symbols. Cognition is the process of symbol processing while inference refers to the process of solving problems by using heuristic knowledge and search.
- Representative of symbolism: inference, including symbolic inference and machine inference

# Three Major Schools of Thought: Connectionism

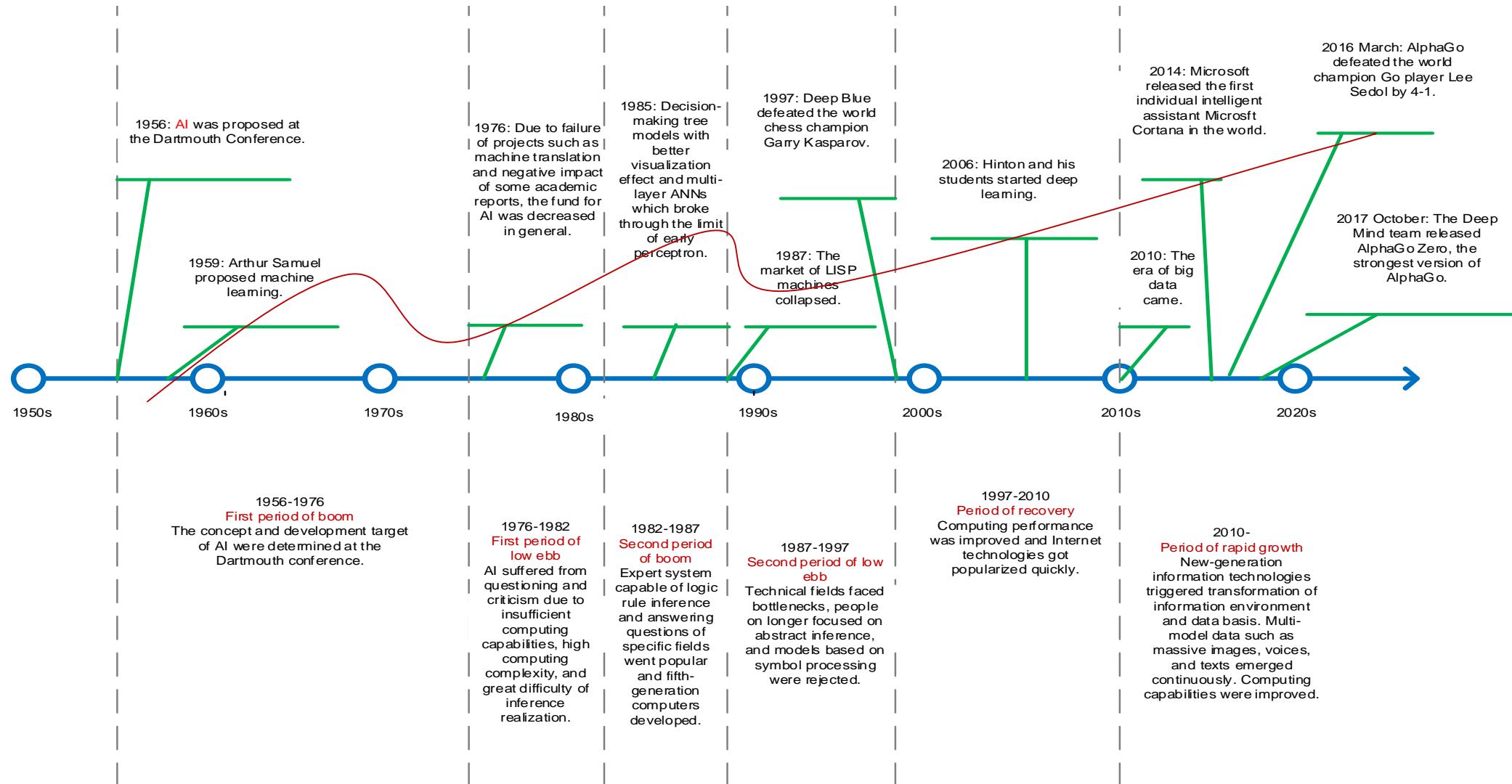
- Basic thoughts
  - The basis of thinking is neurons rather than the process of symbol processing.
  - Human brains vary from computers. A computer working mode based on connectionism is proposed to replace the computer working mode based on symbolic operation.
- Representative of connectionism: neural networks and deep learning



# Three Major Schools of Thought: Behaviorism

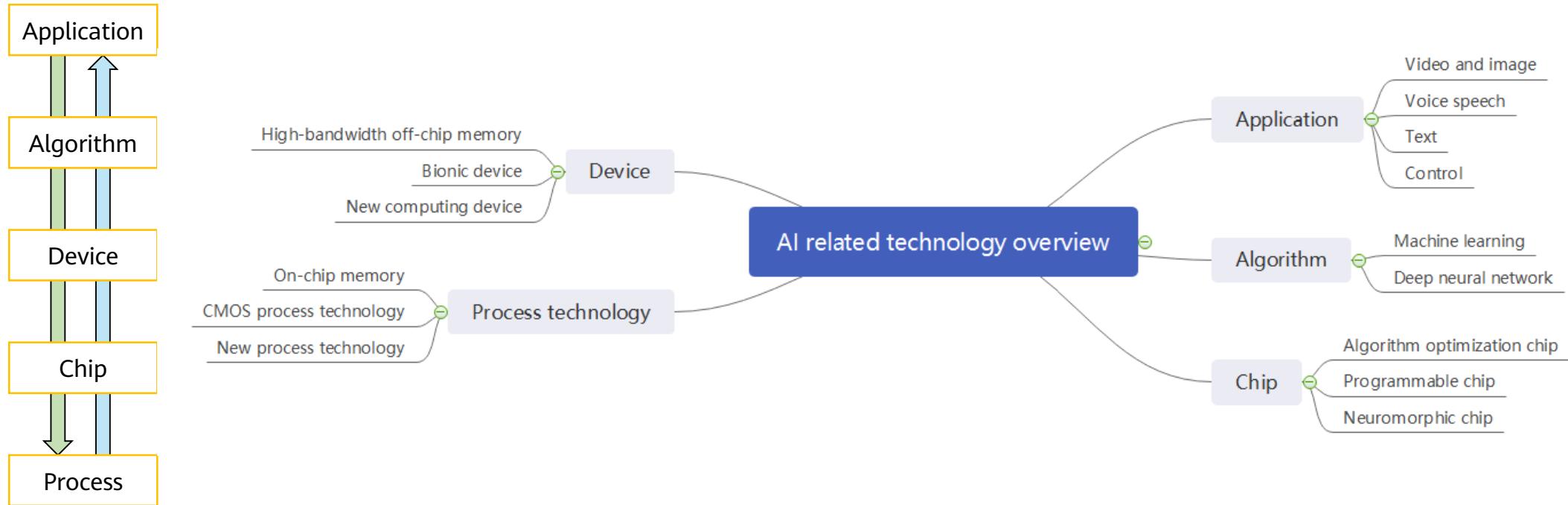
- Basic thoughts:
  - Intelligence depends on perception and action. The perception-action mode of intelligent behavior is proposed.
  - Intelligence requires no knowledge, representation, or inference. AI can evolve like human intelligence. Intelligent behavior can only be demonstrated in the real world through the constant interaction with the surrounding environment.
- Representative of behaviorism: behavior control, adaptation, and evolutionary computing

# Brief Development History of AI



# Overview of AI Technologies

- AI technologies are multi-layered, covering the application, algorithm mechanism, toolchain, device, chip, process, and material layers.



# Types of AI

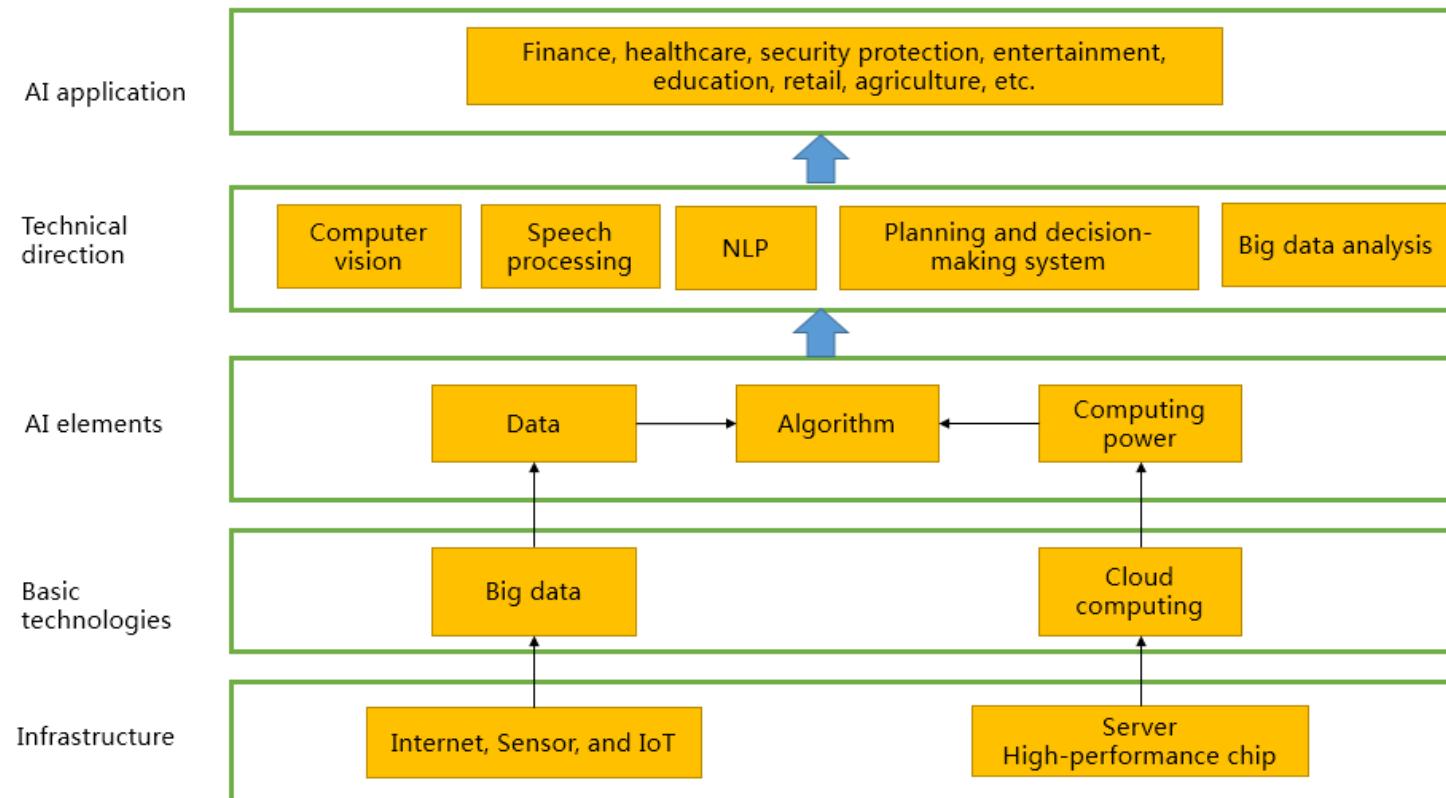
- Strong AI
  - The strong AI view holds that it is possible to create intelligent machines that can really reason and solve problems. Such machines are considered to be conscious and self-aware, can independently think about problems and work out optimal solutions to problems, have their own system of values and world views, and have all the same instincts as living things, such as survival and security needs. It can be regarded as a new civilization in a certain sense.
- Weak AI
  - The weak AI view holds that intelligent machines cannot really reason and solve problems. These machines only look intelligent, but do not have real intelligence or self-awareness.

# Classification of Intelligent Robots

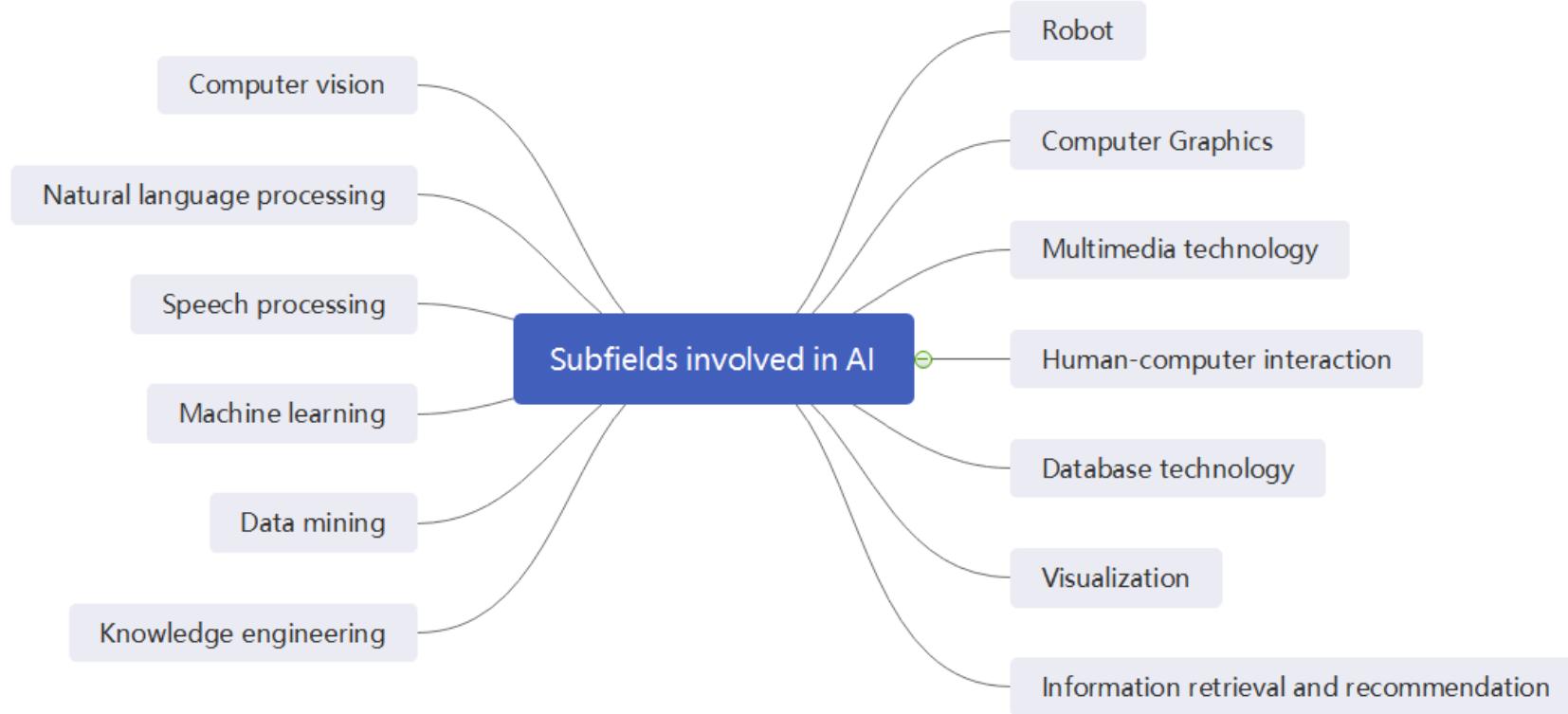
- Currently, there is no unified definition of AI research. Intelligent robots are generally classified into the following four types:
  - "Thinking like human beings": weak AI, such as Watson and AlphaGo
  - "Acting like human beings": weak AI, such as humanoid robot, iRobot, and Atlas of Boston Dynamics
  - "Thinking rationally": strong AI (Currently, no intelligent robots of this type have been created due to the bottleneck in brain science.)
  - "Acting rationally": strong AI

# AI Industry Ecosystem

- The four elements of AI are data, algorithm, computing power, and scenario. To meet requirements of these four elements, we need to combine AI with cloud computing, big data, and IoT to build an intelligent society.



# Sub-fields of AI



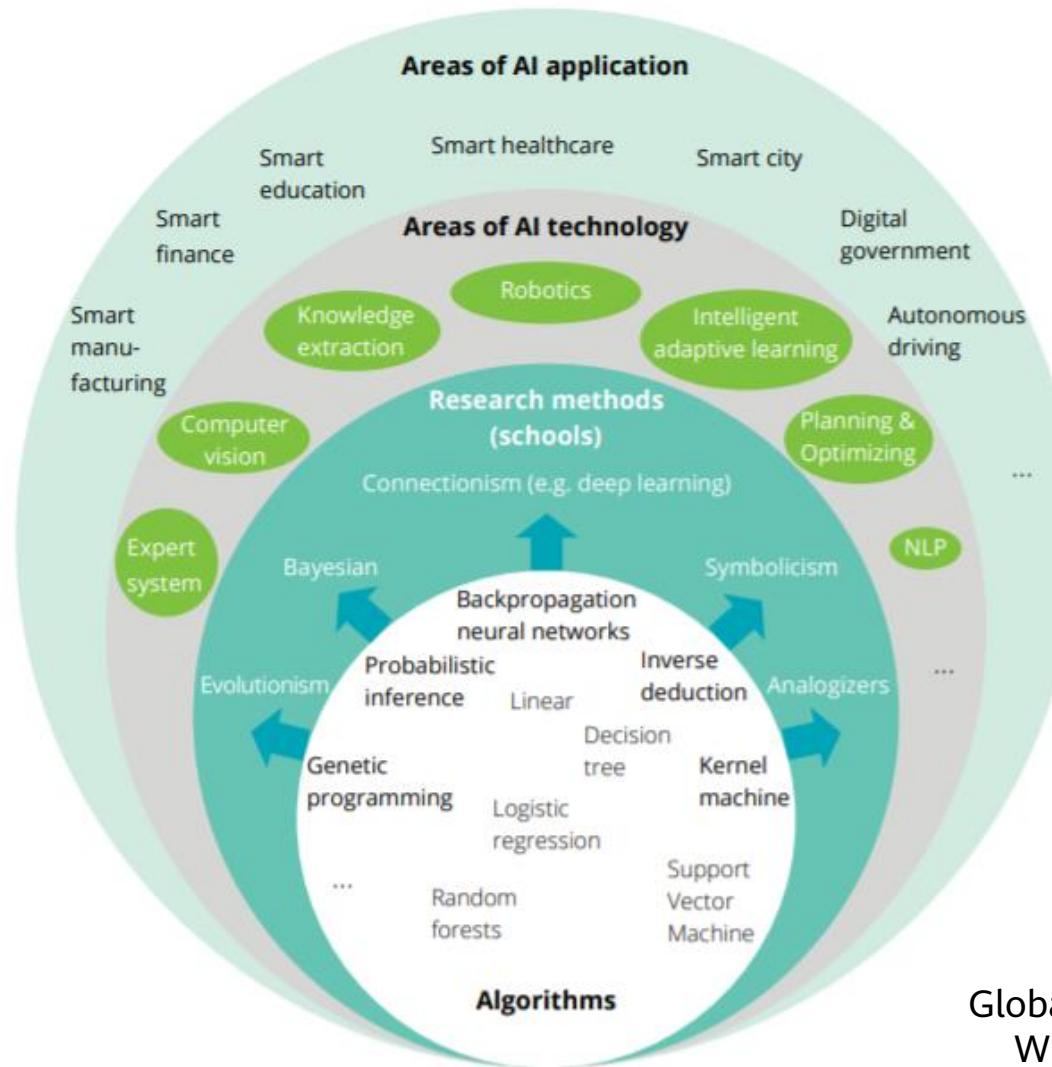
AI Development Report 2020

# Contents

---

1. AI Overview
- 2. Technical Fields and Application Fields of AI**
3. Huawei's AI Development Strategy
4. AI Disputes
5. Future Prospects of AI

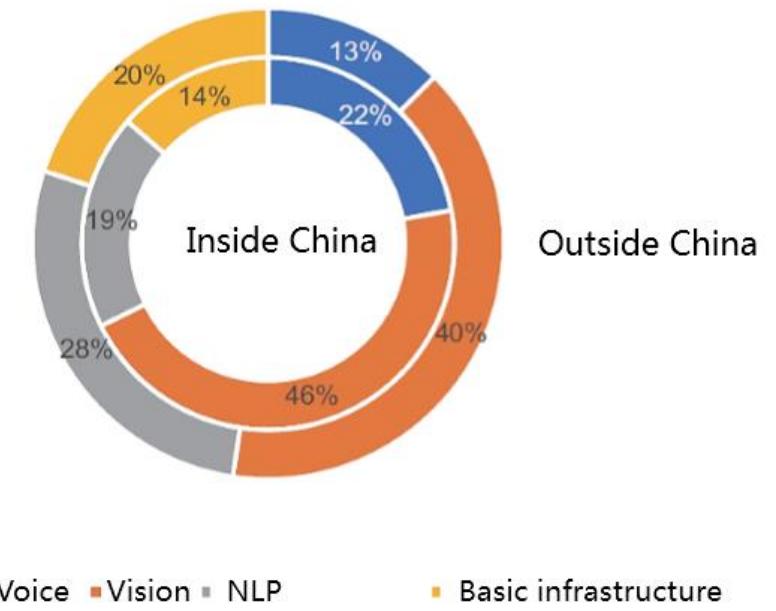
# Technical Fields and Application Fields of AI



Global AI Development  
White Paper 2020

# Distribution of AI Application Technologies in Enterprises Inside and Outside China

- At present, application directions of AI technologies mainly include:
  - **Computer vision:** a science of how to make computers "see"
  - **Speech processing:** a general term for various processing technologies used to research the voicing process, statistical features of speech signals, speech recognition, machine-based speech synthesis, and speech perception
  - **Natural language processing (NLP):** a subject that uses computer technologies to understand and use natural language

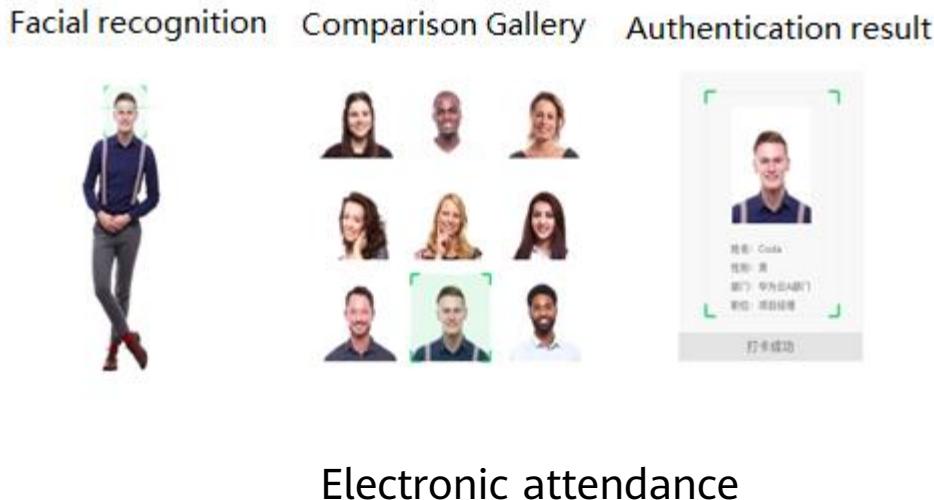


Distribution of AI application technologies in enterprises inside and outside China

China AI Development Report 2018

# Computer Vision Application Scenario (1)

- Computer vision is the most mature technology among the three AI technologies. The main topics of computer vision research include image classification, target detection, image segmentation, target tracking, optical character recognition (OCR), and facial recognition.
- In the future, computer vision is expected to enter the advanced stage of autonomous understanding, analysis, and decision-making, enabling machines to "see" and bringing greater value to scenarios such as unmanned vehicles and smart homes.
- Application scenarios:



# Computer Vision Application Scenario (2)

Action analysis



Authentication

Facial verification passed



Facial verification failed



Smart album

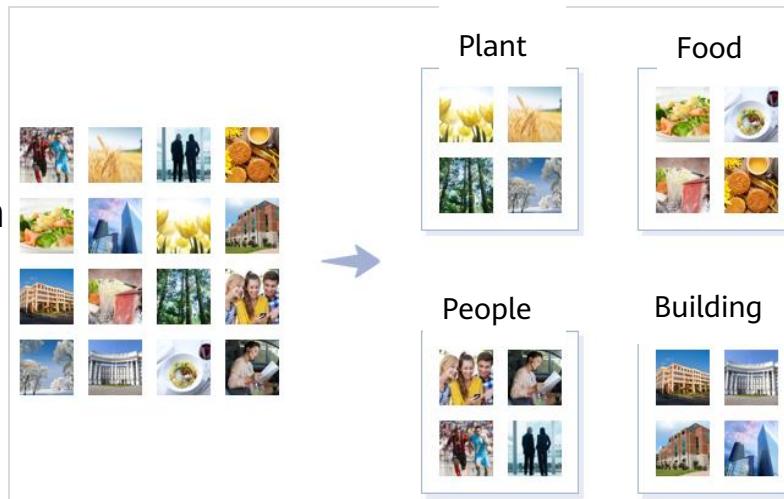
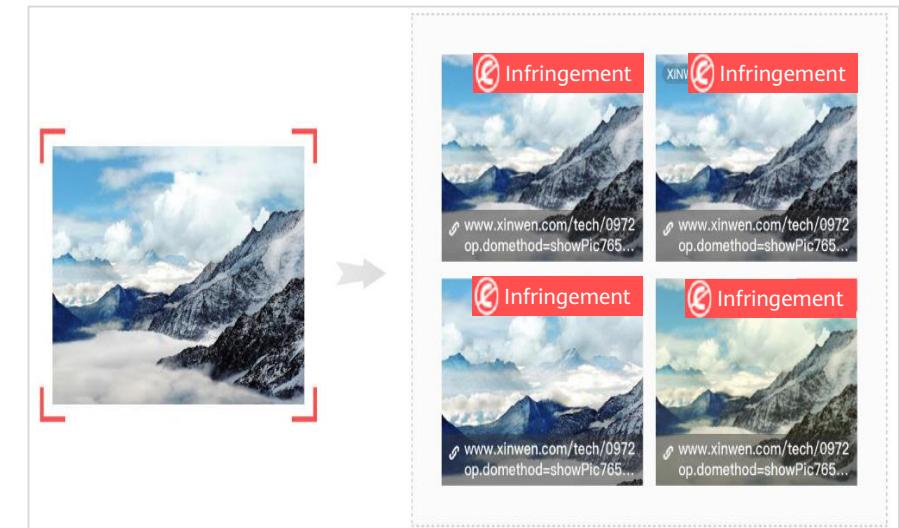


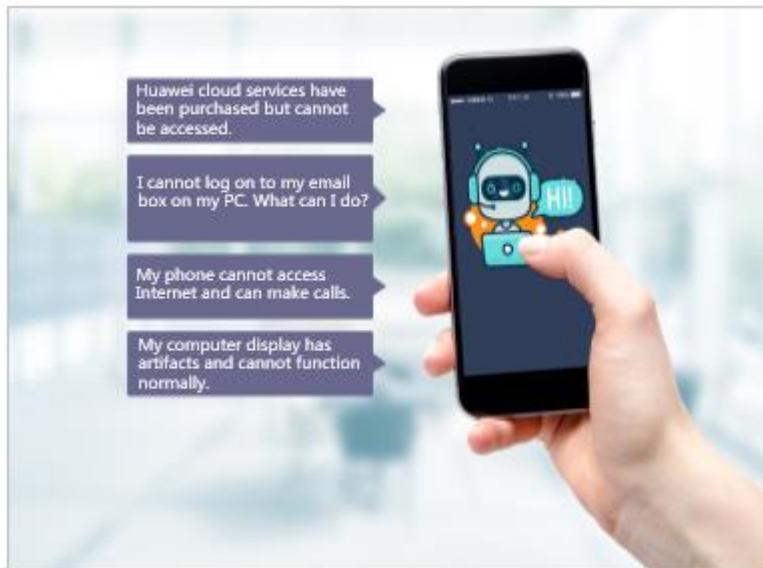
Image search



# Voice Processing Application Scenario (1)

- The main topics of voice processing research include voice recognition, voice synthesis, voice wakeup, voiceprint recognition, and audio-based incident detection. Among them, the most mature technology is voice recognition. As for near field recognition in a quite indoor environment, the recognition accuracy can reach 96%.
- Application scenarios:

Question Answering Bot (QABot)



Voice navigation



# Voice Processing Application Scenario (2)

Intelligent education



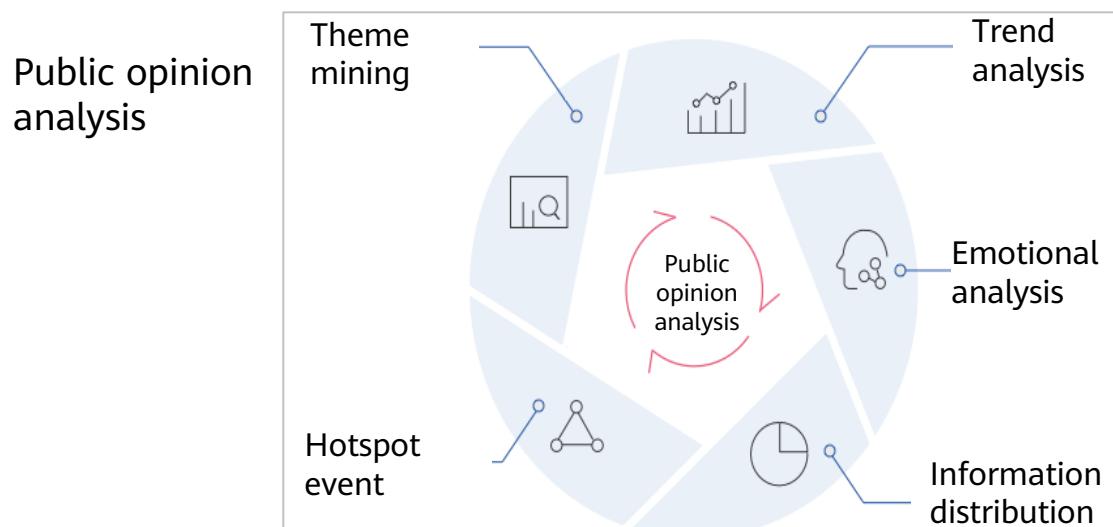
Real-time conference records



- Other applications:
  - Spoken language evaluation
  - Diagnostic robot
  - Voiceprint recognition
  - Smart sound box
  - ...

# NLP Application Scenario (1)

- The main topics of NLP research include machine translation, text mining, and sentiment analysis. NLP imposes high requirements on technologies but confronts low technology maturity. Due to high complexity of semantics, it is hard to reach the human understanding level using parallel computing based on big data and parallel computing only.
- In future, NLP will achieve more growth: understanding of shallow semantics → automatic extraction of features and understanding of deep semantics; single-purpose intelligence (ML) → hybrid intelligence (ML, DL, and RL)
- Application scenarios:

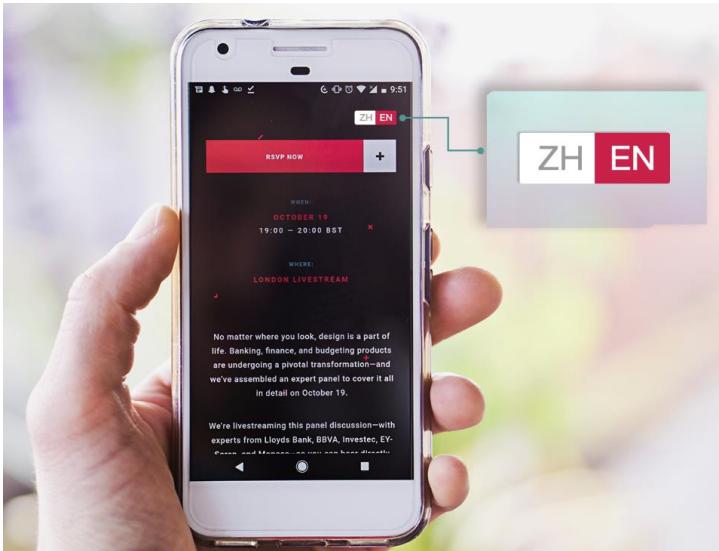


Evaluation analysis



# NLP Application Scenario (2)

Machine translation



Text classification



- Other applications:
  - Knowledge graph
  - Intelligent copywriting
  - Video subtitle
  - ...

# AI Application Field - Intelligent Healthcare

**Medicine mining:** quick development of personalized medicines by AI assistants

**Health management:** nutrition, and physical/mental health management

**Hospital management:** structured services concerning medical records (focus)

**Assistance for medical research:** assistance for biomedical researchers in research

**Virtual assistant:** electronic voice medical records, intelligent guidance, intelligent diagnosis, and medicine recommendation

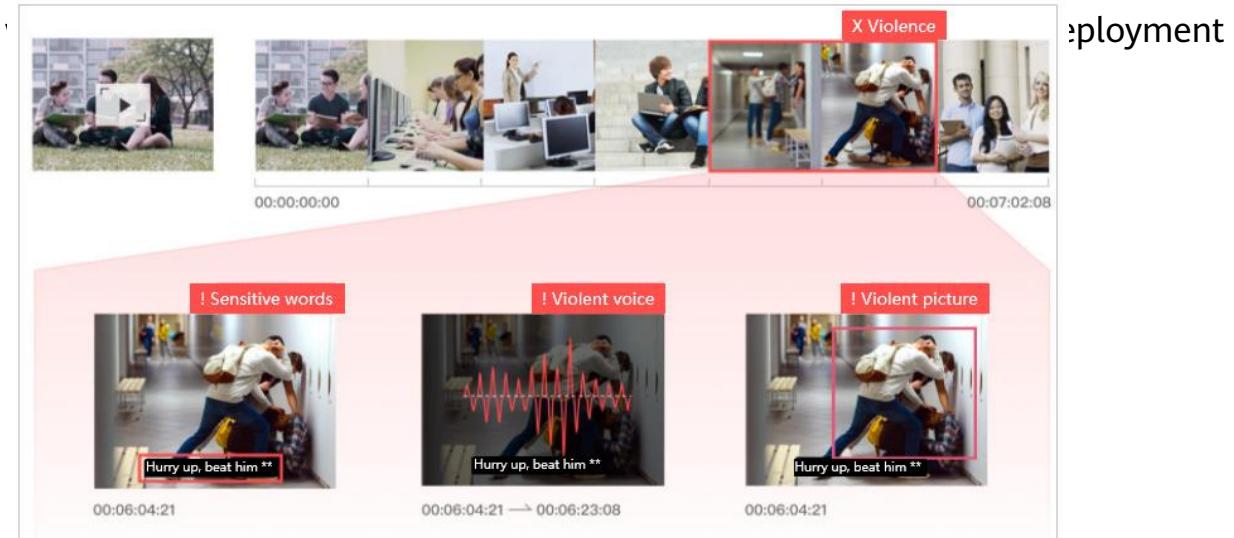
**Medical image:** medical image recognition, image marking, and 3D image reconstruction

**Assistance for diagnosis and treatment:** diagnostic robot

**Disease risk forecast:** disease risk forecast based on gene sequencing

# AI Application Field - Intelligent Security

- Security protection is considered the easiest field for AI implementation. AI technologies applied in this field are relatively mature. The field involves massive data of images and videos, laying a sound foundation for training of AI algorithms and models. Currently, AI technologies are applied to two directions in the security protection field, namely, civil use and police use.
- Application scenarios:
  - **Police use:** suspect identification, vehicle analysis, suspect tracking, suspect search and comparison, and access control at key places
  - **Civil use:** facial recognition, ...



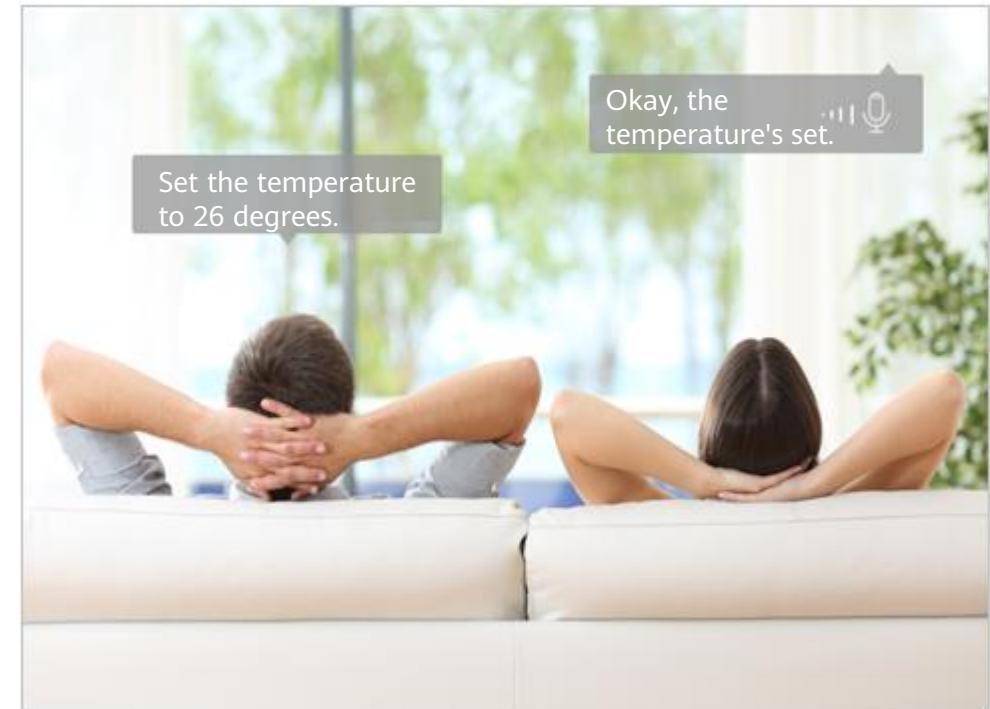
# AI Application Field - Smart Home

- Based on IoT technologies, a smart home ecosystem is formed with hardware, software, and cloud platforms, providing users personalized life services and making home life more convenient, comfortable, and safe.

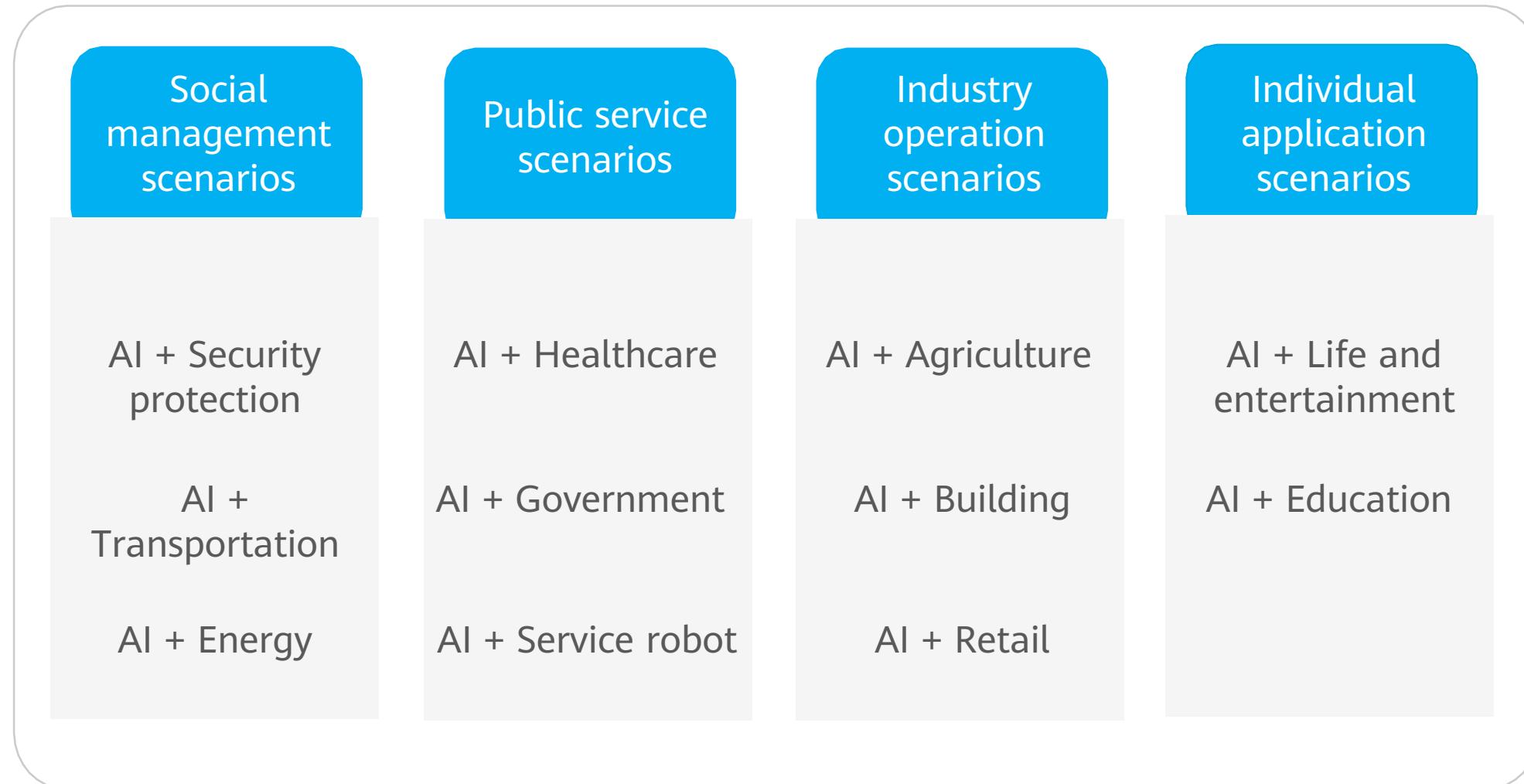
**Control smart home products with voice processing** such as air conditioning temperature adjustment, curtain switch control, and voice control on the lighting system.

Implement **home security protection** with computer vision technologies, for example, facial or fingerprint recognition for unlocking, real-time intelligent camera monitoring, and illegal intrusion detection.

**Develop user profiles and recommend content to users** with the help of machine learning and deep learning technologies and based on historical records of smart speakers and smart TVs.



# AI Application Field - Smart City



# AI Application Field - Retail

- AI will bring revolutionary changes to the retail industry. A typical symptom is unmanned supermarkets. For example, Amazon Go, unmanned supermarket of Amazon, uses sensors, cameras, computer vision, and deep learning algorithms to completely cancel the checkout process, allowing customers to pick up goods and "just walk out".
- One of the biggest challenges for unmanned supermarket is how to charge the right fees to the right customers. So far, Amazon Go is the only successful business case and even this case involves many controlled factors. For example, only Prime members can enter Amazon Go. Other enterprises, to follow the example of Amazon, have to build their membership system first.



# AI Application Field - Autonomous Driving

- The Society of Automotive Engineers (SAE) in the U.S. defines 6 levels of driving automation ranging from 0 (fully manual) to 5 (fully autonomous). L0 indicates that the driving of a vehicle completely depends on the driver's operation. The system above L3 can implement the driver's hand-off operation in specific cases, L5 depends on the system when vehicles are driving in all scenarios.
- Currently, only some commercial passenger vehicle models, such as Audi A8, Tesla, and Cadillac, support L2 and L3 Advanced driver-assistance systems (ADAS). It is estimated that by 2020, more L3 vehicle models will emerge with the further improvement of sensors and vehicle-mounted processors. L4 and L5 autonomous driving is expected to be first implemented on **commercial vehicles in closed campuses**. A wider range of passenger vehicles require advanced autonomous driving, which requires further improvement of technologies, policies, and infrastructure. It is estimated that L4 and L5 autonomous driving will be supported by common roads in 2025–2030.

# AI Will Change All Industries



## Public sector

- Safe City
- Intelligent transport
- Disaster prediction



## Education

- Personalization
- Attention improvement
- Robot teacher



## Healthcare

- Early prevention
- Diagnosis assistance
- Precision cure



## Media

- Real-time translation
- Abstraction
- Inspection



## Pharmacy

- Fast R&D
- Precise trial
- Targeted medicine



## Logistics

- Routing planning
- Monitoring
- Auto sorting



## Finance

- Doc process
- Real-time fraud prevention
- Up-sell



## Insurance

- Auto detection
- Fraud prevention
- Innovative service



## Retail

- Staff-less shops
- Real-time inventory
- Precise recommendations



## Manufacturing

- Defect detection
- Industrial internet
- Predictive maintenance



## Telecom

- Customer service
- Auto O&M
- Auto optimization



## Oil and gas

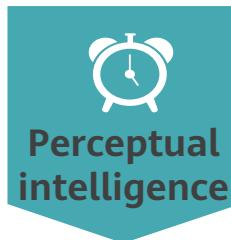
- Localization
- Remote maintenance
- Operation optimization



## Agriculture

- Fertilization improvement
- Remote operation
- Seeds development

# AI: Still in Its Infancy

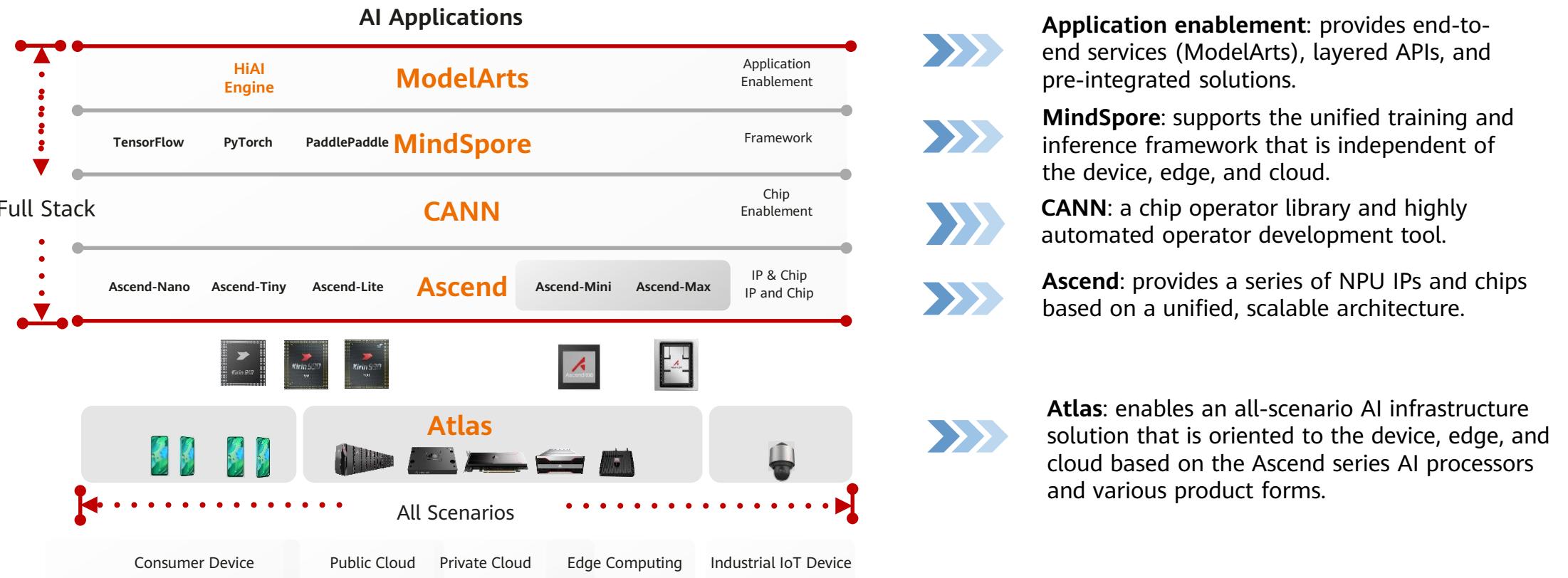
Three Phases of AI	Ability	Example Value	Benefits
 <b>Computing intelligence</b>	<b>Capable of storage and computing:</b> Machines can compute and transfer information as human beings do.	Distributed computing and neural network	Help human beings store and quickly process massive data, laying a foundation for perception and cognition.
 <b>Perceptual intelligence</b>	<b>Capable of listening and seeing:</b> Machines can listen and see, make judgments, and take simple actions.	Cameras capable of facial recognition and speakers able to understand speeches	Help human beings efficiently finish work related to listening and seeing.
 <b>Cognitive intelligence</b>	<b>Capable of understanding and thinking:</b> Machines can understand, think, and make decisions like human beings.	Unmanned vehicles enabling autonomous driving and robots acting autonomously	Fully assist in or replace partial work of human beings.

# Contents

---

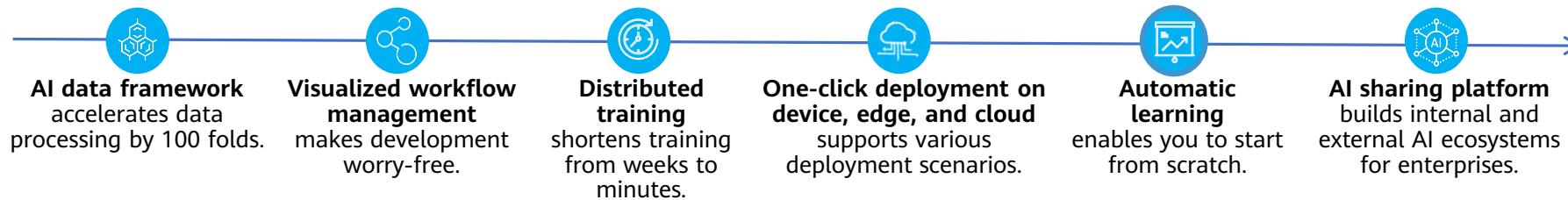
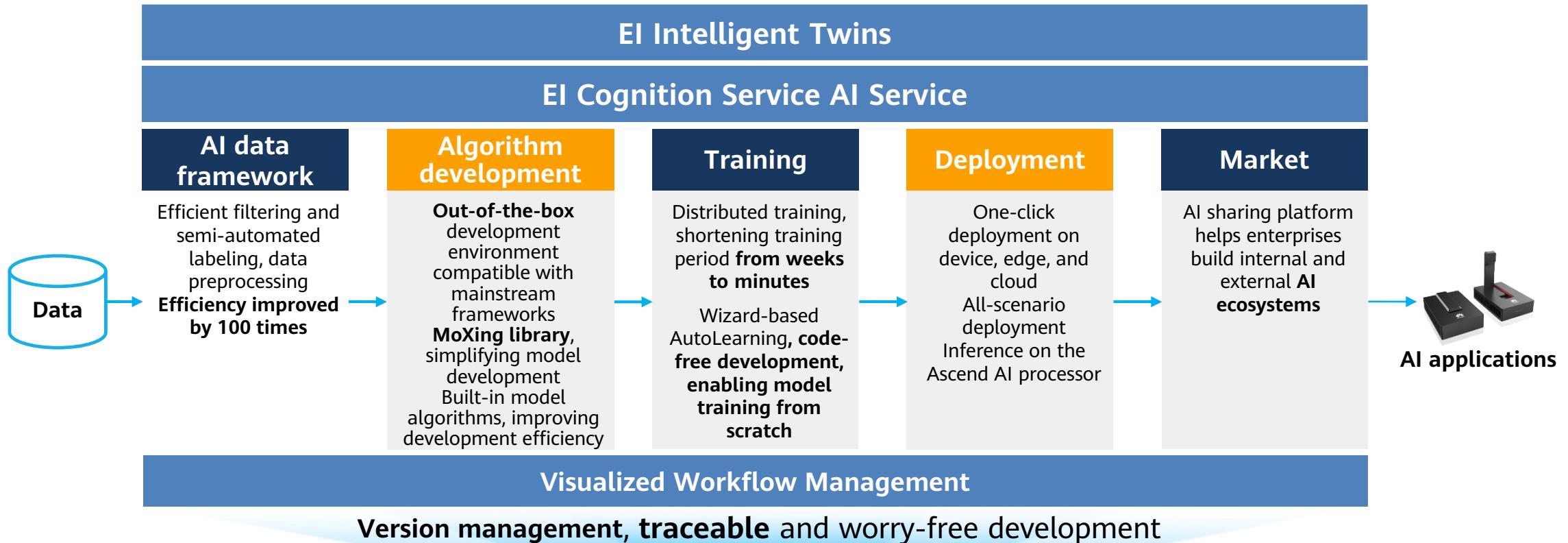
1. AI Overview
2. Technical Fields and Application Fields of AI
- 3. Huawei's AI Development Strategy**
4. AI Disputes
5. Future Prospects of AI

# Huawei's Full-Stack, All-Scenario AI Portfolio



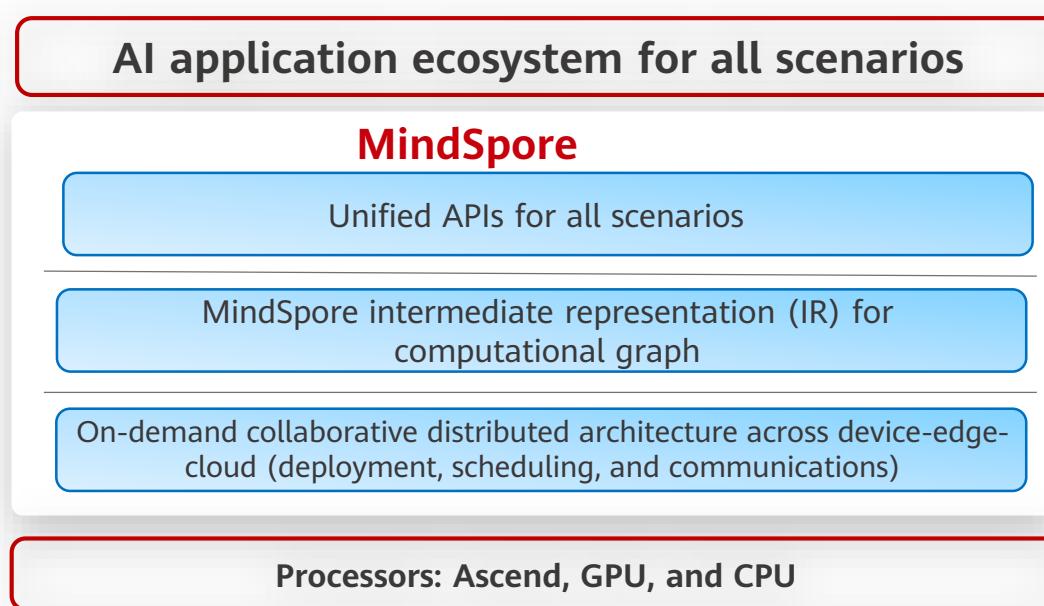
Huawei's "all AI scenarios" indicate different deployment scenarios for AI, including public clouds, private clouds, edge computing in all forms, industrial IoT devices, and consumer devices.

# Full Stack - ModelArts Full-Cycle AI Workflow

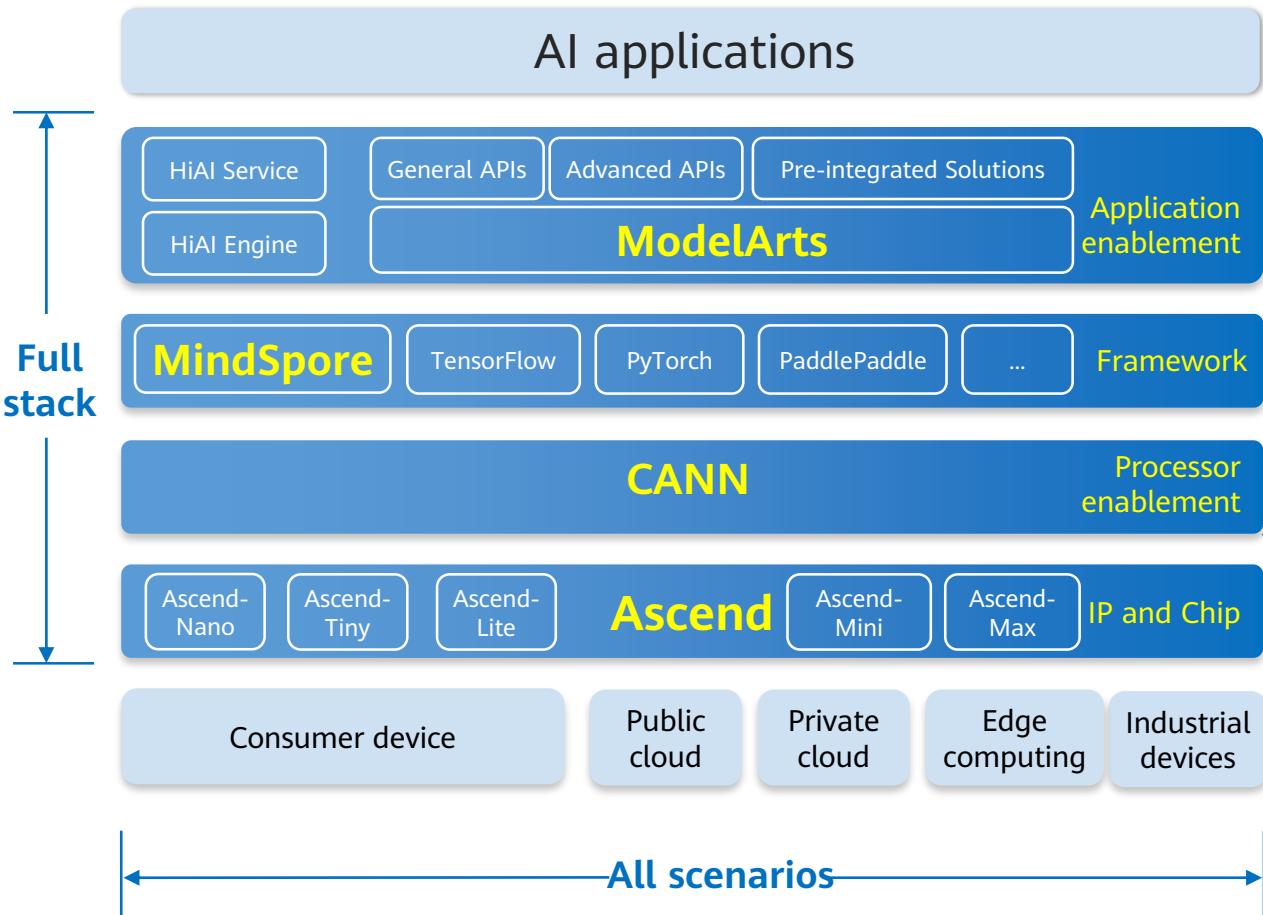


# Full Stack — MindSpore (Huawei AI Computing Framework)

- MindSpore provides automatic parallel capabilities. With MindSpore, senior algorithm engineers and data scientists who focus on data modeling and problem solving can run algorithms on dozens or even thousands of AI computing nodes with only a few lines of description.
- The MindSpore framework supports both large-scale and small-scale deployment, adapting to independent deployment in all scenarios. In addition to the Ascend AI processors, MindSpore also supports other processors such as GPUs and CPUs.

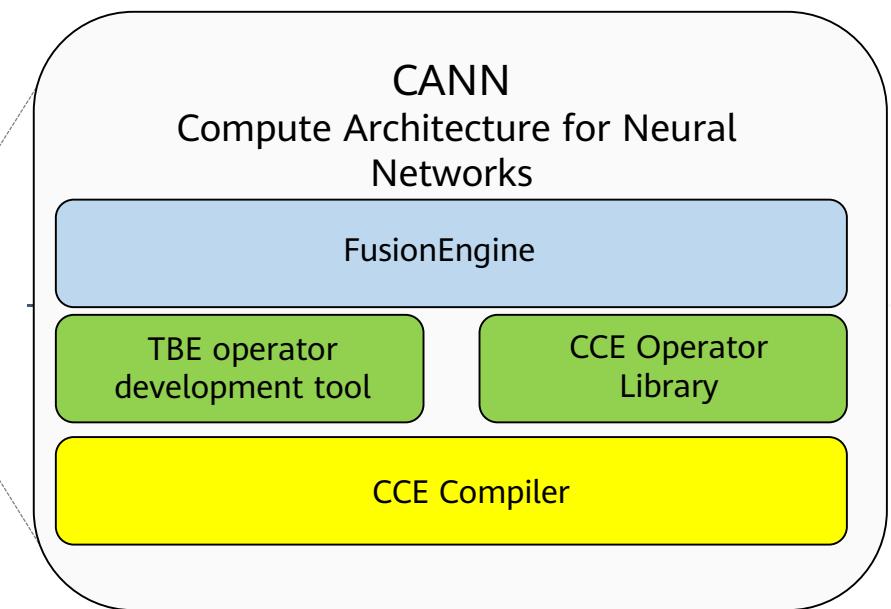


# Full Stack — CANN



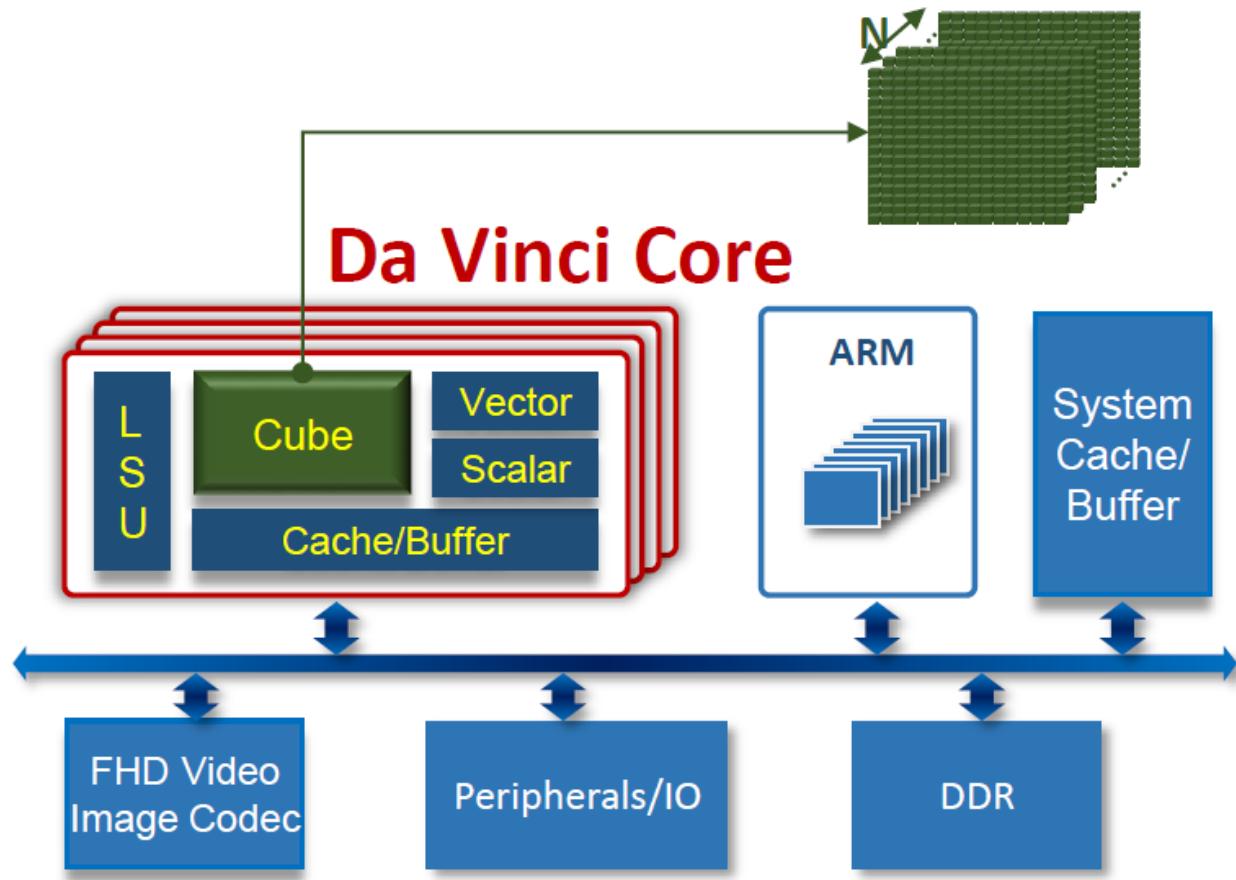
## CANN:

**A chip operators library and highly automated operator development toolkit**  
**Optimal development efficiency**, in-depth optimization of the common operator library, and abundant APIs  
**Operator convergence, best matching the performance of the Ascend chip**



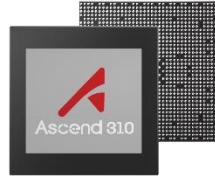
# Full Stack — Ascend 310 AI Processor and Da Vinci Core

SPECIFICATIONS	Description
Architecture	AI co-processor
Performance	Up to 8T @FP16
	Up to 16T@INT8
Codec	16 Channel Decoder – H.264/265 1080P30 1 Channel Encoder
Memory Controller	LPDDR4X
Memory Bandwidth	2*64bit @3733MT/S
System Interface	PCIe3.0 /USB 3.0/GE
Package	15mm*15mm
Max Power	8Tops@4W, 16Tops@8W
Process	12nm FFC



**Note:** This is typical configuration, high performance and low power sku can be offered based on your requirement.

# Ascend AI Processors: Infusing Superior Intelligence for Computing



## Ascend 310

AI SoC with ultimate energy efficiency

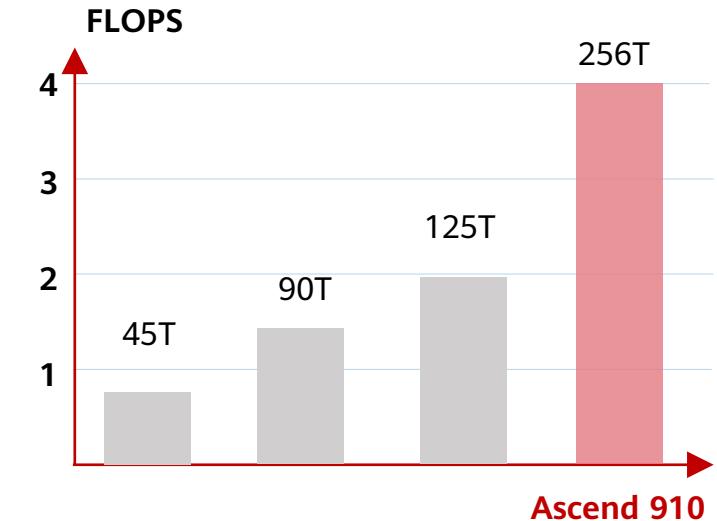
- Ascend-Mini  
Architecture: Da Vinci
- Half-precision (FP16): 8 TFLOPS**
- Integer precision (INT8): 16 TOPS**
- 16-channel full-HD video decoder: H.264/265**
- 1-channel full-HD video encoder: H.264/265**
- Max. power: 8 W**



## Ascend 910

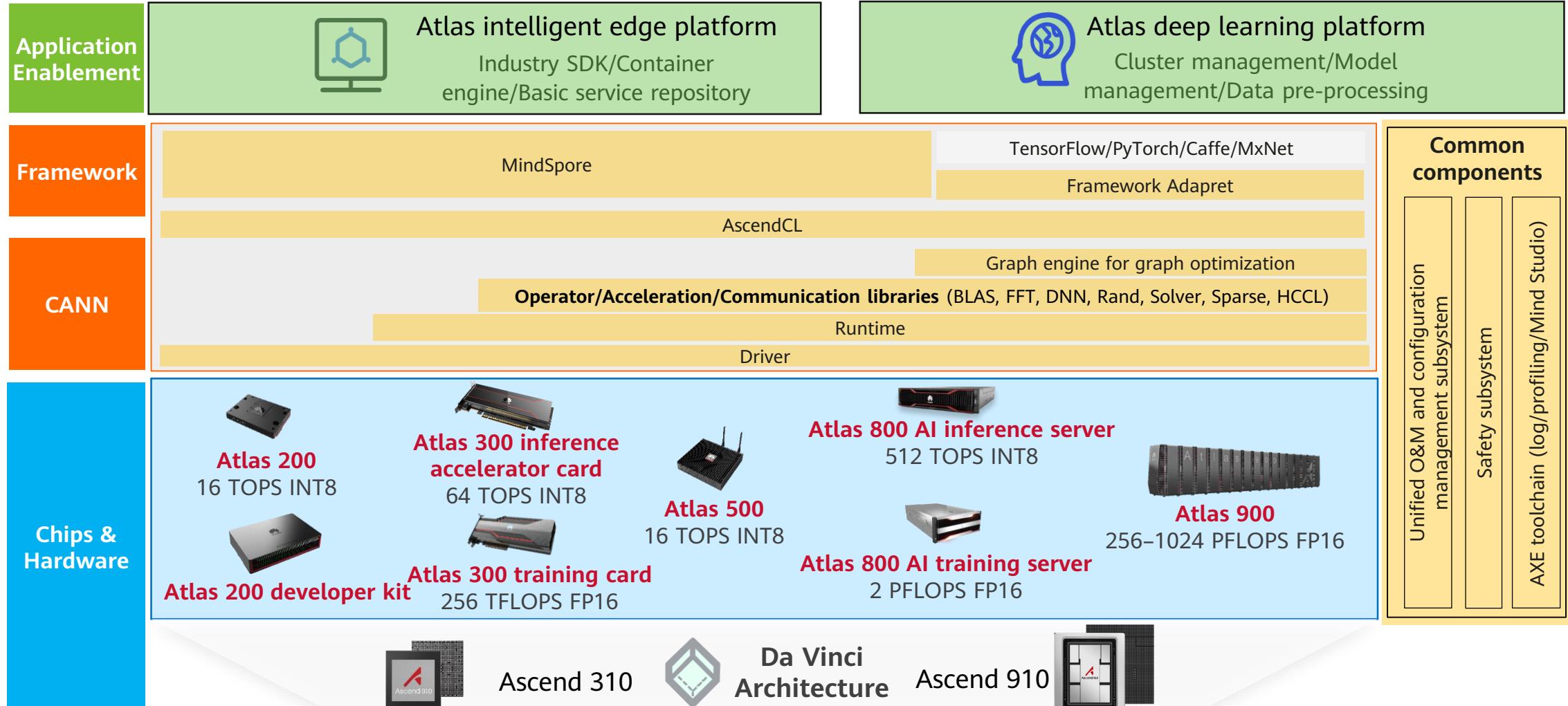
Most powerful AI processor

- Ascend-Max  
Architecture: Da Vinci
- Half-precision (FP16): 256 TFLOPS**
- Integer precision (INT8): 512 TOPS**
- 128-channel full HD video decoder: H.264/265**
- Max. power: 310 W**

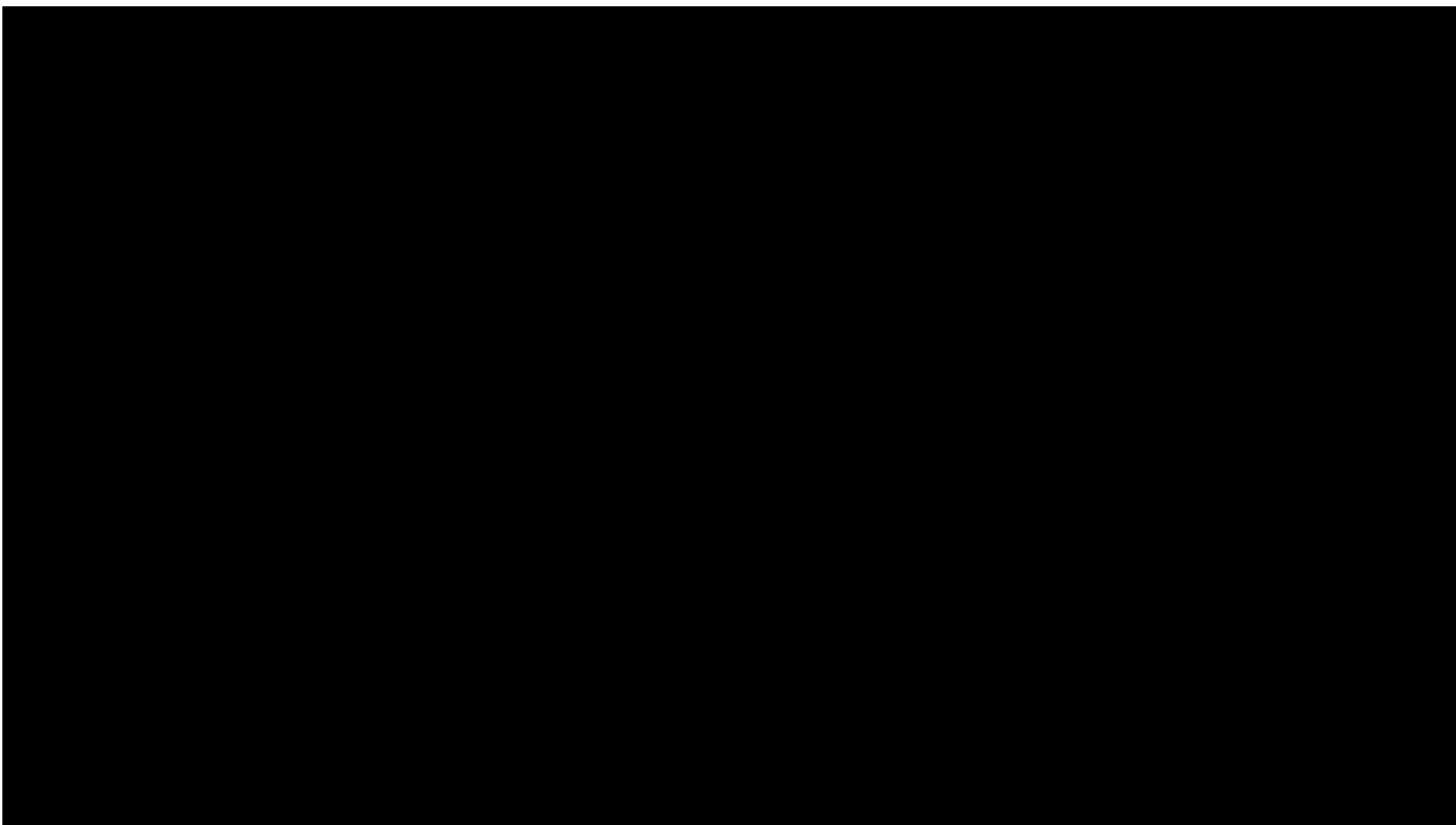


# Atlas AI Computing Platform Portfolio

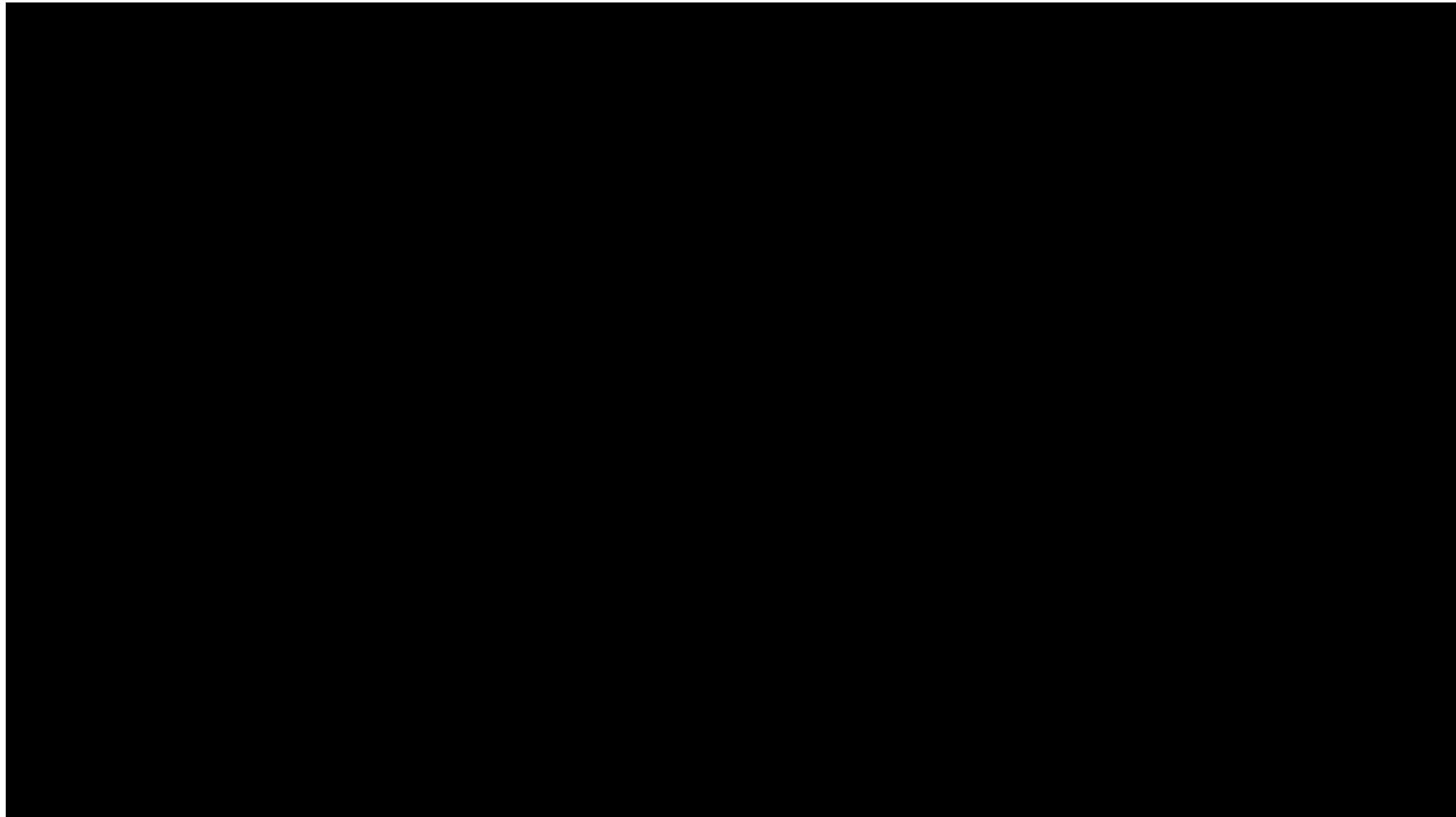
Internet, security, finance, transportation, power, etc.



# Huawei Atlas Computational Reasoning Platform



# HUAWEI CLOUD AI and HUAWEI Mobile Phones Help RFCx Protect the Rainforest



# Contents

---

1. AI Overview
2. Technical Fields and Application Fields of AI
3. Huawei's AI Development Strategy
- 4. AI Disputes**
5. Future Prospects of AI

# Algorithmic Bias

- Algorithmic biases are mainly caused by data biases.
- When we use AI algorithms for decision-making, the algorithms may learn to discriminate an individual based on existing data including race and gender, and therefore create unfair outcomes, such as decisions that are discriminatory based on race, sex or other factors. Even if factors such as race or gender are excluded from the data, the algorithms can make discriminatory decisions based on information of names and addresses.

If we search with a name sounds like an African American, an advertisement for **a tool used to search criminal records** may be displayed. The advertisement, however, is not likely displayed in other cases.

Online advertisers tend to display advertisements of lower-priced goods to **female users**.

Google's image software once mistakenly labeled an image of black people as "**gorilla**".

# Privacy Issues

- The existing AI algorithms are all data-driven. In this case, we need a large amount of data to train models. We enjoy the convenience brought by AI every day while technology companies like Facebook, Google, Amazon, and Alibaba are obtaining an enormous amount of user data, which will reveal various aspects of our lives including politics, religions, and gender.

**In principle, technology companies can record each click, each page scrolling, time of viewing any content, and browsing history when users access the Internet.**

**Technology companies can know our privacy including where we are, where we go, what we have done, education background, consumption capabilities, and personal preferences based on our ride-hailing records and consumption records.**

# Seeing = Believing?

- With the development of computer vision technologies, reliability of images and videos is decreasing. Fake images can be produced with technologies such as PS and generative adversarial networks (GAN), making it hard to identify whether images are true or not.
- Example:
  - A suspect provided fake evidence by forging an image in which the suspect is in a place where he has never been to or with someone he has never seen using PS technologies.
  - In advertisements for diet pills, people's appearances before and after weight loss can be changed with PS technologies to exaggerate the effect of the pills.
  - Lyrebird, a tool for simulating voice of human beings based on recording samples of minutes, may be used by criminals.
  - Household images released on rent and hotel booking platforms may be generated through GAN.

# AI Development = Rising Unemployment?

- Looking back, human beings have always been seeking ways to improve efficiency, that is, obtain more with less resources. We used sharp stones to hunt and collect food more efficiently. We used steam engines to reduce the need for horses. Every step in achieving automation will change our life and work. In the era of AI, what jobs will be replaced by AI?
- The answer is repetitive jobs that involve little creativity and social interaction.

Jobs Most Likely to Be Replaced by AI	Jobs Most Unlikely to Be Replaced by AI
Courier	Writer
Taxi driver	Management personnel
Soldier	Software engineers
Accounting	HR manager
Telesales personnel	Designer
Customer service	Activity planner
...	...

# Problems to Be Solved

- Are AI-created works protected by copyright laws?
- Who gives authority to robots?
- What rights shall be authorized to robots?
- ...

# Contents

1. AI Overview
2. Technical Fields and Application Fields of AI
3. Huawei's AI Development Strategy
4. AI Disputes
- 5. Future Prospects of AI**

# Development Trends of AI Technologies

- Framework: easier-to-use development framework
- Algorithm: algorithm models with better performance and smaller size
- Computing power: comprehensive development of device-edge-cloud computing
- Data: more comprehensive basic data service industry and more secure data sharing
- Scenario: continuous breakthroughs in industry applications

# Easier-to-Use Development Framework

- Various AI development frameworks are evolving towards ease-of-use and omnipotent, continuously lowering the threshold for AI development.



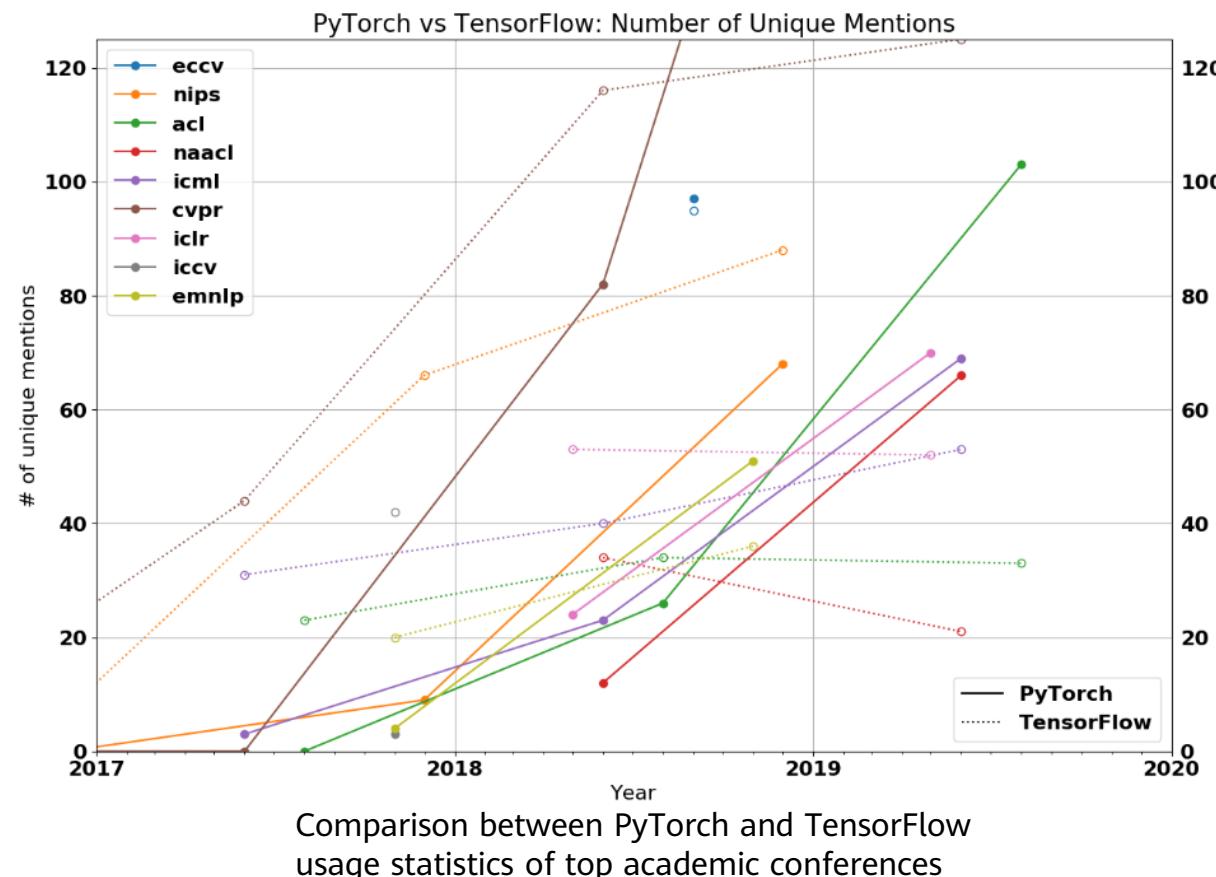
# Tensorflow 2.0

- TensorFlow 2.0 has been officially released. It integrates Keras as its high-level API, greatly improving usability.



# Pytorch vs Tensorflow

- PyTorch is widely recognized by academia for its ease of use.

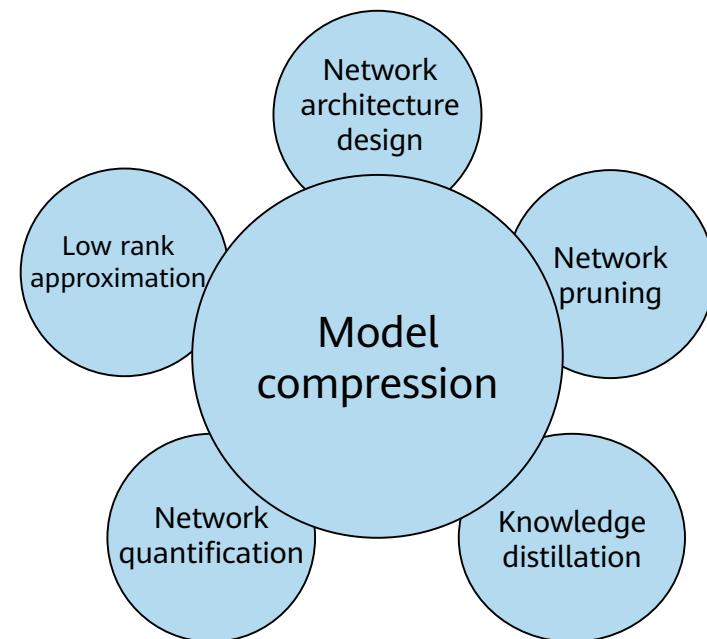


# Algorithms Model with Better Performance

- In the computer vision field, GAN has been able to generate high-quality images that cannot be identified by human eyes. GAN-related algorithms have been applied to other vision-related tasks, such as semantic segmentation, facial recognition, video synthesis, and unsupervised clustering.
- In the NLP field, the pre-training model based on the Transformer architecture has made a significant breakthrough. Related models such as BERT, GPT, and XLNet are widely used in industrial scenarios.
- In the reinforcement learning field, AlphaStar of the DeepMind team defeated the top human player in StarCraft II.
- ...

# Smaller Deep Learning Models

- A model with better performance usually has a larger quantity of parameters, and a large model has lower running efficiency in industrial applications. More and more model compression technologies are proposed to further compress the model size while ensuring the model performance, meeting the requirements of industrial applications.
  - Low rank approximation
  - Network pruning
  - Network quantification
  - Knowledge distillation
  - Compact network design



# Computing Power with Comprehensive Device-Edge-Cloud Development

- The scale of AI chips applied to the cloud, edge devices, and mobile devices keeps increasing, further meeting the computing power demand of AI.

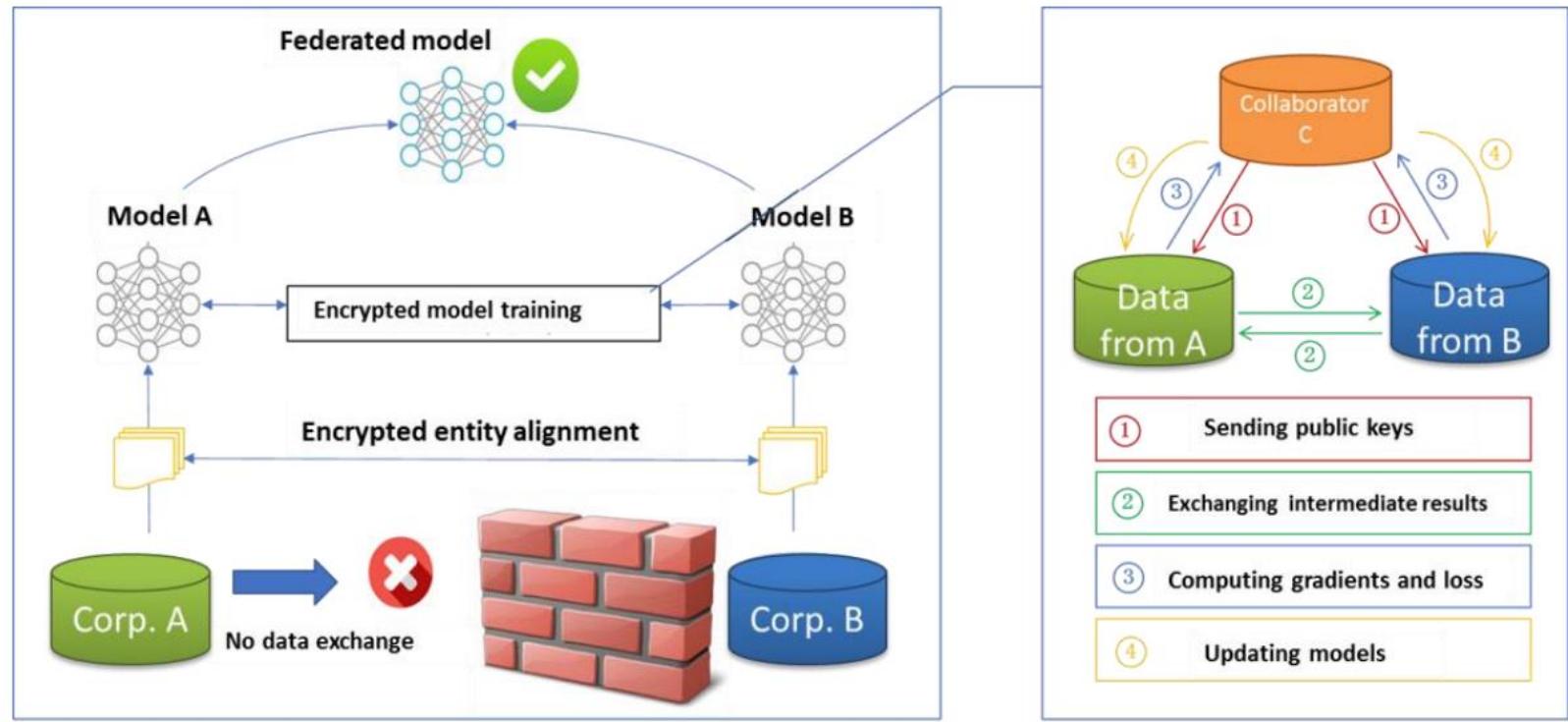


China AI Chip Industry Development White Paper 2020

Market Scale and Growth Prediction of AI Chips in China from 2020 to 2021

# More Secure Data Sharing

- Federated learning uses different data sources to train models, further breaking data bottlenecks while ensuring data privacy and security.



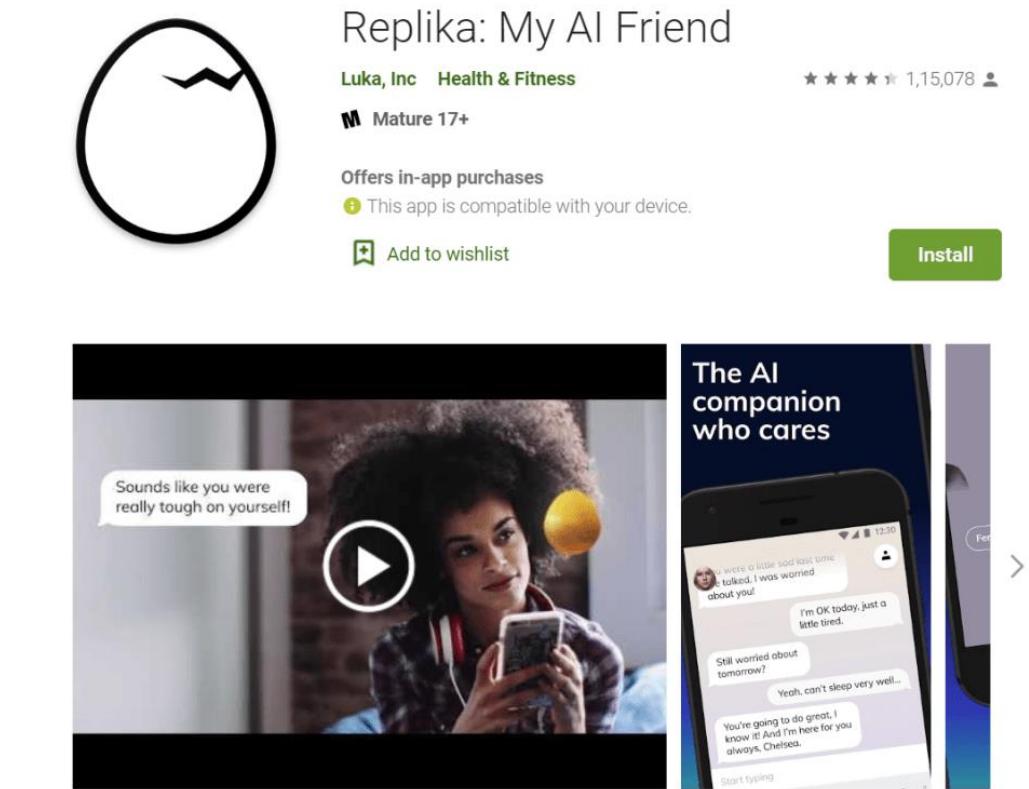
Federated Learning White Paper V1.0

# Continuous Breakthroughs in Application Scenarios

- With the continuous exploration of AI in various verticals, the application scenarios of AI will be continuously broken through.
  - Mitigating psychological problems
  - Automatic vehicle insurance and loss assessment
  - Office automation
  - ...

# Mitigating Psychological Problems

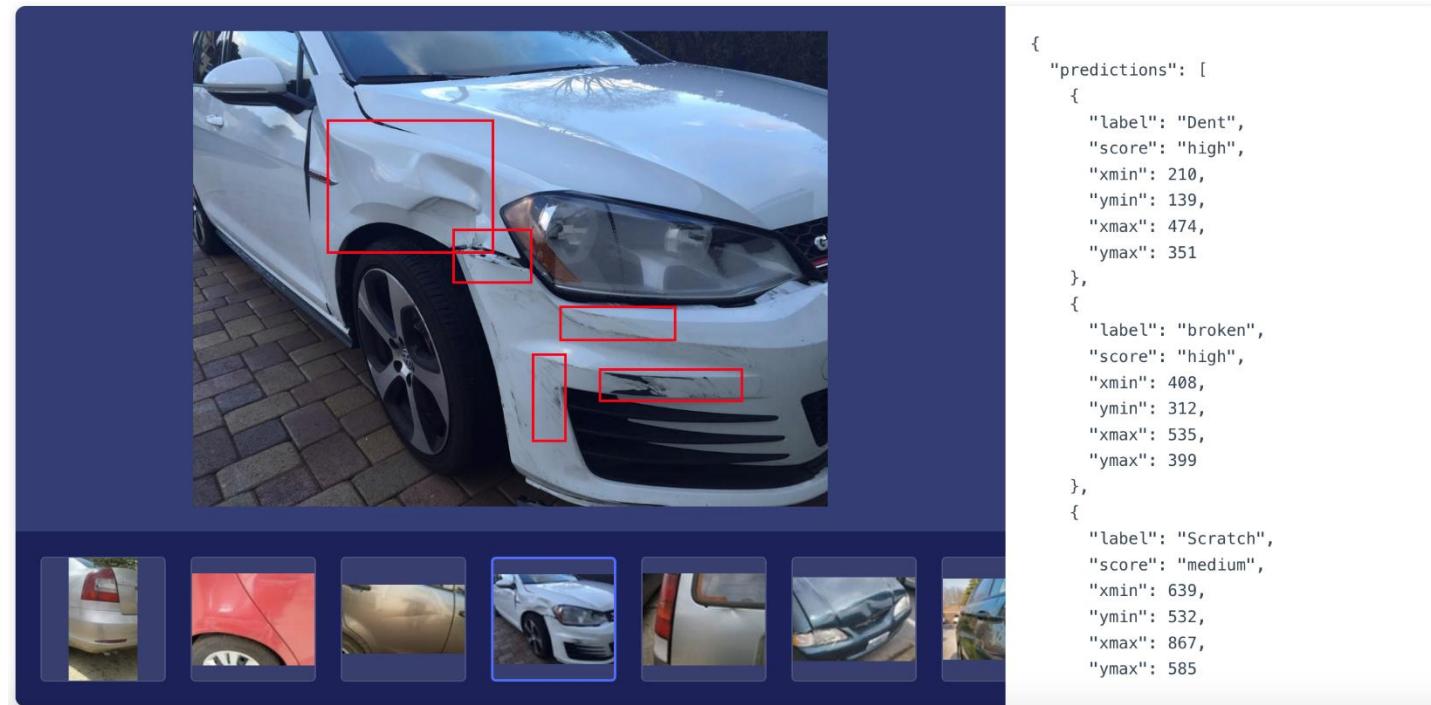
- AI chat robots help alleviate mental health problems such as autism by combining psychological knowledge.



# Automatic Vehicle Insurance and Loss Assessment

- AI technologies help insurance companies optimize vehicle insurance claims and complete vehicle insurance loss assessment using deep learning algorithms such as image recognition.

Vehicle Damage Assessment



# Office Automation

- AI is automating management, but the different nature and format of data makes it a challenging task. While each industry and application has its own unique challenges, different industries are gradually adopting machine learning-based workflow solutions.



# Summary

- This chapter introduces the definition and development history of AI, describes the technical fields and application fields of AI, briefly introduces Huawei's AI development strategy, and finally discusses the disputes and the development trends of AI.

# Quiz

1. (Multiple-answer question) Which of the following are AI application fields?
  - A. Smart household
  - B. Smart healthcare
  - C. Smart city
  - D. Smart education
2. (True or False) By "all AI scenarios", Huawei means different deployment scenarios for AI, including public clouds, private clouds, edge computing in all forms, industrial IoT devices, and consumer devices.
  - A. True
  - B. False

# More Information

---

Online learning website

- <https://e.huawei.com/en/talent/#/home>

Huawei Knowledge Base

- <https://support.huawei.com/enterprise/en/knowledge?lang=en>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

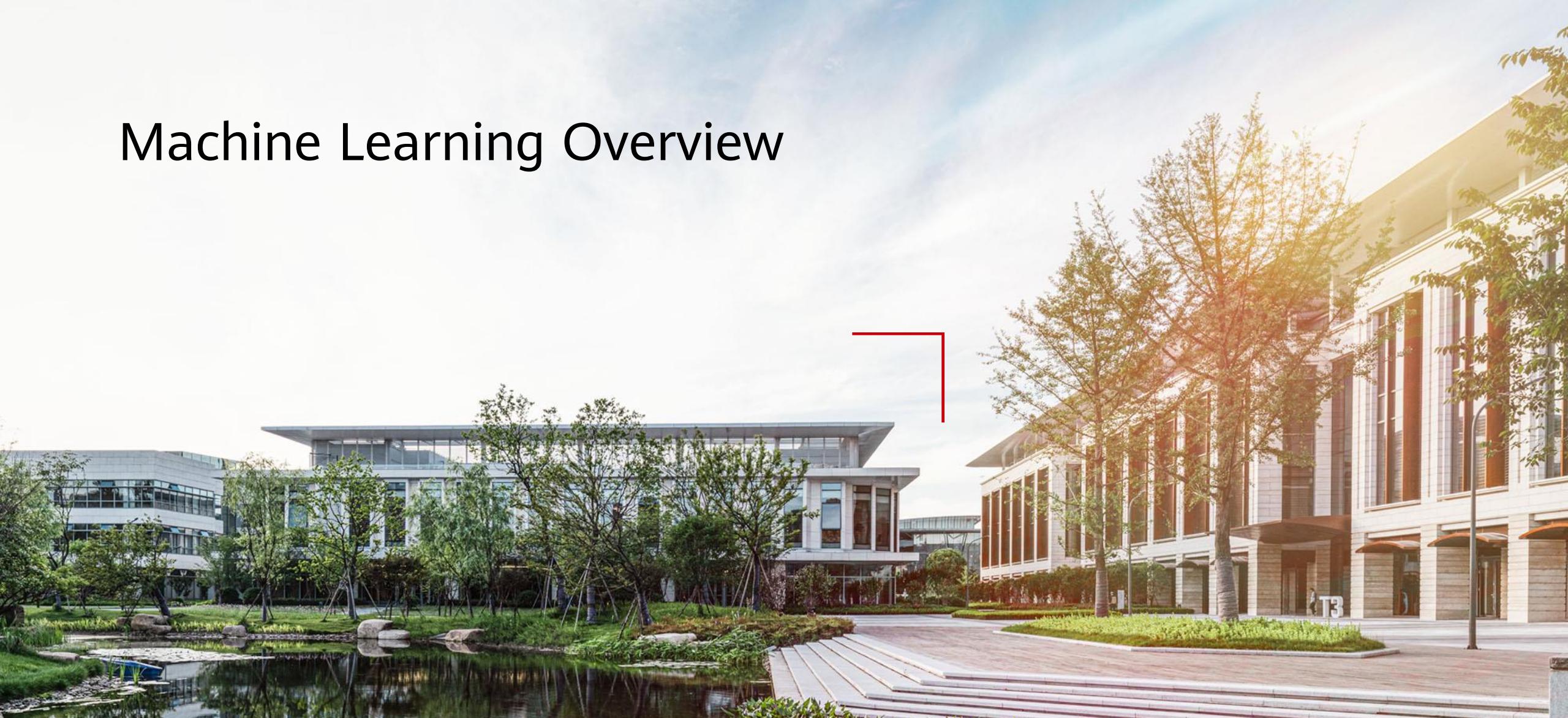
Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Machine Learning Overview



# Foreword

---

- Machine learning is a core research field of AI, and it is also a necessary knowledge for deep learning. Therefore, this chapter mainly introduces the main concepts of machine learning, the classification of machine learning, the overall process of machine learning, and the common algorithms of machine learning.

# Objectives

Upon completion of this course, you will be able to:

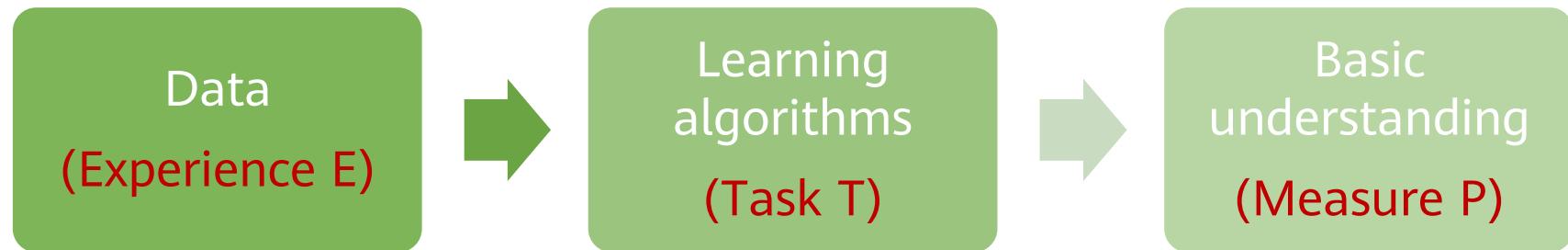
- Master the learning algorithm definition and machine learning process.
- Know common machine learning algorithms.
- Understand concepts such as hyperparameters, gradient descent, and cross validation.

# Contents

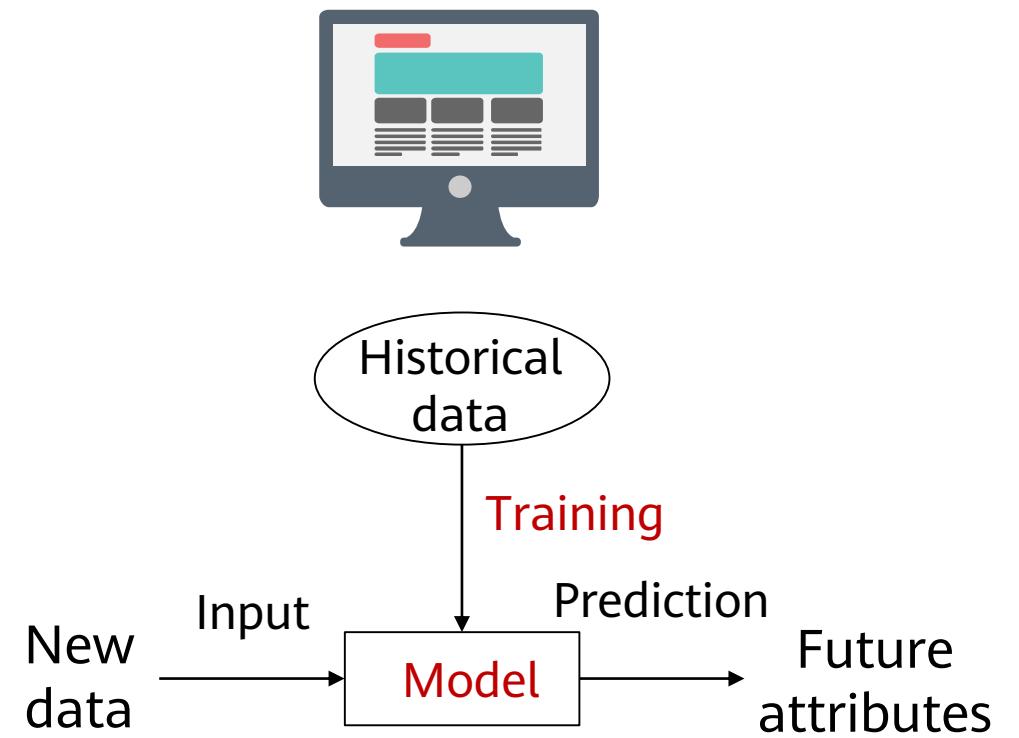
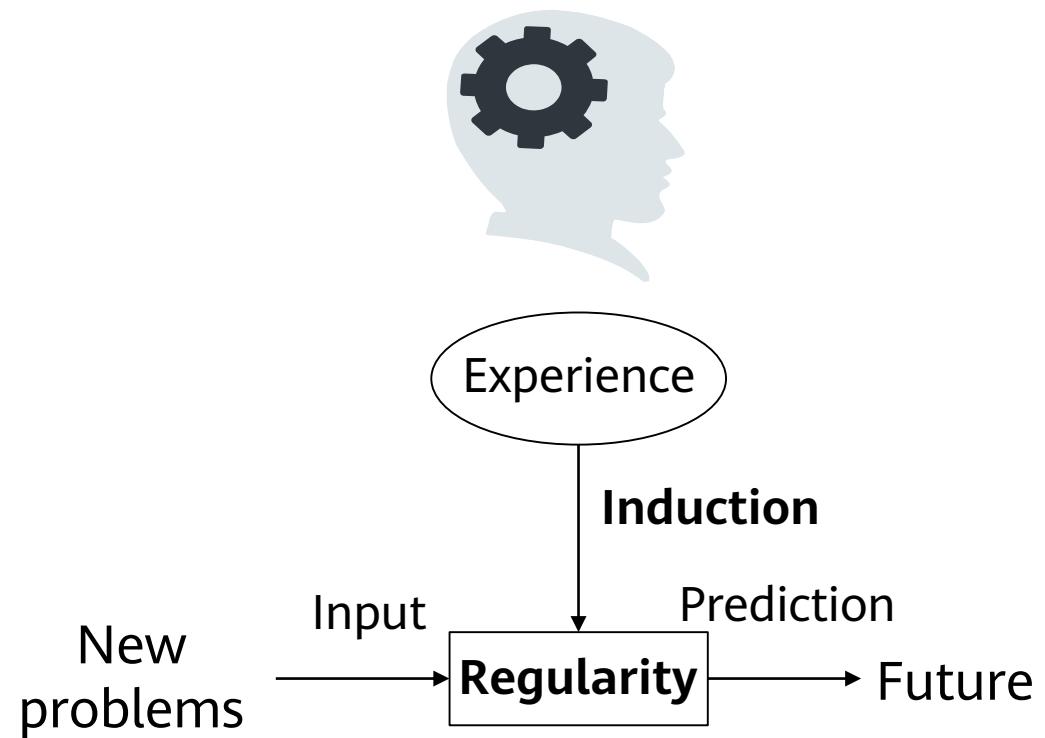
- 1. Machine Learning Definition**
2. Machine Learning Types
3. Machine Learning Process
4. Other Key Machine Learning Methods
5. Common Machine Learning Algorithms
6. Case Study

# Machine Learning Algorithms (1)

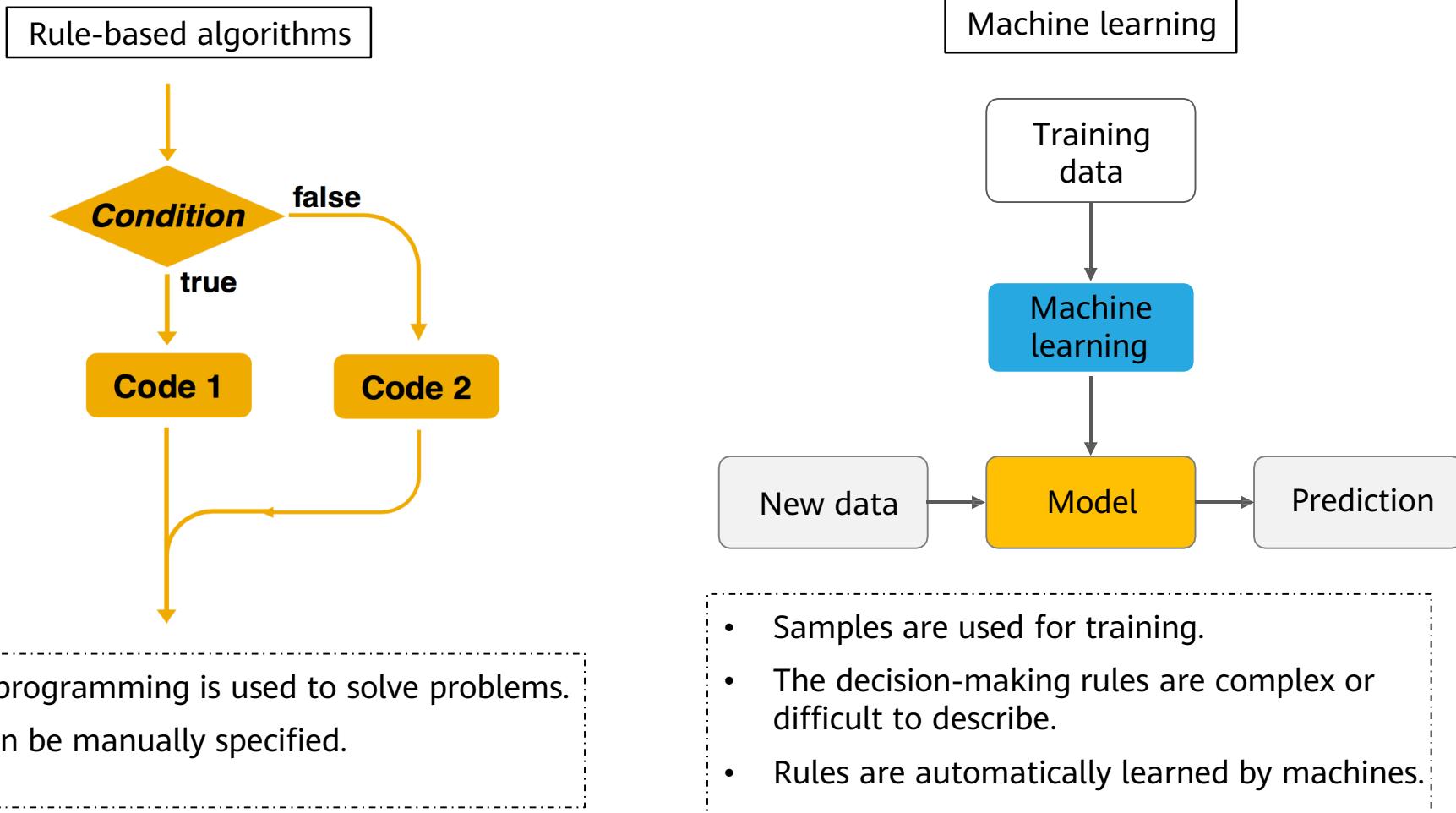
- Machine learning (including deep learning) is a study of learning algorithms. A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .



# Machine Learning Algorithms (2)



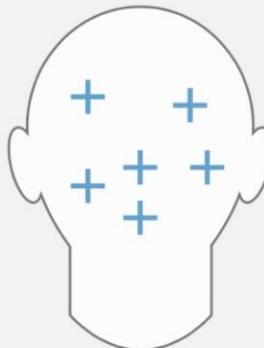
# Differences Between Machine Learning Algorithms and Traditional Rule-Based Algorithms



# Application Scenarios of Machine Learning (1)

- The solution to a problem is complex, or the problem may involve a large amount of data without a clear data distribution function.
- Machine learning can be used in the following scenarios:

*Rules are complex or cannot be described, such as facial recognition and voice recognition.*



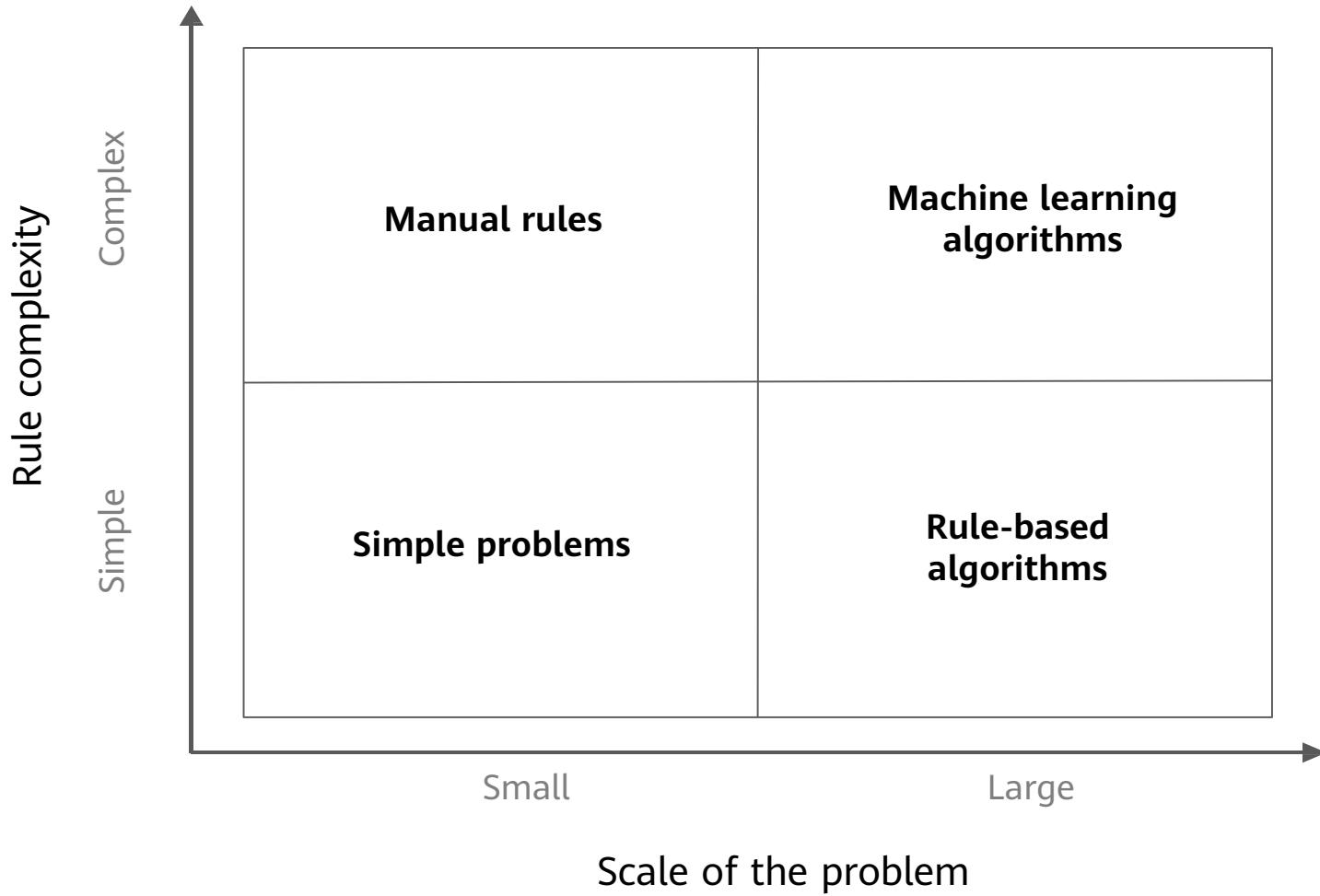
*Task rules change over time. For example, in the part-of-speech tagging task, new words or meanings are generated at any time.*



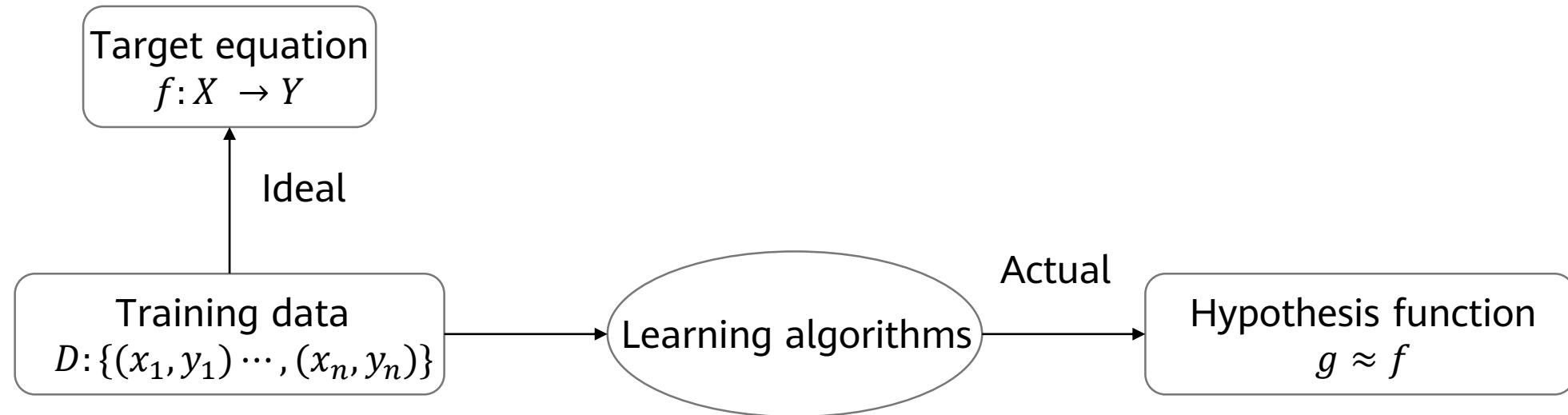
*Data distribution changes over time, requiring constant readaptation of programs, such as predicting the trend of commodity sales.*



# Application Scenarios of Machine Learning (2)



# Rational Understanding of Machine Learning Algorithms



- Target function  $f$  is unknown. Learning algorithms cannot obtain a perfect function  $f$ .
- Assume that hypothesis function  $g$  **approximates** function  $f$ , but may be different from function  $f$ .

# Main Problems Solved by Machine Learning

- Machine learning can deal with many types of tasks. The following describes the most typical and common types of tasks.
  - Classification: A computer program needs to specify which of the  $k$  categories some input belongs to. To accomplish this task, learning algorithms usually output a function  $f: R^n \rightarrow (1, 2, \dots, k)$ . For example, the image classification algorithm in computer vision is developed to handle classification tasks.
  - Regression: For this type of task, a computer program predicts the output for the given input. Learning algorithms typically output a function  $f: R^n \rightarrow R$ . An example of this task type is to predict the claim amount of an insured person (to set the insurance premium) or predict the security price.
  - Clustering: A large amount of data from an unlabeled dataset is divided into multiple categories according to internal similarity of the data. Data in the same category is more similar than that in different categories. This feature can be used in scenarios such as image retrieval and user profile management.
- Classification and regression are two main types of prediction, accounting from 80% to 90%. The output of classification is discrete category values, and the output of regression is continuous numbers.

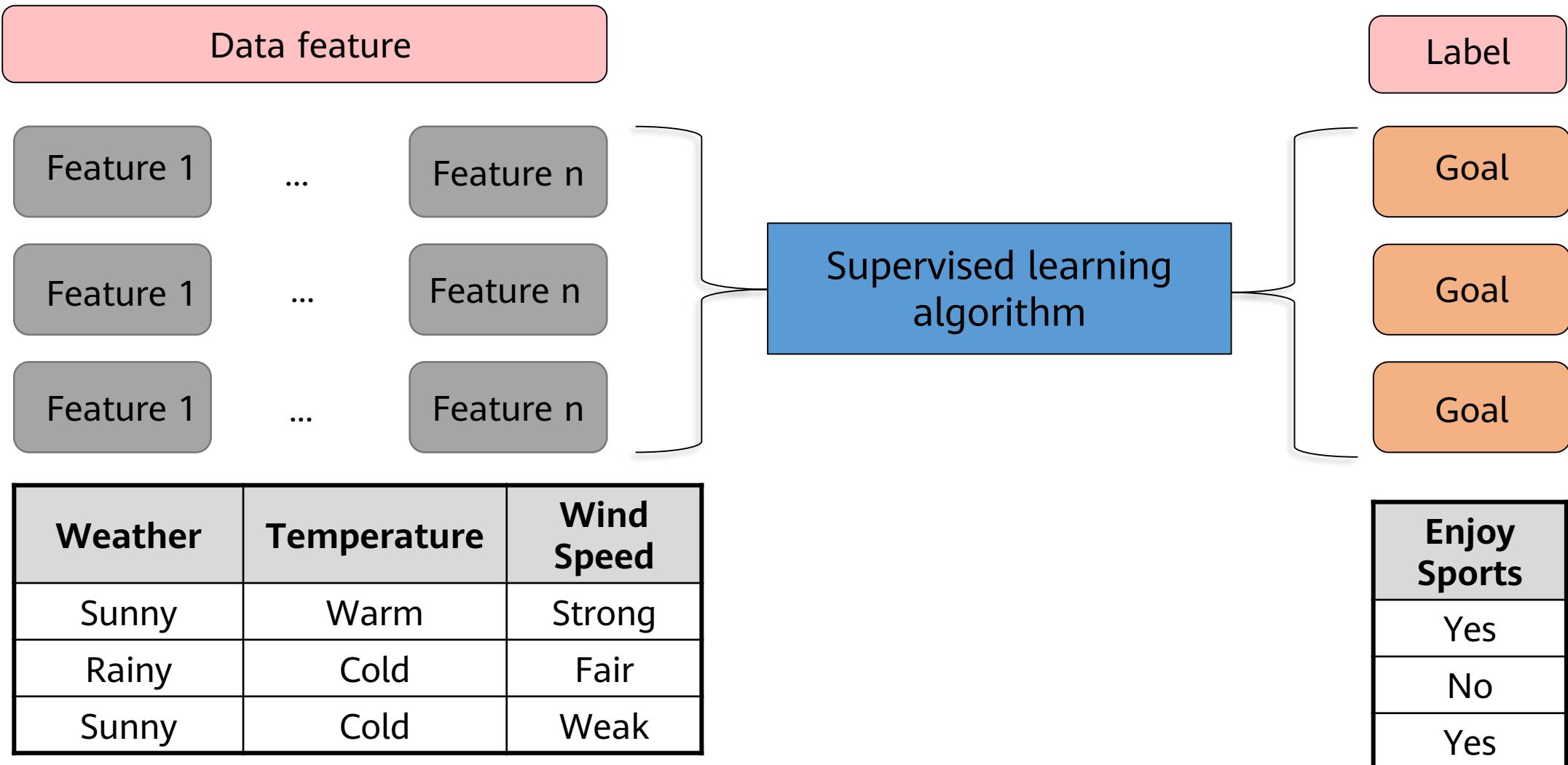
# Contents

1. Machine Learning Definition
- 2. Machine Learning Types**
3. Machine Learning Process
4. Other Key Machine Learning Methods
5. Common Machine Learning Algorithms
6. Case study

# Machine Learning Classification

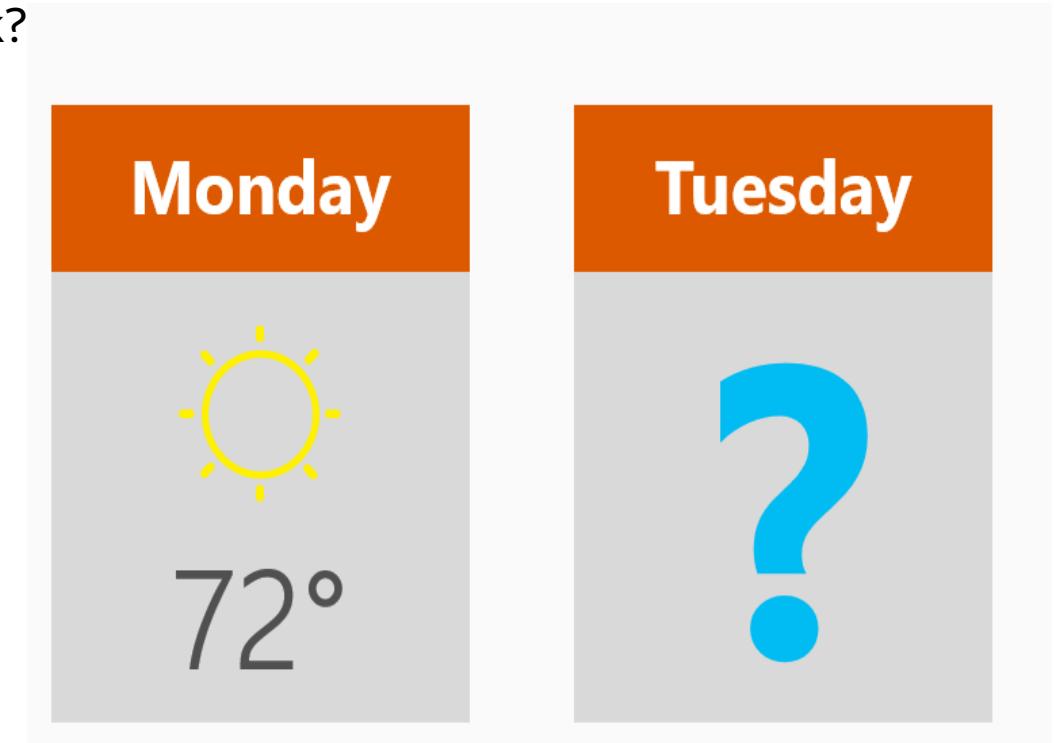
- **Supervised learning:** Obtain an optimal model with required performance through training and learning based on the samples of known categories. Then, use the model to map all inputs to outputs and check the output for the purpose of classifying unknown data.
- **Unsupervised learning:** For unlabeled samples, the learning algorithms directly model the input datasets. Clustering is a common form of unsupervised learning. We only need to put highly similar samples together, calculate the similarity between new samples and existing ones, and classify them by similarity.
- **Semi-supervised learning:** In one task, a machine learning model that automatically uses a large amount of unlabeled data to assist learning directly of a small amount of labeled data.
- **Reinforcement learning:** It is an area of machine learning concerned with how agents ought to take actions in an environment to maximize some notion of cumulative reward. The difference between reinforcement learning and supervised learning is the teacher signal. The reinforcement signal provided by the environment in reinforcement learning is used to evaluate the action (scalar signal) rather than telling the learning system how to perform correct actions.

# Supervised Learning



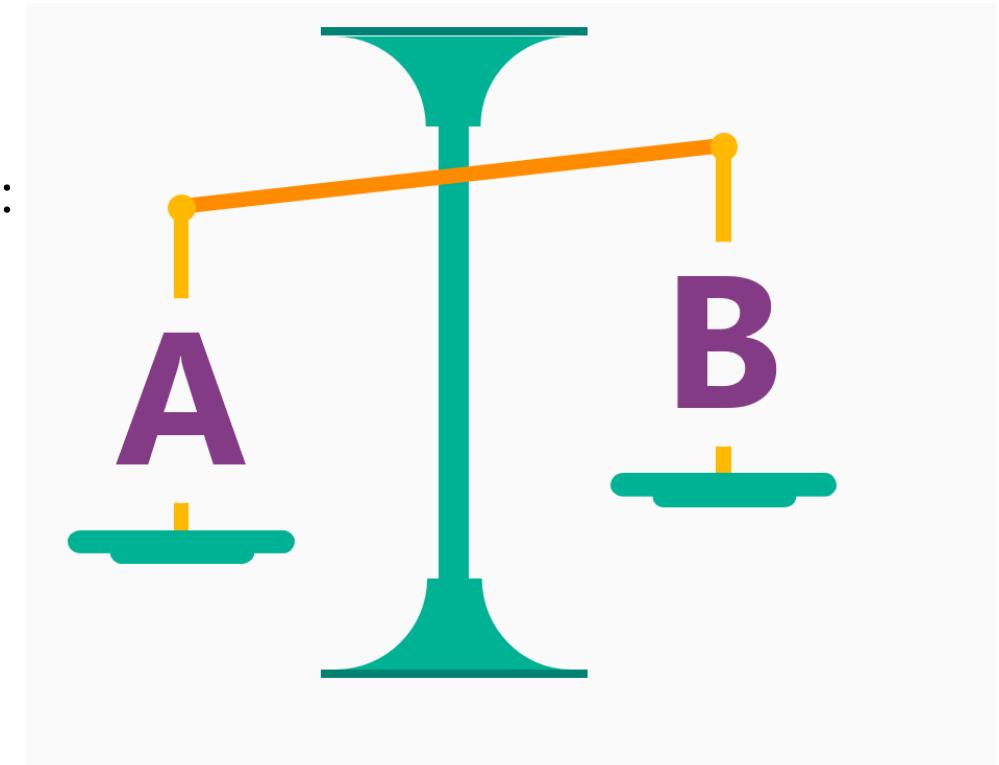
# Supervised Learning - Regression Questions

- Regression: reflects the features of attribute values of samples in a sample dataset. The dependency between attribute values is discovered by expressing the relationship of sample mapping through functions.
  - How much will I benefit from the stock next week?
  - What's the temperature on Tuesday?

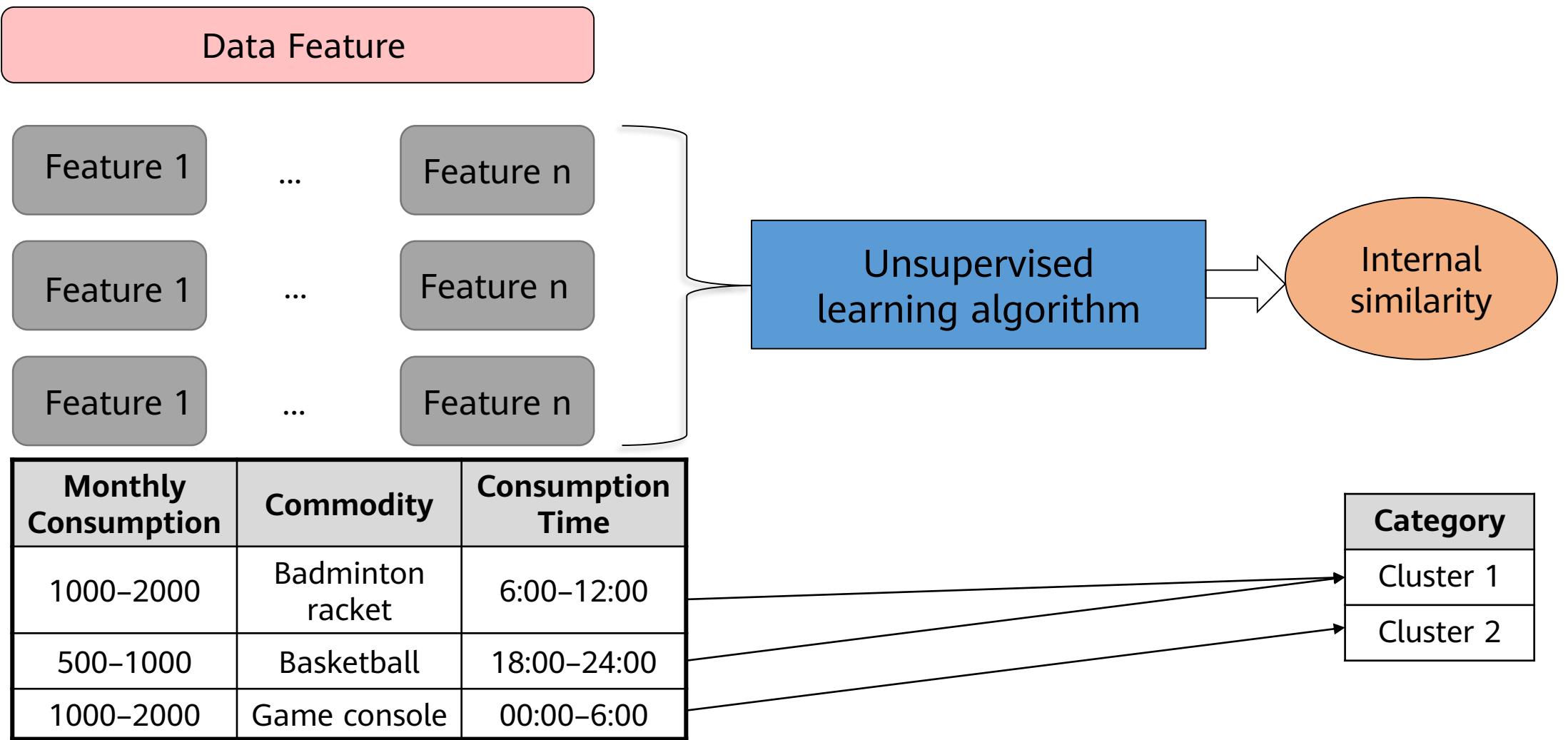


# Supervised Learning - Classification Questions

- Classification: maps samples in a sample dataset to a specified category by using a classification model.
  - Will there be a traffic jam on XX road during the morning rush hour tomorrow?
  - Which method is more attractive to customers: 5 yuan voucher or 25% off?

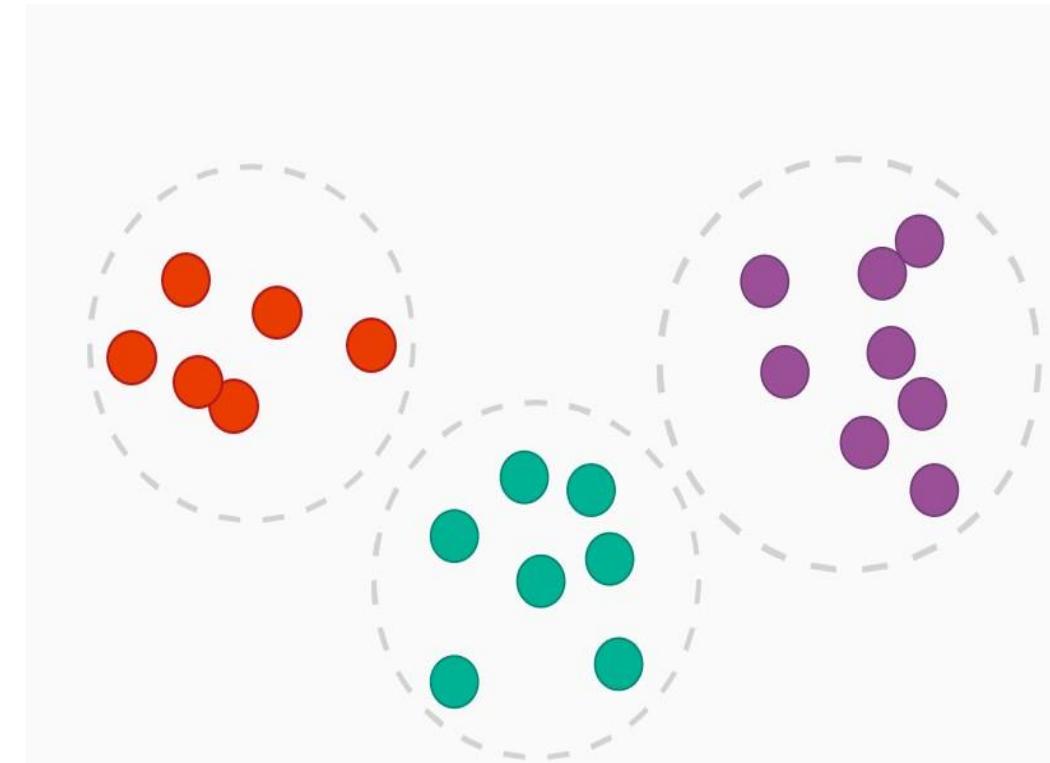


# Unsupervised Learning

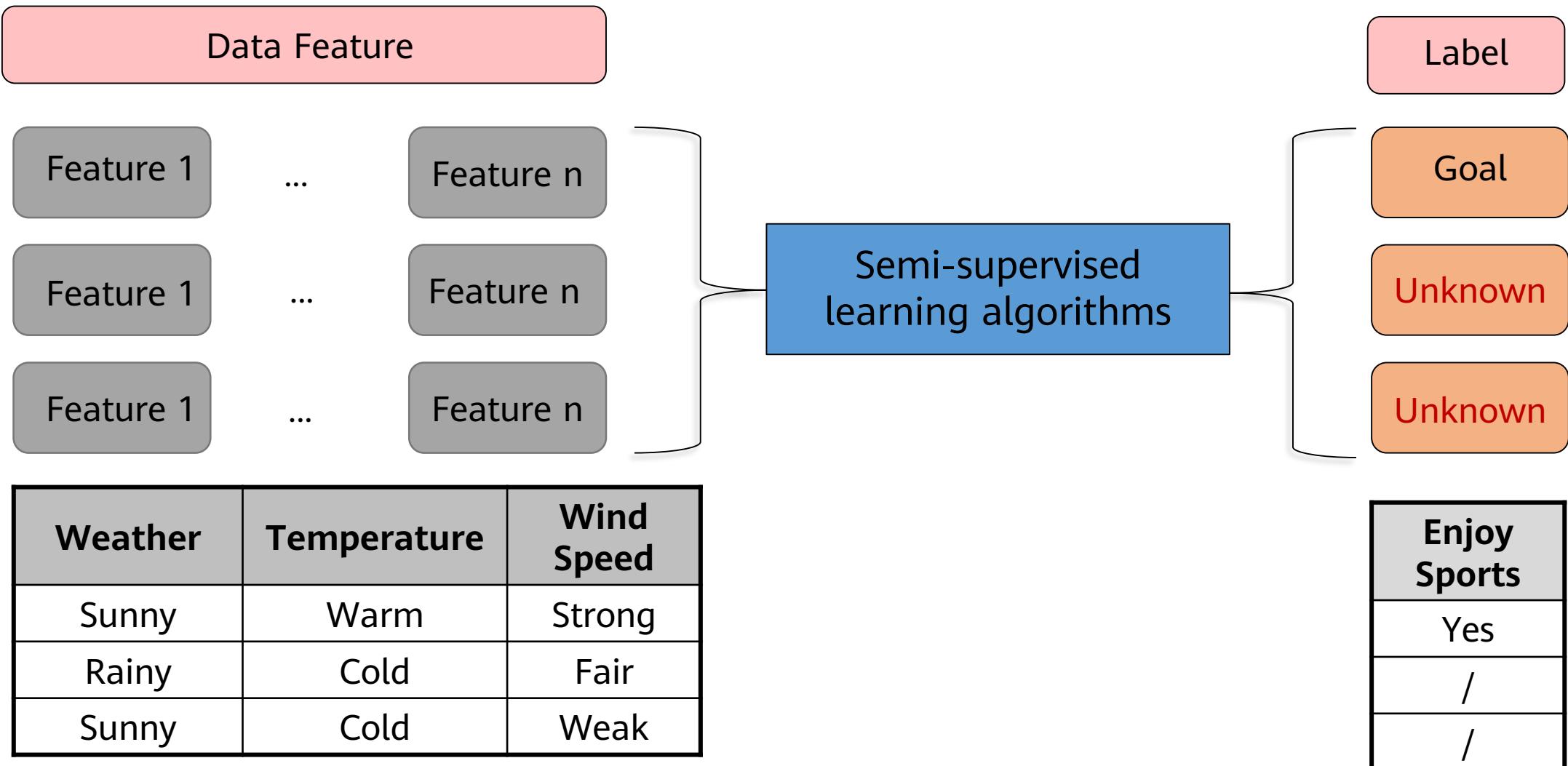


# Unsupervised Learning - Clustering Questions

- Clustering: classifies samples in a sample dataset into several categories based on the clustering model. The similarity of samples belonging to the same category is high.
  - Which audiences like to watch movies of the same subject?
  - Which of these components are damaged in a similar way?

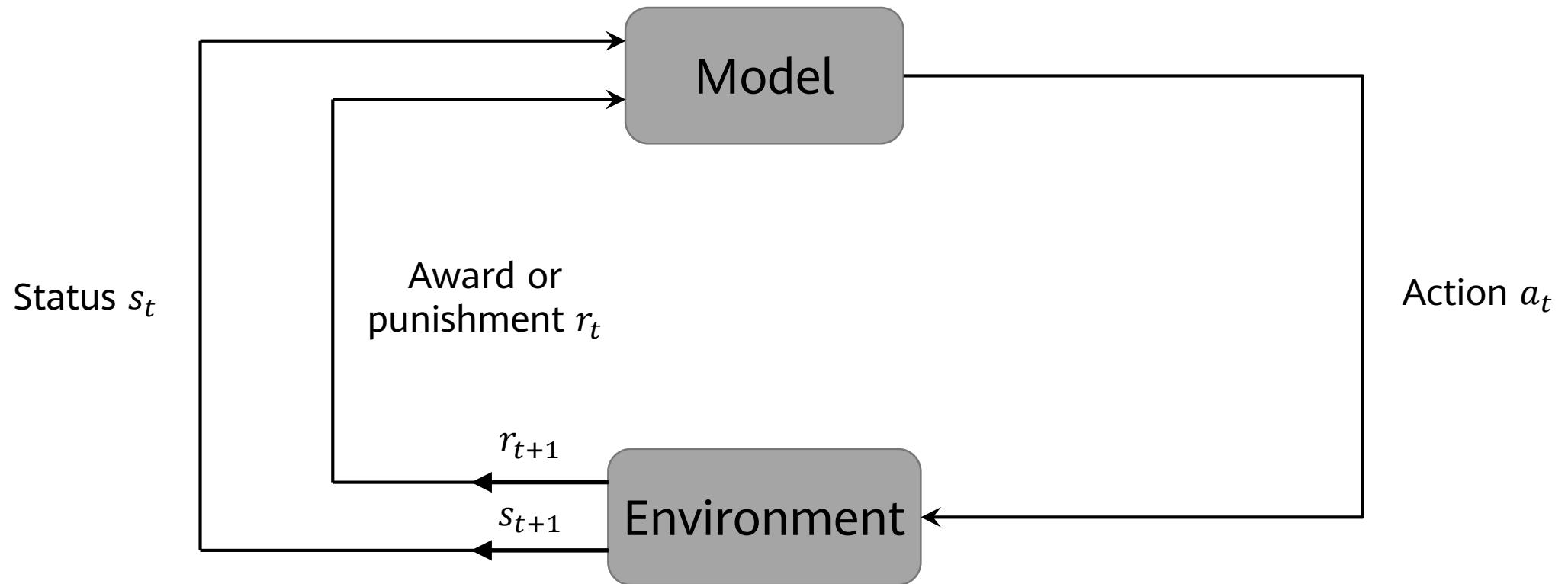


# Semi-Supervised Learning



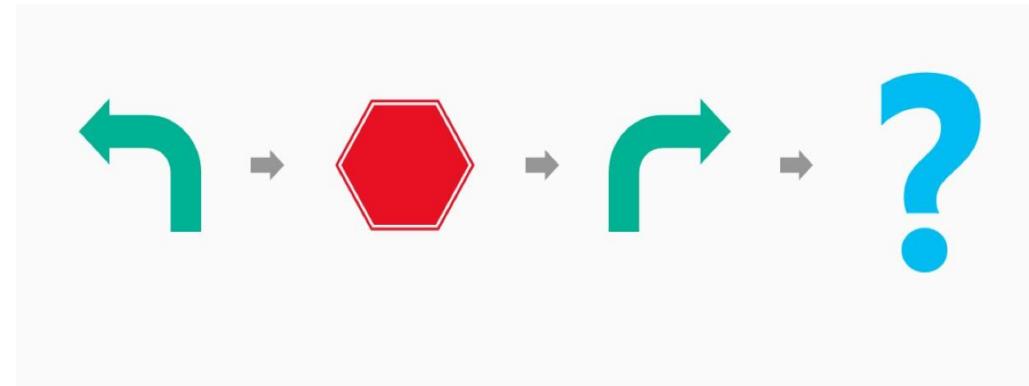
# Reinforcement Learning

- The model perceives the environment, takes actions, and makes adjustments and choices based on the status and award or punishment.



# Reinforcement Learning - Best Behavior

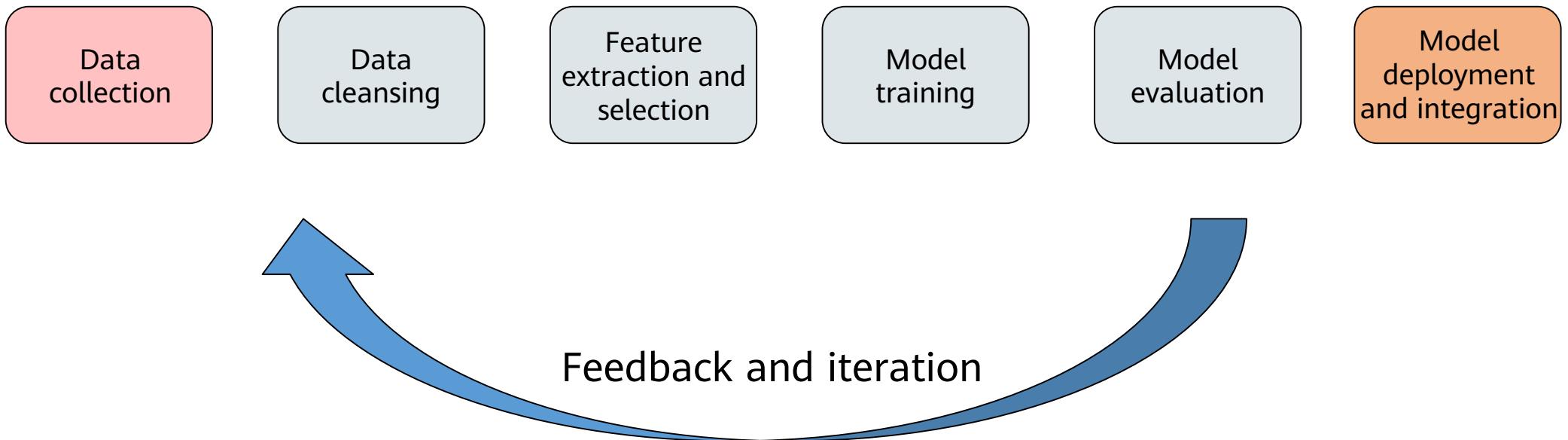
- Reinforcement learning: always looks for best behaviors. Reinforcement learning is targeted at machines or robots.
  - Autopilot: Should it brake or accelerate when the yellow light starts to flash?
  - Cleaning robot: Should it keep working or go back for charging?



# Contents

1. Machine learning algorithm
2. Machine Learning Classification
- 3. Machine Learning Process**
4. Other Key Machine Learning Methods
5. Common Machine Learning Algorithms
6. Case study

# Machine Learning Process



# Basic Machine Learning Concept — Dataset

- Dataset: a collection of data used in machine learning tasks. Each data record is called a sample. Events or attributes that reflect the performance or nature of a sample in a particular aspect are called features.
- Training set: a dataset used in the training process, where each sample is referred to as a training sample. The process of creating a model from data is called learning (training).
- Test set: Testing refers to the process of using the model obtained after learning for prediction. The dataset used is called a test set, and each sample is called a test sample.

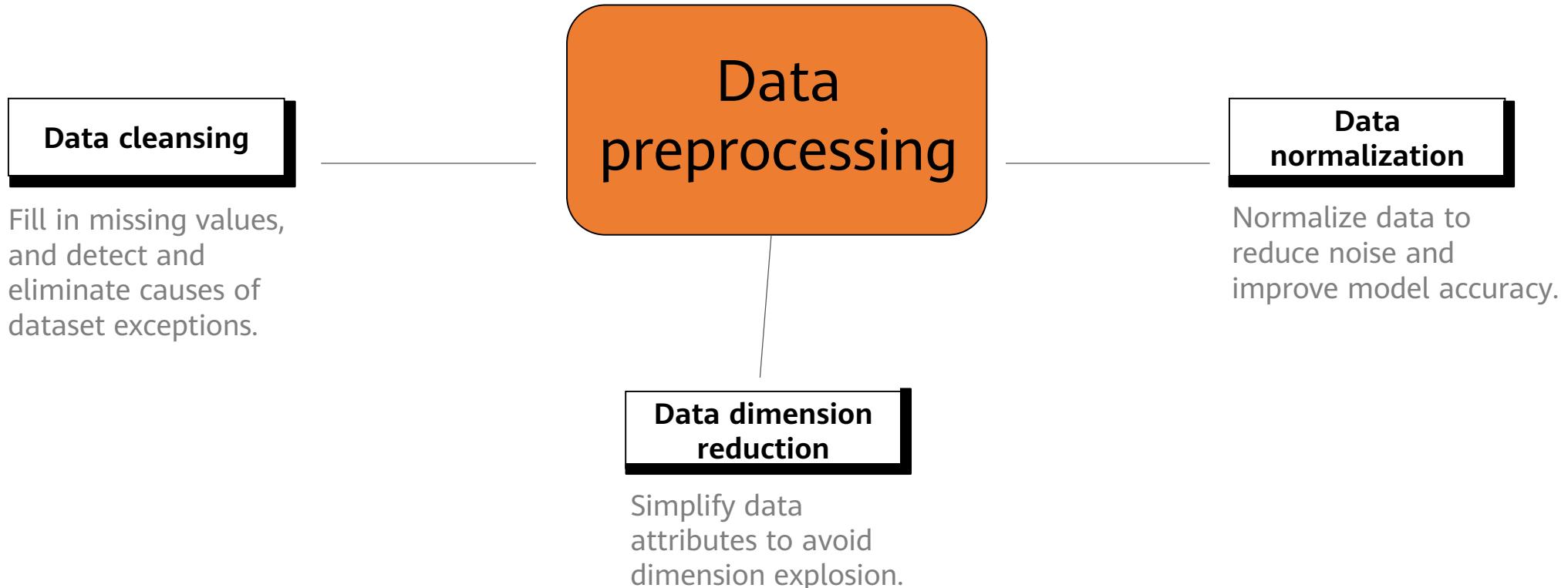
# Checking Data Overview

- Typical dataset form

	Feature 1	Feature 2	Feature 3	Label
No.	Area	School Districts	Direction	House Price
Training set	1	100	8	South
	2	120	9	Southwest
	3	60	6	North
	4	80	9	Southeast
Test set	5	95	3	South
				850

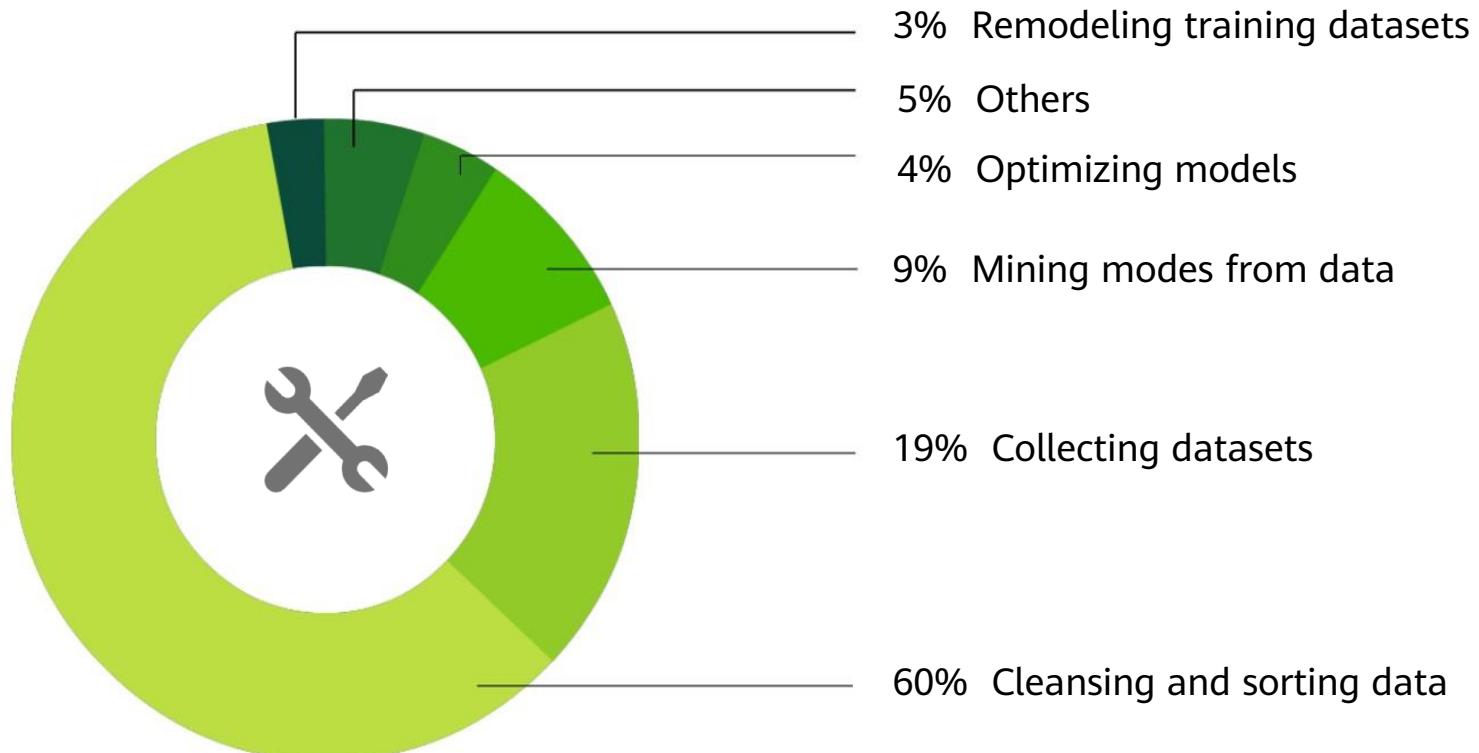
# Importance of Data Processing

- Data is crucial to models. It is the ceiling of model capabilities. Without good data, there is no good model.



# Workload of Data Cleansing

- Statistics on data scientists' work in machine learning



*CrowdFlower Data Science Report 2016*

# Data Cleansing

- Most machine learning models process features, which are usually numeric representations of input variables that can be used in the model.
- In most cases, the collected data can be used by algorithms only after being preprocessed. The preprocessing operations include the following:
  - Data filtering
  - Processing of lost data
  - Processing of possible exceptions, errors, or abnormal values
  - Combination of data from multiple data sources
  - Data consolidation

# Dirty Data (1)

- Generally, real data may have some quality problems.
  - Incompleteness: contains missing values or the data that lacks attributes
  - Noise: contains incorrect records or exceptions.
  - Inconsistency: contains inconsistent records.

# Dirty Data (2)

#	Id	Name	Birthday	Gender	IsTe acher	#Stu dent s	Country	City
1	111	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	Spain	Madri d
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerlan d	Genev a
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

Annotations pointing to dirty data:

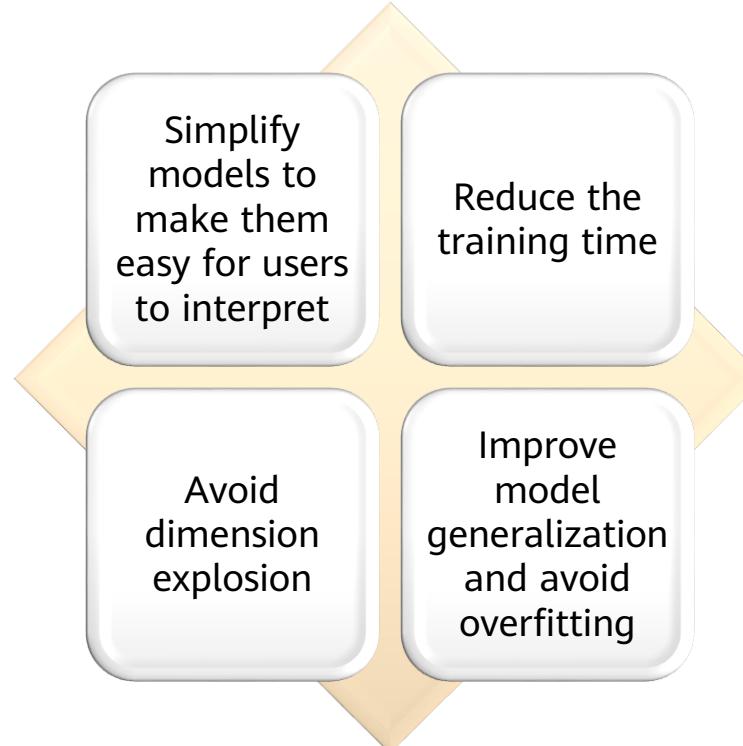
- Invalid duplicate item: Points to the Id column of row 6.
- Incorrect format: Points to the Birthday column of row 6.
- Attribute dependency: Points to the IsTeacher column of row 9.
- Missing value: Points to the City column of row 2.
- Invalid value: Points to the Gender column of row 5.
- Value that should be in another column: Points to the City column of row 7.
- Misspelling: Points to the City column of row 10.

# Data Conversion

- After being preprocessed, the data needs to be converted into a representation form suitable for the machine learning model. Common data conversion forms include the following:
  - With respect to classification, category data is encoded into a corresponding numerical representation.
  - Value data is converted to category data to reduce the value of variables (for age segmentation).
  - Other data
    - In the text, the word is converted into a word vector through word embedding (generally using the word2vec model, BERT model, etc).
    - Process image data (color space, grayscale, geometric change, Haar feature, and image enhancement)
  - Feature engineering
    - Normalize features to ensure the same value ranges for input variables of the same model.
    - Feature expansion: Combine or convert existing variables to generate new features, such as the average.

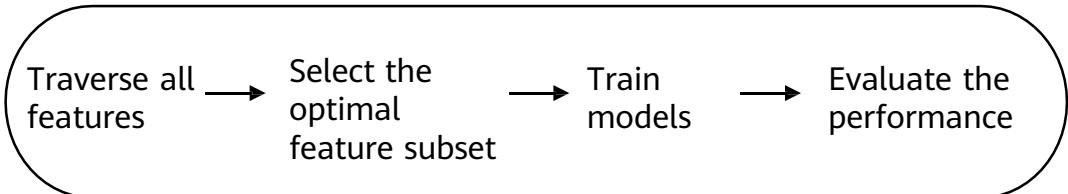
# Necessity of Feature Selection

- Generally, a dataset has many features, some of which may be redundant or irrelevant to the value to be predicted.
- Feature selection is necessary in the following aspects:



# Feature Selection Methods - Filter

- Filter methods are independent of the model during feature selection.



Procedure of a filter method

By evaluating the correlation between each feature and the target attribute, these methods use a statistical measure to assign a value to each feature. Features are then sorted by score, which is helpful for preserving or eliminating specific features.

## Common methods

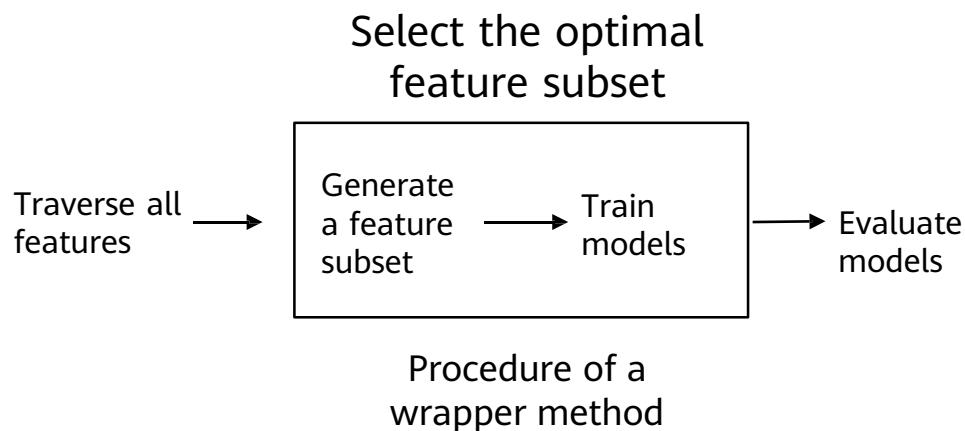
- Pearson correlation coefficient
- Chi-square coefficient
- Mutual information

## Limitations

- The filter method tends to select redundant variables as the relationship between features is not considered.

# Feature Selection Methods - Wrapper

- Wrapper methods use a prediction model to score feature subsets.



Wrapper methods consider feature selection as a search issue for which different combinations are evaluated and compared. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy.

## Common methods

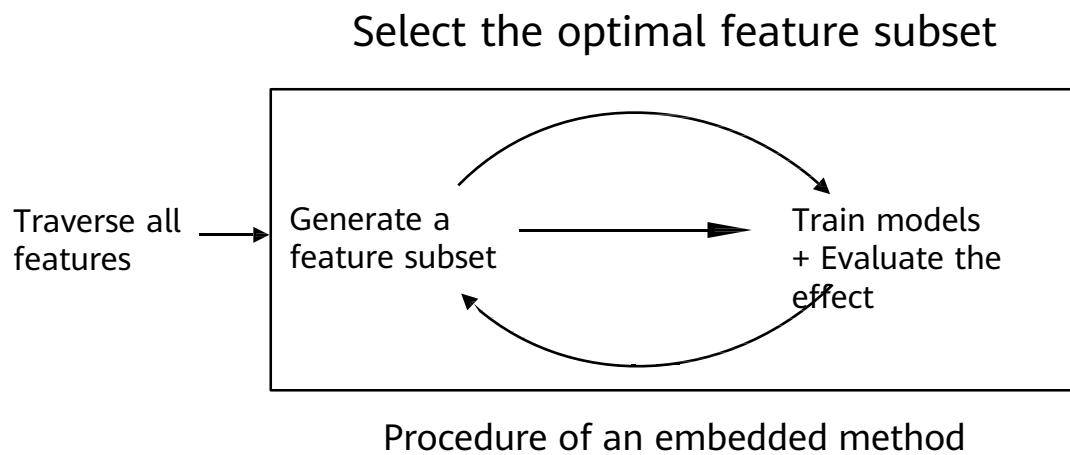
- Recursive feature elimination (RFE)

## Limitations

- Wrapper methods train a new model for each subset, resulting in **a huge number of computations**.
- A feature set with the best performance is usually provided for a specific type of model.

# Feature Selection Methods - Embedded

- Embedded methods consider feature selection as a part of model construction.

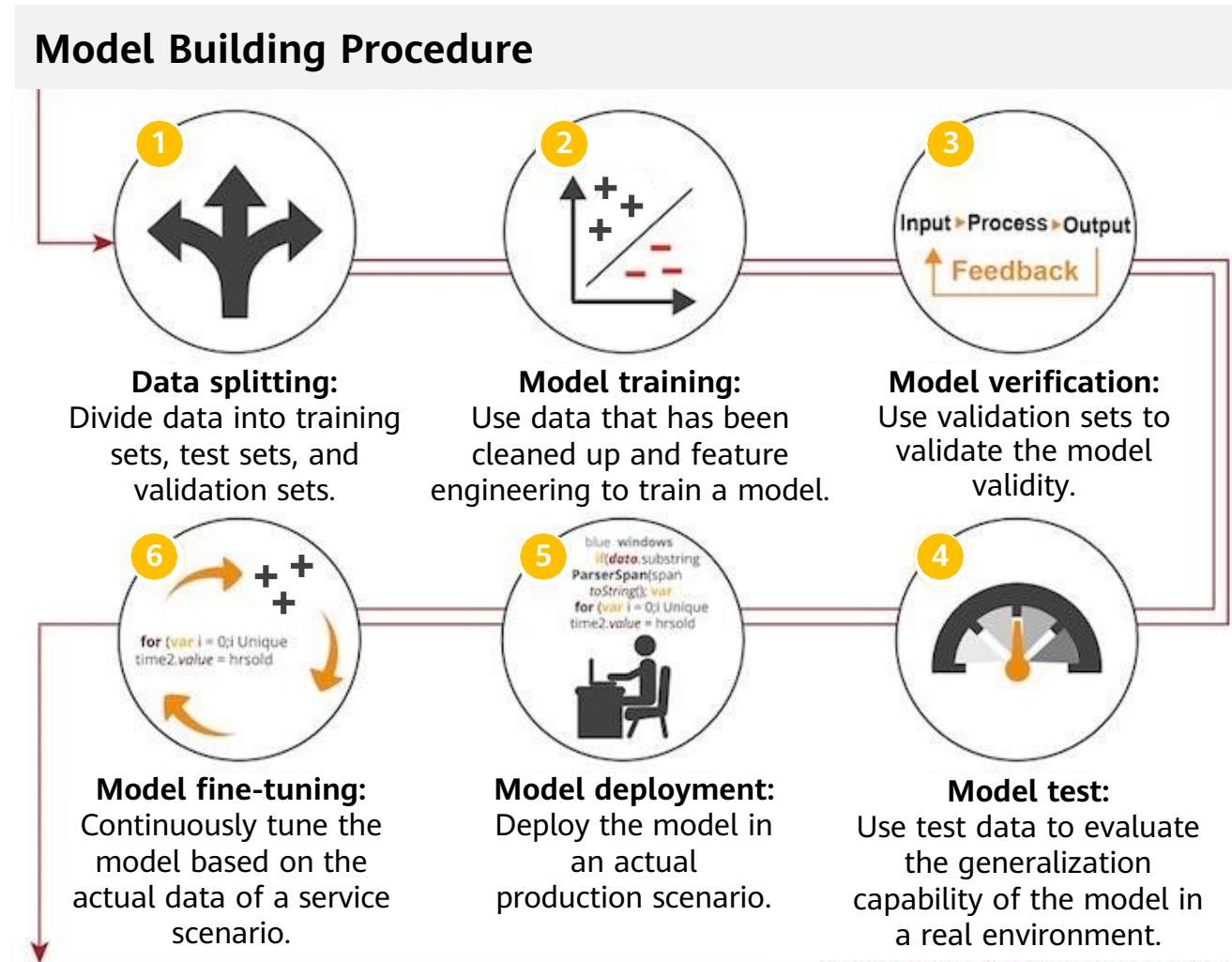


The most common type of embedded feature selection method is the **regularization method**. Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm that bias the model toward lower complexity and reduce the number of features.

## Common methods

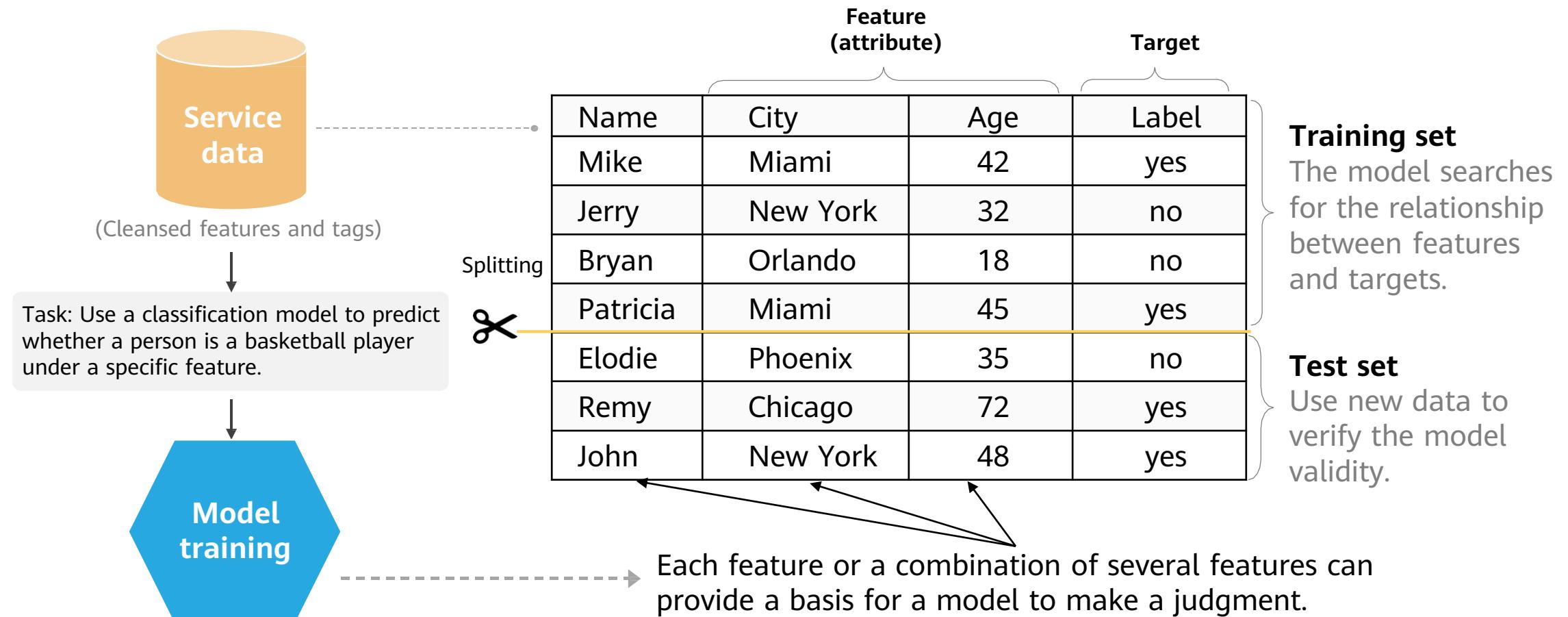
- Lasso regression
- Ridge regression

# Overall Procedure of Building a Model

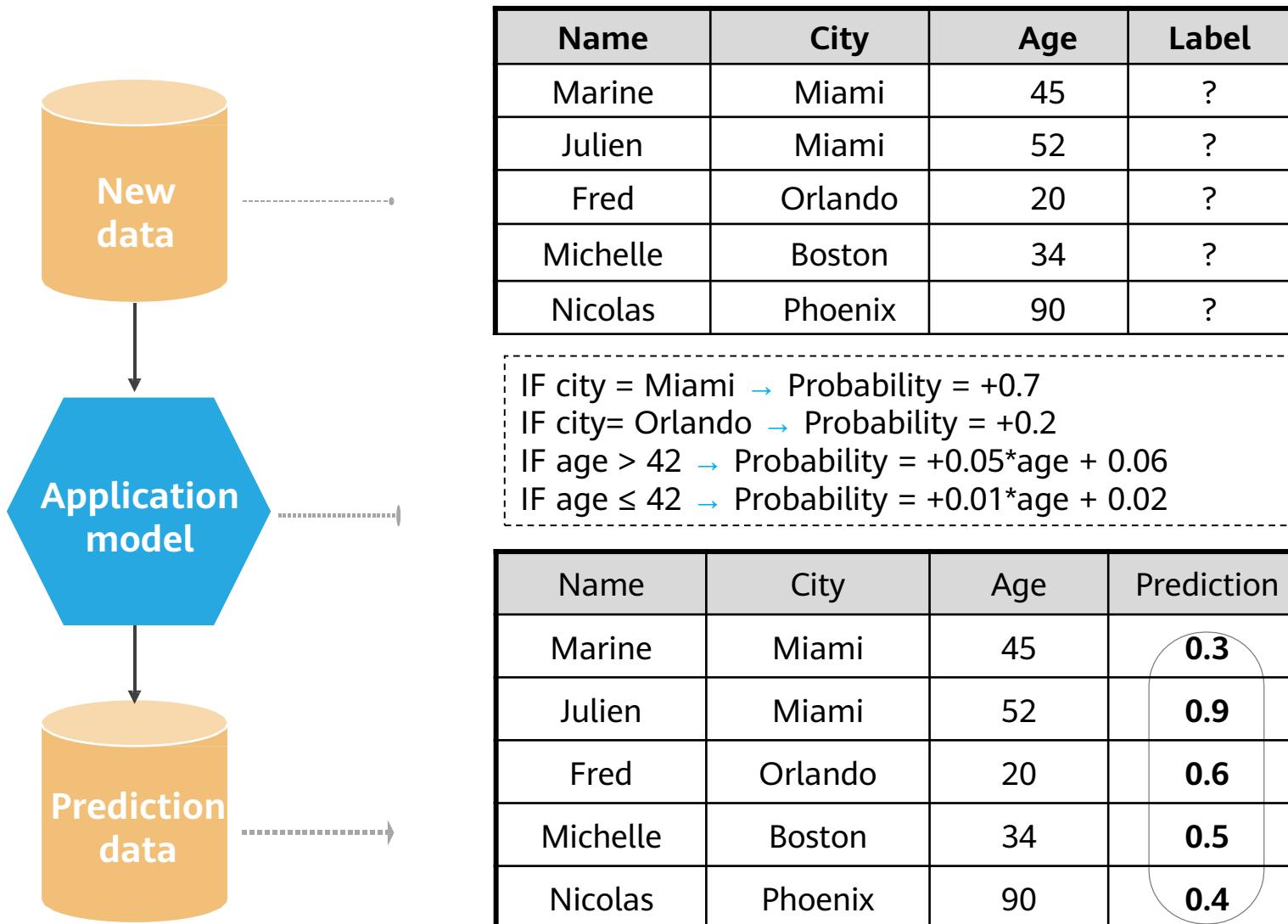


# Examples of Supervised Learning - Learning Phase

- Use the classification model to predict whether a person is a basketball player.



# Examples of Supervised Learning - Prediction Phase



## Unknown data

Recent data, it is not known whether the people are basketball players.

## Possibility prediction

Apply the model to the new data to predict whether the customer will change the supplier.

# What Is a Good Model?



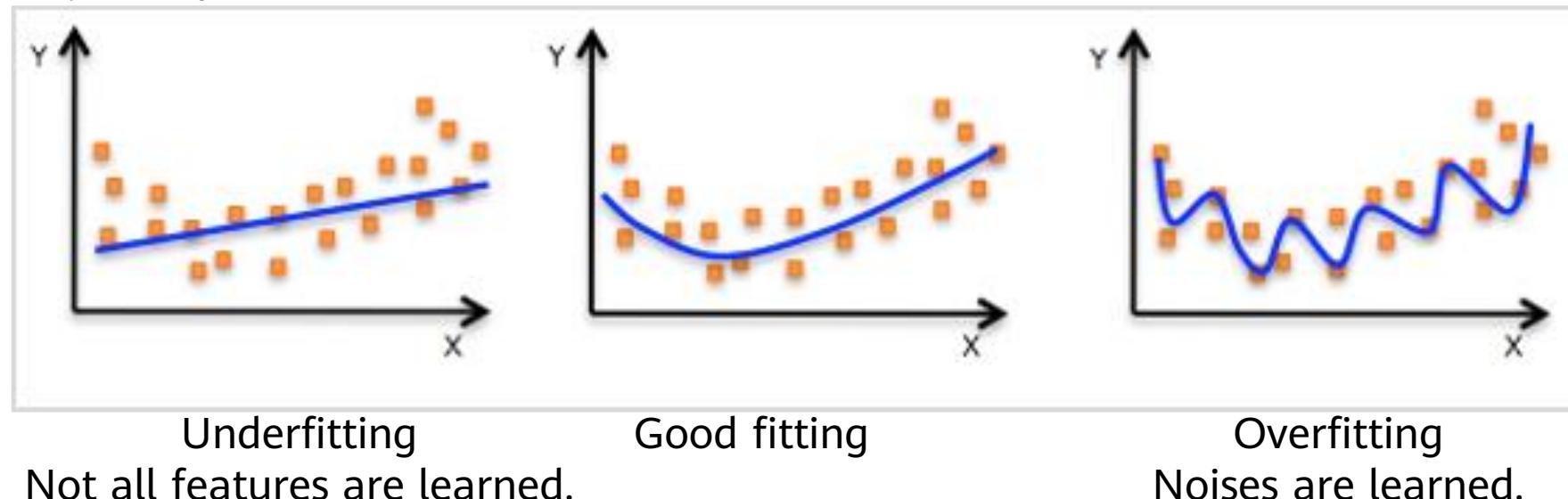
- **Generalization capability**  
Can it accurately predict the actual service data?
- **Interpretability**  
Is the prediction result easy to interpret?
- **Prediction speed**  
How long does it take to predict each piece of data?
- **Practicability**  
Is the prediction rate still acceptable when the service volume increases with a huge data volume?

# Model Validity (1)

- Generalization capability: The goal of machine learning is that the model obtained after learning should perform well on new samples, not just on samples used for training. The capability of applying a model to new samples is called generalization or robustness.
- Error: difference between the sample result predicted by the model obtained after learning and the actual sample result.
  - Training error: error that you get when you run the model on the training data.
  - Generalization error: error that you get when you run the model on new samples. Obviously, we prefer a model with a smaller generalization error.
- Underfitting: occurs when the model or the algorithm does not fit the data well enough.
- Overfitting: occurs when the training error of the model obtained after learning is small but the generalization error is large (poor generalization capability).

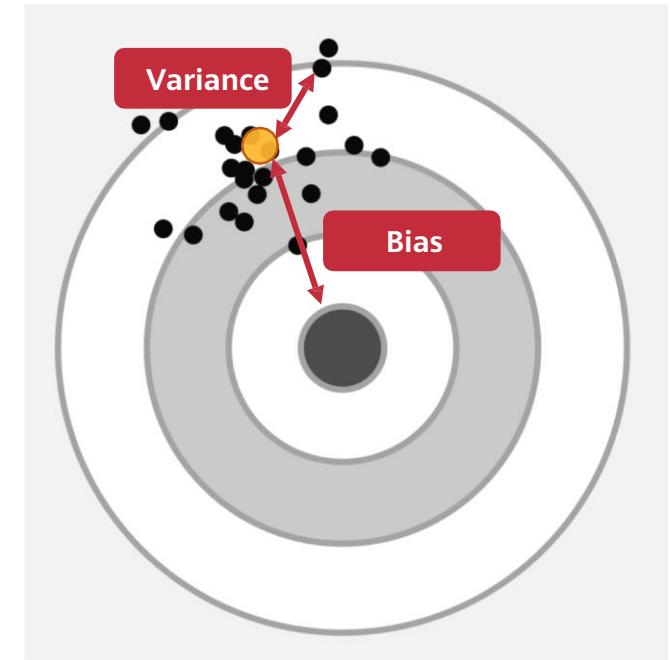
# Model Validity (2)

- Model capacity: model's capability of fitting functions, which is also called model complexity.
  - When the capacity suits the task complexity and the amount of training data provided, the algorithm effect is usually optimal.
  - Models with insufficient capacity cannot solve complex tasks and underfitting may occur.
  - A high-capacity model can solve complex tasks, but overfitting may occur if the capacity is higher than that required by a task.



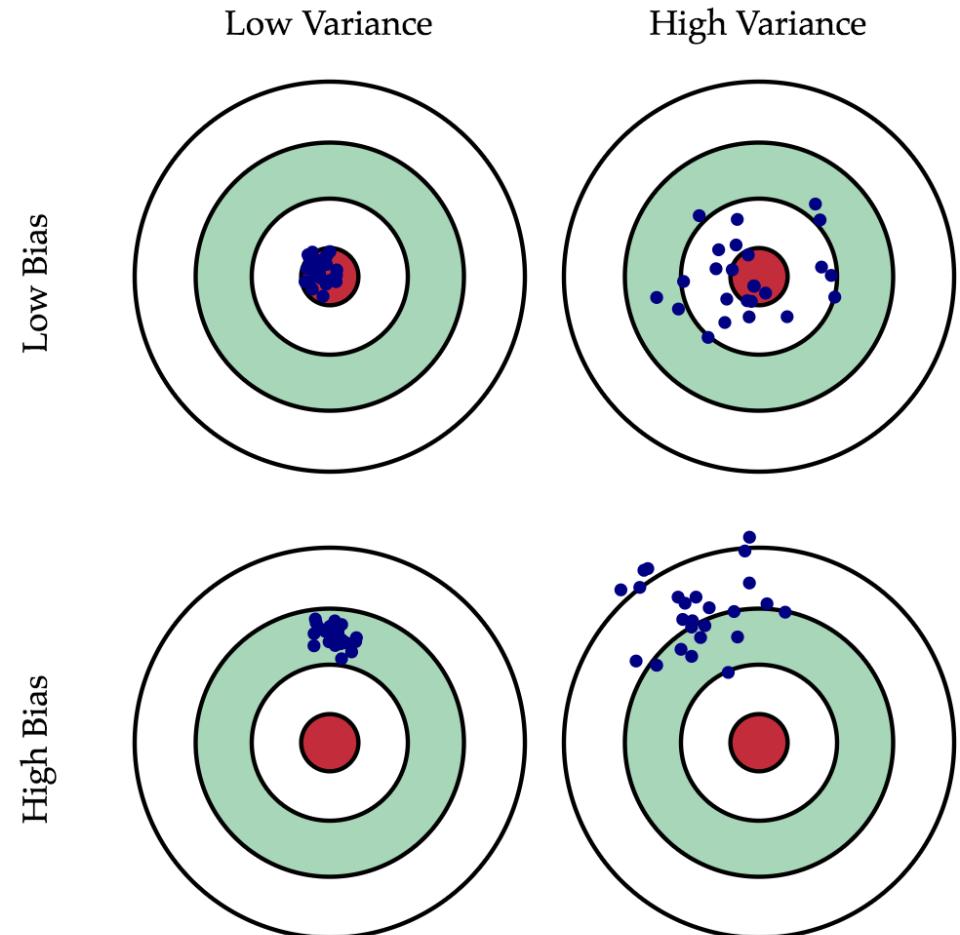
# Overfitting Cause — Error

- Total error of final prediction = Bias<sup>2</sup> + Variance + Irreducible error
- Generally, the prediction error can be divided into two types:
  - Error caused by "bias"
  - Error caused by "variance"
- Variance:
  - Offset of the prediction result from the average value
  - Error caused by the model's sensitivity to small fluctuations in the training set
- Bias:
  - Difference between the expected (or average) prediction value and the correct value we are trying to predict.



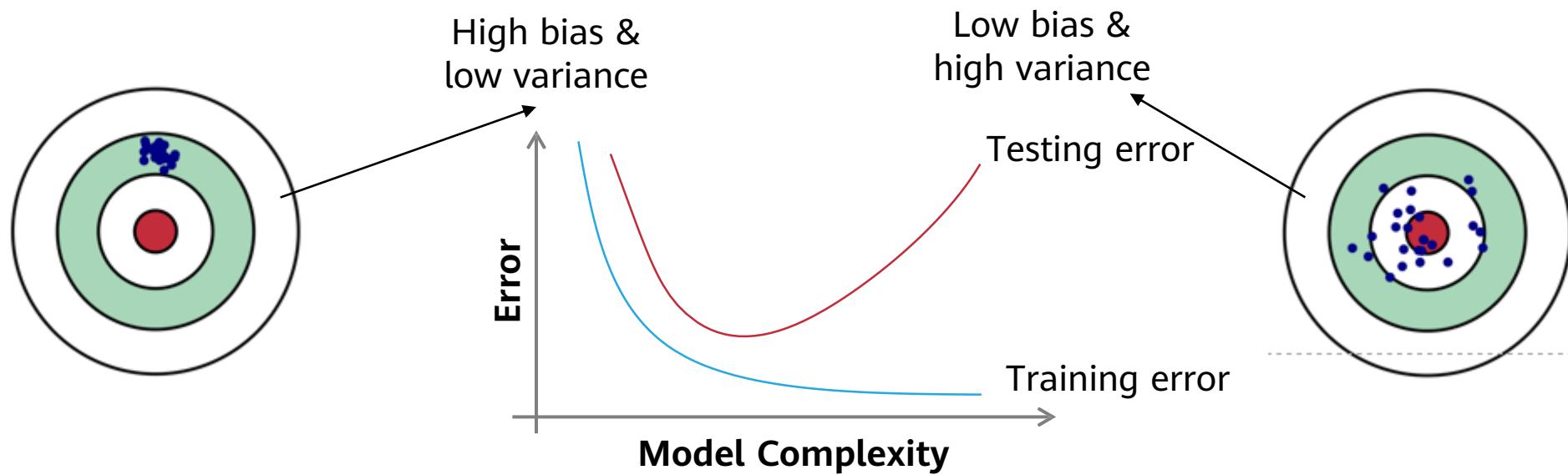
# Variance and Bias

- Combinations of variance and bias are as follows:
  - Low bias & low variance → Good model
  - Low bias & high variance
  - High bias & low variance
  - High bias & high variance → Poor model
- Ideally, we want a model that can accurately capture the rules in the training data and summarize the invisible data (new data). However, it is usually impossible for the model to complete both tasks at the same time.



# Model Complexity and Error

- As the model complexity increases, the training error decreases.
- As the model complexity increases, the test error decreases to a certain point and then increases in the reverse direction, forming a convex curve.



# Machine Learning Performance Evaluation - Regression

- The closer the Mean Absolute Error (MAE) is to 0, the better the model can fit the training data.

$$MAE = \frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

- Mean Square Error (MSE)

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- The value range of  $R^2$  is  $(-\infty, 1]$ . A larger value indicates that the model can better fit the training data. TSS indicates the difference between samples. RSS indicates the difference between the predicted value and sample value.

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y}_i)^2}$$

# Machine Learning Performance Evaluation - Classification (1)

- Terms and definitions:
  - $P$ : positive, indicating the number of real positive cases in the data.
  - $N$ : negative, indicating the number of real negative cases in the data.
  - $TP$ : true positive, indicating the number of positive cases that are correctly classified by the classifier.
  - $TN$ : true negative, indicating the number of negative cases that are correctly classified by the classifier.
  - $FP$ : false positive, indicating the number of positive cases that are incorrectly classified by the classifier.
  - $FN$ : false negative, indicating the number of negative cases that are incorrectly classified by the classifier.
- Confusion matrix: at least an  $m \times m$  table.  $CM_{i,j}$  of the first  $m$  rows and  $m$  columns indicates the number of cases that actually belong to class  $i$  but are classified into class  $j$  by the classifier.
  - Ideally, for a high accuracy classifier, most prediction values should be located in the diagonal from  $CM_{1,1}$  to  $CM_{m,m}$  of the table while values outside the diagonal are 0 or close to 0. That is,  $FP$  and  $FN$  are close to 0.

		Estimated amount	yes	no	Total
		Actual amount			
		yes	$TP$	$FN$	$P$
		no	$FP$	$TN$	$N$
		Total	$P'$	$N'$	$P + N$

Confusion matrix

# Machine Learning Performance Evaluation - Classification (2)

Measurement	Ratio
Accuracy and recognition rate	$\frac{TP + TN}{P + N}$
Error rate and misclassification rate	$\frac{FP + FN}{P + N}$
Sensitivity, true positive rate, and recall	$\frac{TP}{P}$
Specificity and true negative rate	$\frac{TN}{N}$
Precision	$\frac{TP}{TP + FP}$
$F_1$ , harmonic mean of the recall rate and precision	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
$F_\beta$ , where $\beta$ is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

# Example of Machine Learning Performance Evaluation

- We have trained a machine learning model to identify whether the object in an image is a cat. Now we use 200 pictures to verify the model performance. Among the 200 images, objects in 170 images are cats, while others are not. The identification result of the model is that objects in 160 images are cats, while others are not.

$$\text{Precision: } P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5\%$$

$$\text{Recall: } R = \frac{TP}{P} = \frac{140}{170} = 82.4\%$$

$$\text{Accuracy: } ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 75\%$$

		Estimated amount		Total
		yes	no	
Actual amount	yes	140	30	170
	no	20	10	30
Total	160	40	200	

# Contents

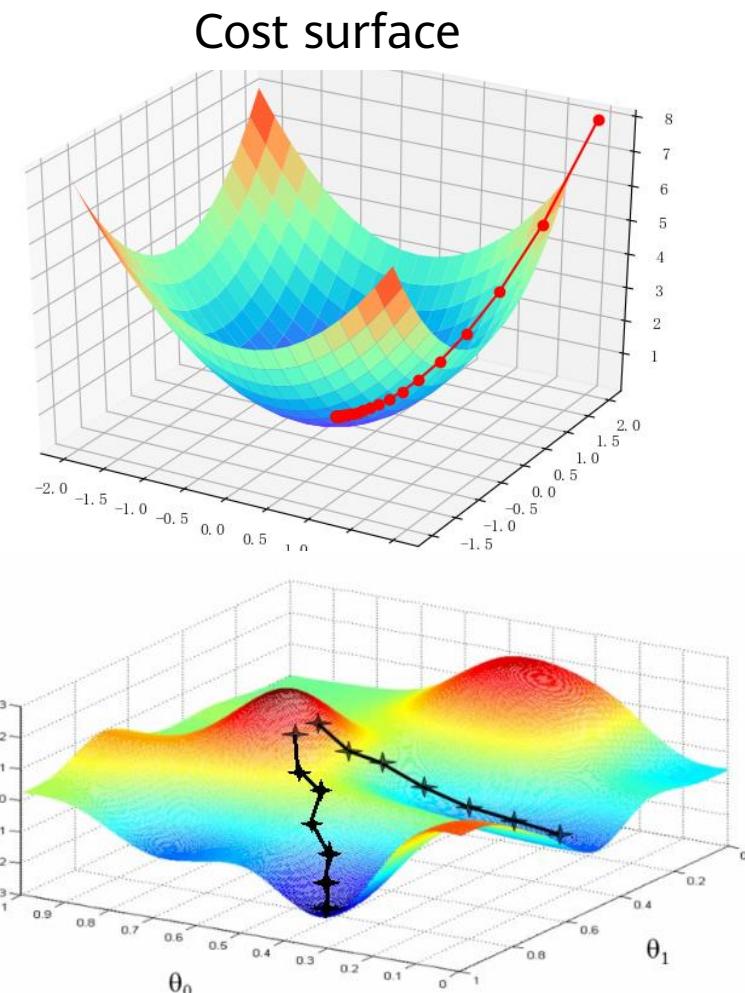
1. Machine Learning Definition
2. Machine Learning Types
3. Machine Learning Process
- 4. Other Key Machine Learning Methods**
5. Common Machine Learning Algorithms
6. Case study

# Machine Learning Training Method - Gradient Descent (1)

- The gradient descent method uses the negative gradient direction of the current position as the search direction, which is the steepest direction. The formula is as follows:

$$w_{k+1} = w_k - \eta \nabla f_{w_k}(x^i)$$

- In the formula,  $\eta$  indicates the learning rate and  $i$  indicates the data record number  $i$ . The weight parameter  $w$  indicates the change in each iteration.
- Convergence: The value of the objective function changes very little, or the maximum number of iterations is reached.



# Machine Learning Training Method - Gradient Descent (2)

- Batch Gradient Descent (BGD) uses the samples ( $m$  in total) in all datasets to update the weight parameter based on the gradient value at the current point.

$$w_{k+1} = w_k - \eta \frac{1}{m} \sum_{i=1}^m \nabla f_{w_k}(x^i)$$

- Stochastic Gradient Descent (SGD) randomly selects a sample in a dataset to update the weight parameter based on the gradient value at the current point.

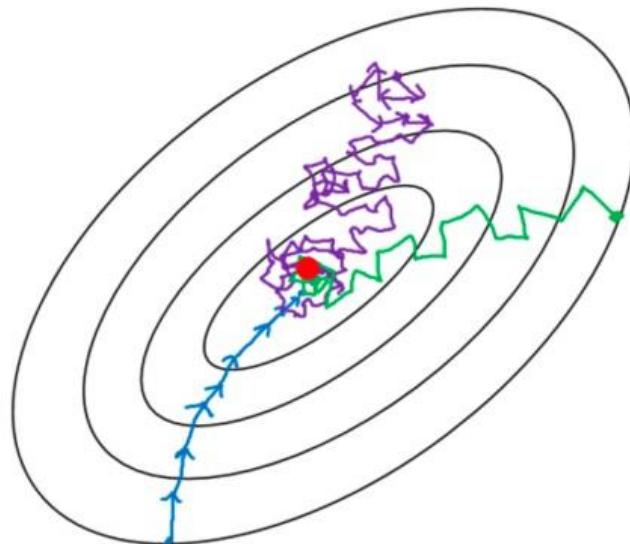
$$w_{k+1} = w_k - \eta \nabla f_{w_k}(x^i)$$

- Mini-Batch Gradient Descent (MBGD) combines the features of BGD and SGD and selects the gradients of  $n$  samples in a dataset to update the weight parameter.

$$w_{k+1} = w_k - \eta \frac{1}{n} \sum_{i=t}^{t+n-1} \nabla f_{w_k}(x^i)$$

# Machine Learning Training Method - Gradient Descent (3)

- Comparison of three gradient descent methods
  - In the SGD, samples selected for each training are stochastic. Such instability causes the loss function to be unstable or even causes reverse displacement when the loss function decreases to the lowest point.
  - BGD has the highest stability but consumes too many computing resources. MBGD is a method that balances SGD and BGD.



## BGD

Uses **all** training samples for training each time.

## SGD

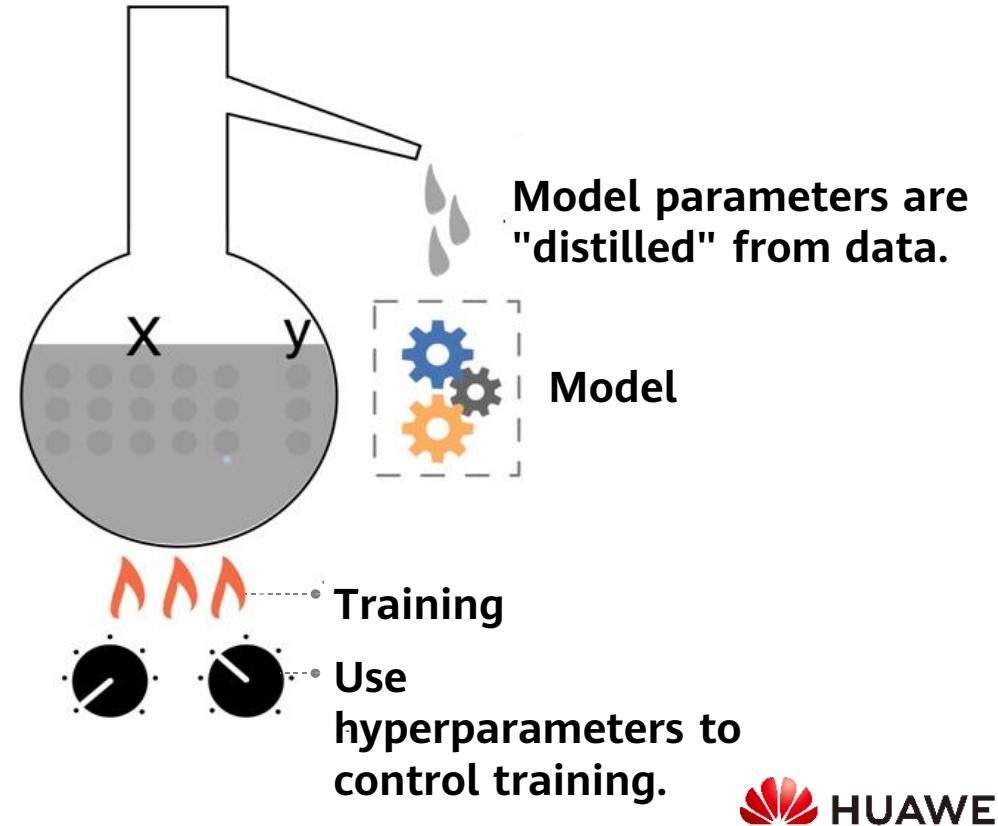
Uses **one** training sample for training each time.

## MBGD

Uses a certain number of training samples for training each time.

# Parameters and Hyperparameters in Models

- The model contains not only parameters but also hyperparameters. The purpose is to enable the model to learn the optimal parameters.
  - Parameters are automatically learned by models.
  - Hyperparameters are manually set.



# Hyperparameters of a Model

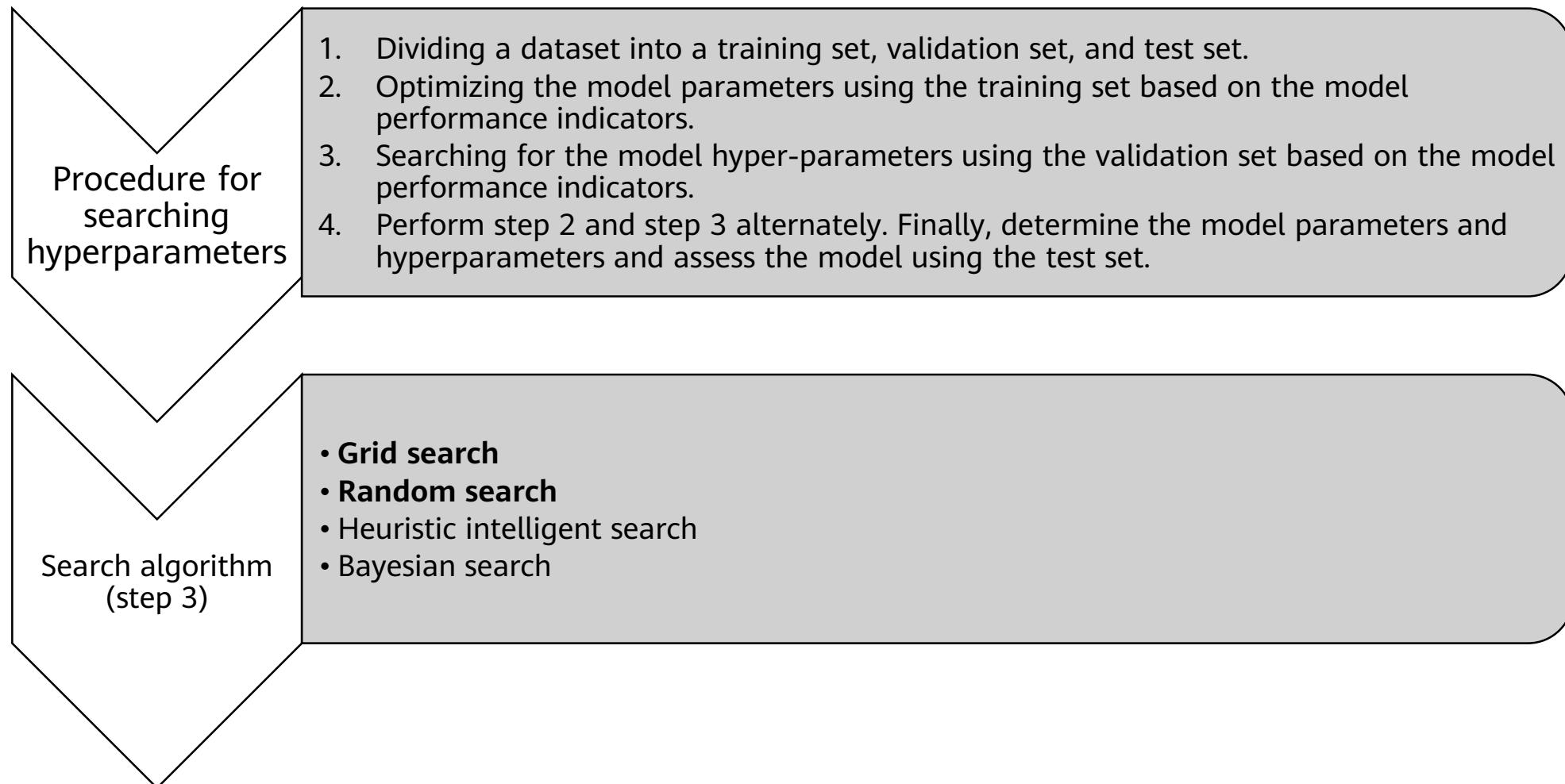
- Often used in model parameter estimation processes.
- Often specified by the practitioner.
- Can often be set using heuristics.
- Often tuned for a given predictive modeling problem.

Model hyperparameters are external configurations of models.

- $\lambda$  during Lasso/Ridge regression
- Learning rate for training a neural network, number of iterations, batch size, activation function, and number of neurons
- $C$  and  $\sigma$  in support vector machines (SVM)
- K in k-nearest neighbor (KNN)
- Number of trees in a random forest

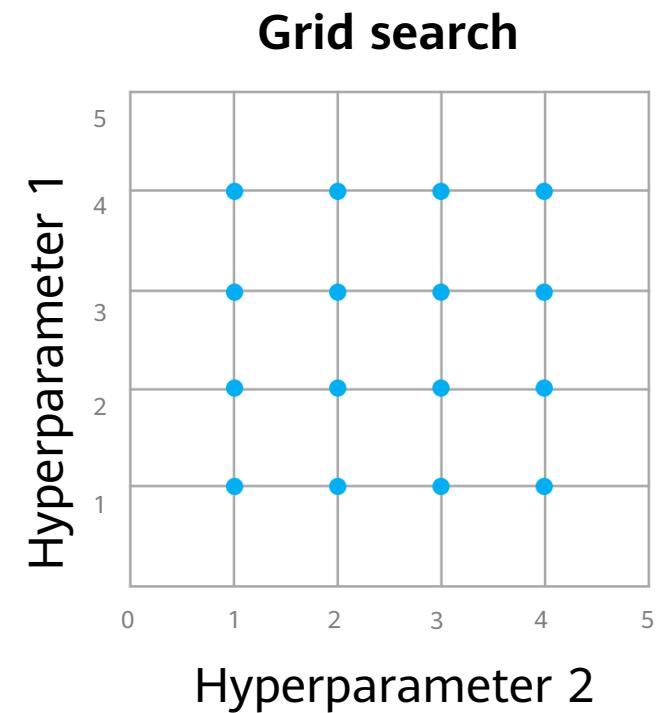
Common model hyperparameters

# Hyperparameter Search Procedure and Method



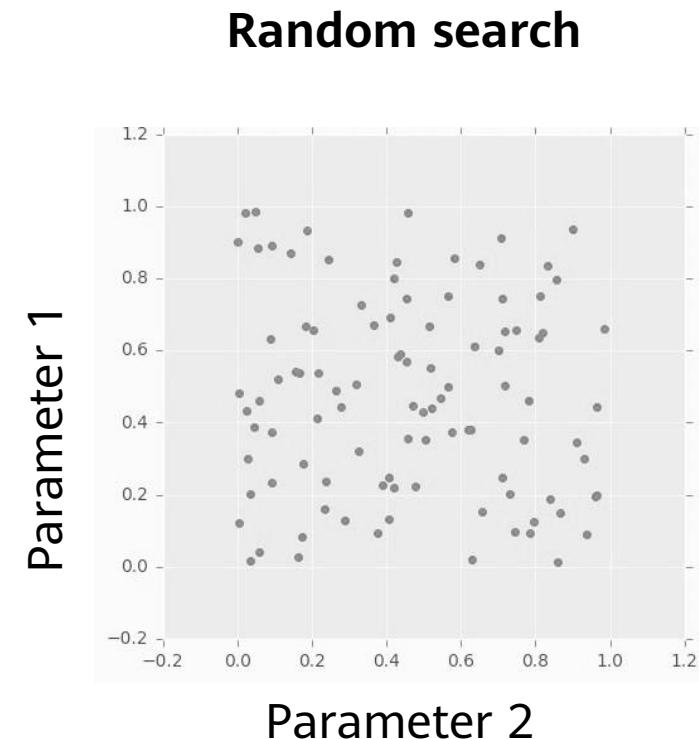
# Hyperparameter Searching Method - Grid Search

- Grid search attempts to **exhaustively search** all possible hyperparameter combinations to form a hyperparameter value grid.
- In practice, the range of hyperparameter values to search is specified manually.
- Grid search is an expensive and time-consuming method.
  - This method works well when the number of hyperparameters is relatively small. Therefore, it is applicable to generally machine learning algorithms but inapplicable to neural networks  
(see the deep learning part).



# Hyperparameter Searching Method - Random Search

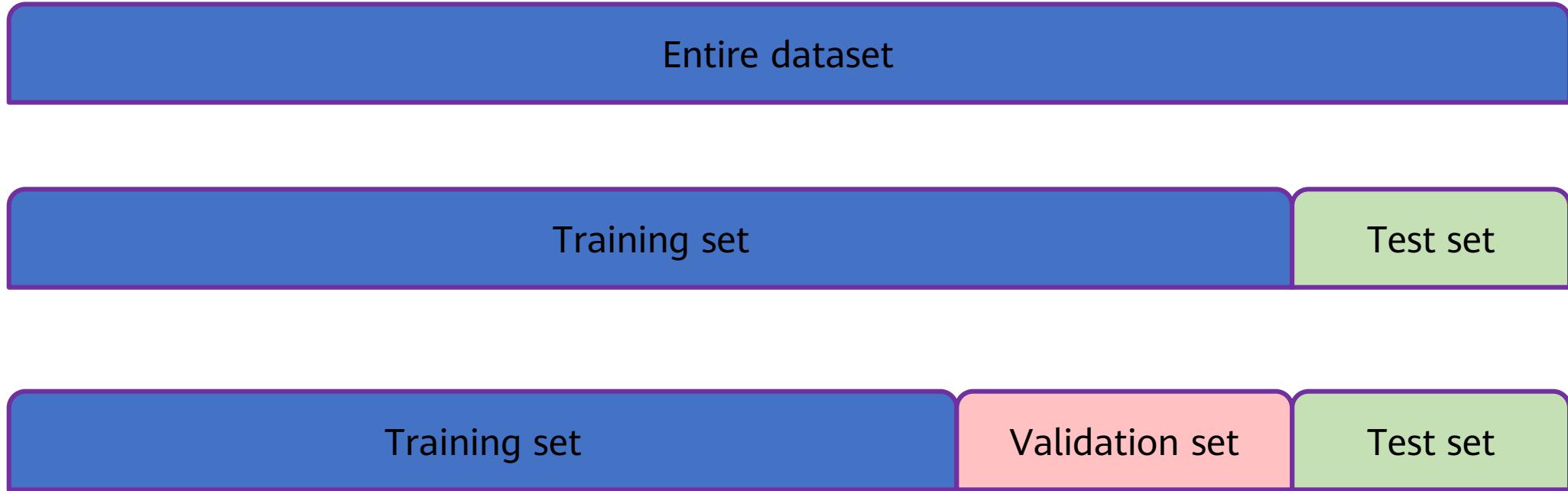
- When the hyperparameter search space is large, **random search** is better than grid search.
- In random search, each setting is sampled from the distribution of possible parameter values, in an attempt to find the best subset of hyperparameters.
- Note:
  - Search is performed within a coarse range, which then will be narrowed based on where the best result appears.
  - Some hyperparameters are more important than others, and the search deviation will be affected during random search.



# Cross Validation (1)

- **Cross validation:** It is a statistical analysis method used to validate the performance of a classifier. The basic idea is to divide the original dataset into two parts: training set and validation set. Train the classifier using the training set and test the model using the validation set to check the classifier performance.
- **k-fold cross validation ( $K - CV$ ):**
  - Divide the raw data into  $k$  groups (generally, evenly divided).
  - Use each subset as a validation set, and use the other  $k - 1$  subsets as the training set. A total of  $k$  models can be obtained.
  - Use the mean classification accuracy of the final validation sets of  $k$  models as the performance indicator of the  $K - CV$  classifier.

# Cross Validation (2)

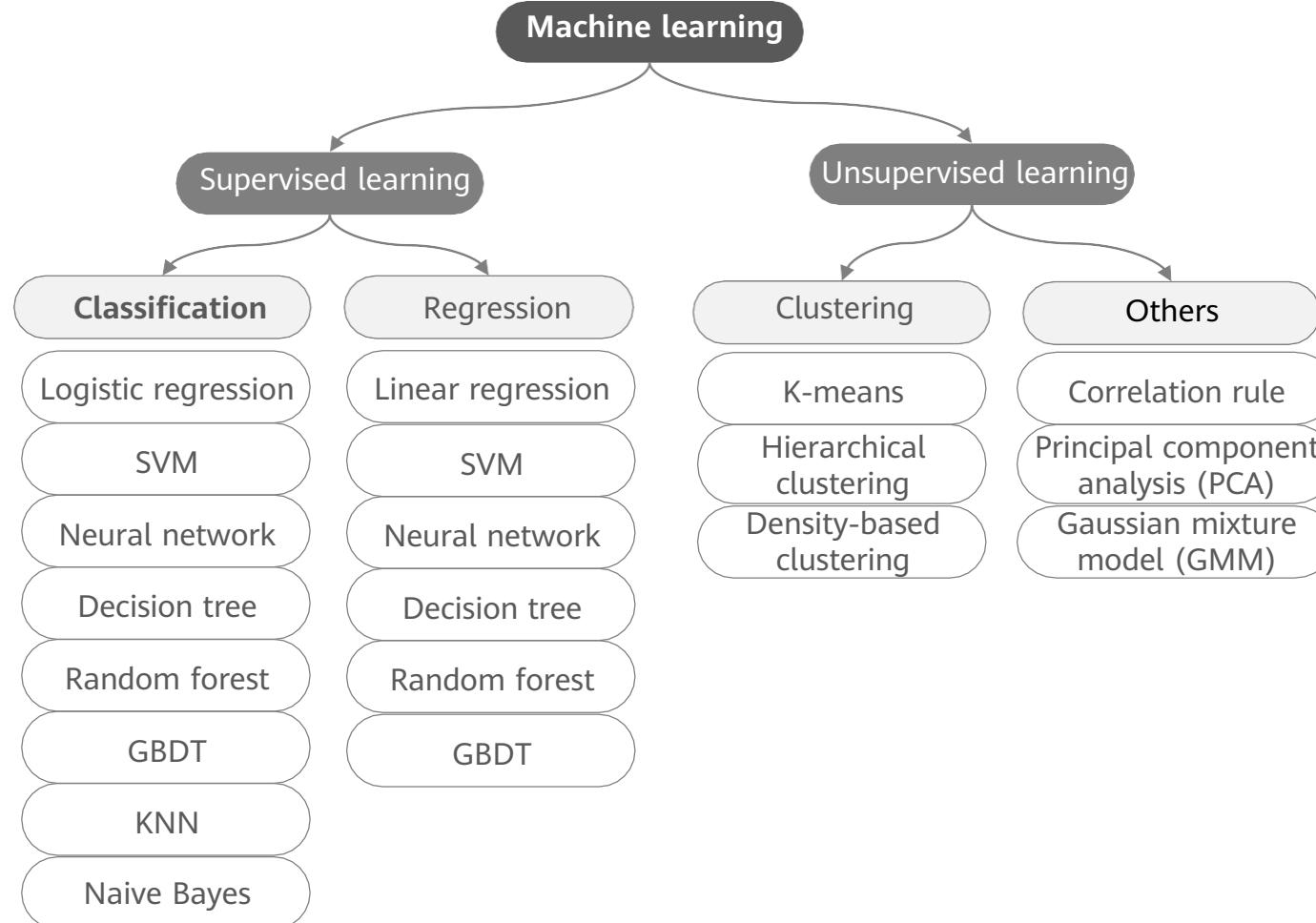


- Note: The K value in K-fold cross validation is also a hyperparameter.

# Contents

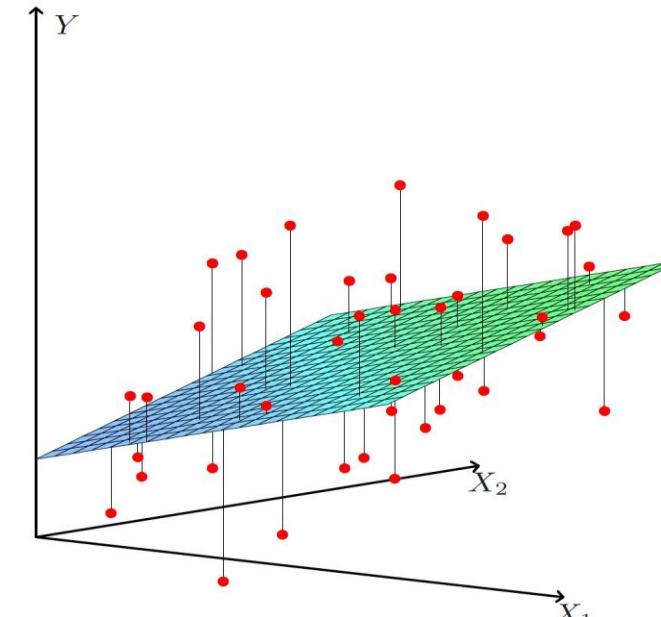
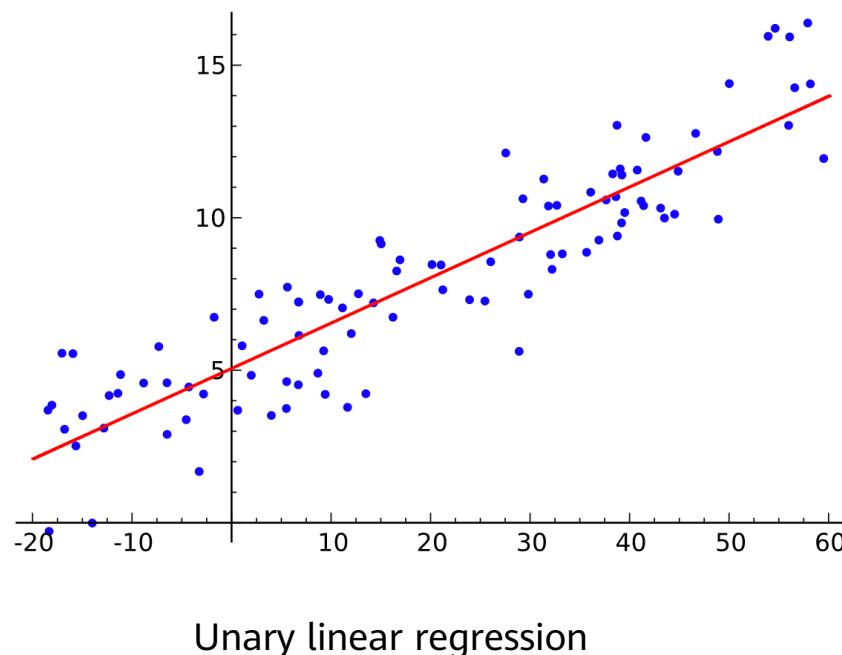
1. Machine Learning Definition
2. Machine Learning Types
3. Machine Learning Process
4. Other Key Machine Learning Methods
- 5. Common Machine Learning Algorithms**
6. Case study

# Machine Learning Algorithm Overview



# Linear Regression (1)

- Linear regression: a statistical analysis method to determine the quantitative relationships between two or more variables through regression analysis in mathematical statistics.
- Linear regression is a type of supervised learning.



# Linear Regression (2)

- The model function of linear regression is as follows, where  $w$  indicates the weight parameter,  $b$  indicates the bias, and  $x$  indicates the sample attribute.

$$h_w(x) = w^T x + b$$

- The relationship between the value predicted by the model and actual value is as follows, where  $y$  indicates the actual value, and  $\varepsilon$  indicates the error.

$$y = w^T x + b + \varepsilon$$

- The error  $\varepsilon$  is influenced by many factors independently. According to the central limit theorem, the error  $\varepsilon$  follows normal distribution. According to the normal distribution function and maximum likelihood estimation, the loss function of linear regression is as follows:

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2$$

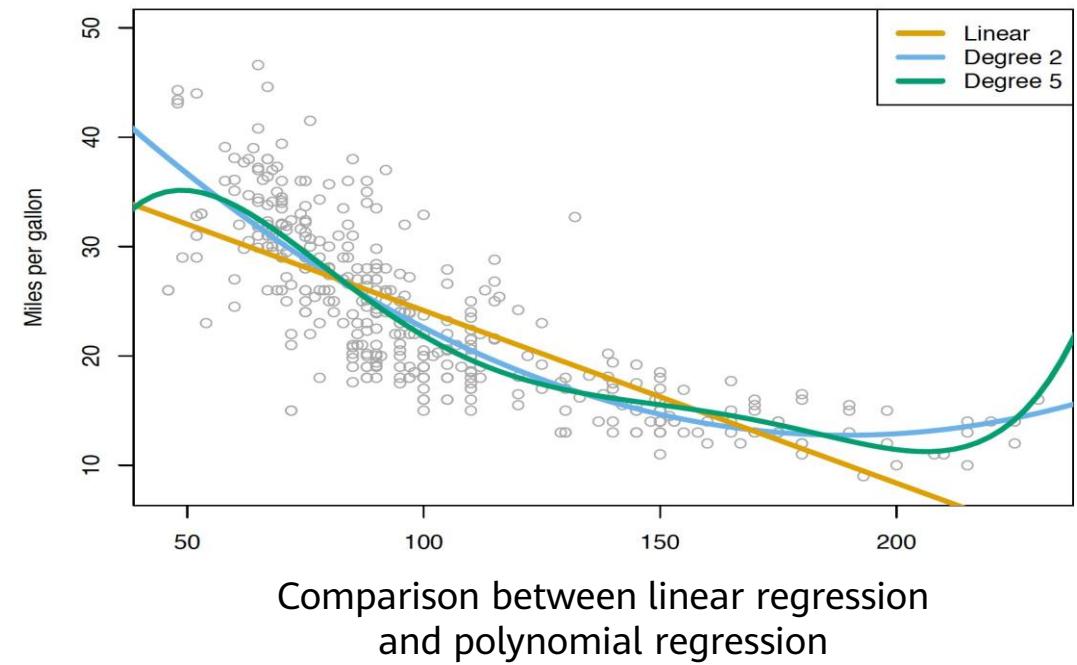
- To make the predicted value close to the actual value, we need to minimize the loss value. We can use the gradient descent method to calculate the weight parameter  $w$  when the loss function reaches the minimum, and then complete model building.

# Linear Regression Extension - Polynomial Regression

- Polynomial regression is an extension of linear regression. Generally, the complexity of a dataset exceeds the possibility of fitting by a straight line. That is, obvious underfitting occurs if the original linear regression model is used. The solution is to use polynomial regression.

$$h_w(x) = w_1x + w_2x^2 + \dots + w_nx^n + b$$

- where, the nth power is a polynomial regression dimension (degree).
- Polynomial regression belongs to linear regression as the relationship between its weight parameters  $w$  is still linear while its nonlinearity is reflected in the feature dimension.



# Linear Regression and Overfitting Prevention

- Regularization terms can be used to reduce overfitting. The value of  $w$  cannot be too large or too small in the sample space. You can add a square sum loss on the target function.

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2 + \lambda \sum \|w\|_2^2$$

- Regularization terms (norm): The regularization term here is called L2-norm. Linear regression that uses this loss function is also called Ridge regression.

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2 + \lambda \sum \|w\|_1$$

- Linear regression with absolute loss is called Lasso regression.

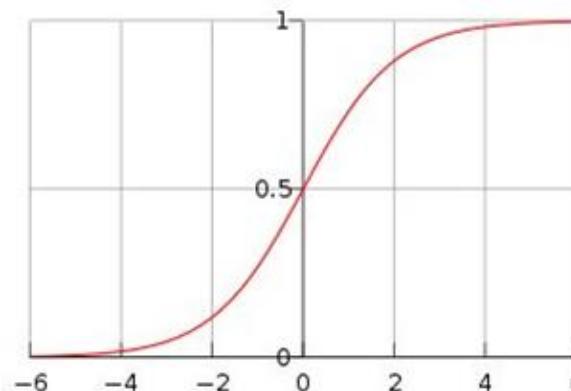
# Logistic Regression (1)

- Logistic regression: The logistic regression model is used to solve classification problems.  
The model is defined as follows:

$$P(Y = 1|x) = \frac{e^{wx+b}}{1 + e^{wx+b}}$$

$$P(Y = 0|x) = \frac{1}{1 + e^{wx+b}}$$

where  $w$  indicates the weight,  $b$  indicates the bias, and  $wx + b$  is regarded as the linear function of  $x$ .  
Compare the preceding two probability values. The class with a higher probability value is the class of  $x$ .



# Logistic Regression (2)

- Both the logistic regression model and linear regression model are generalized linear models. Logistic regression introduces nonlinear factors (the sigmoid function) based on linear regression and sets thresholds, so it can deal with binary classification problems.
- According to the model function of logistic regression, the loss function of logistic regression can be estimated as follows by using the maximum likelihood estimation:

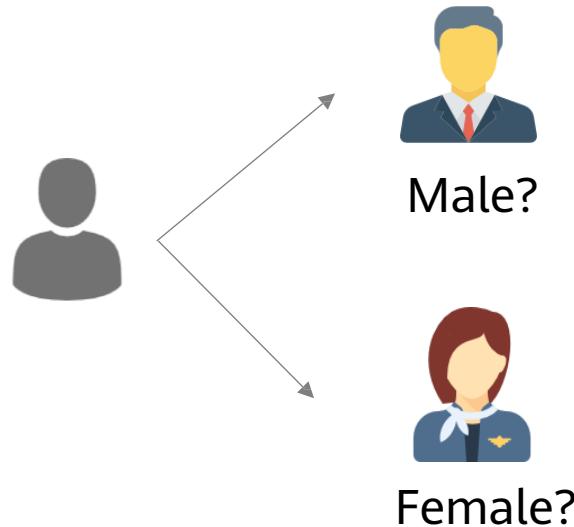
$$J(w) = -\frac{1}{m} \sum (y \ln h_w(x) + (1-y) \ln(1-h_w(x)))$$

- where  $w$  indicates the weight parameter,  $m$  indicates the number of samples,  $x$  indicates the sample, and  $y$  indicates the real value. The values of all the weight parameters  $w$  can also be obtained through the gradient descent algorithm.

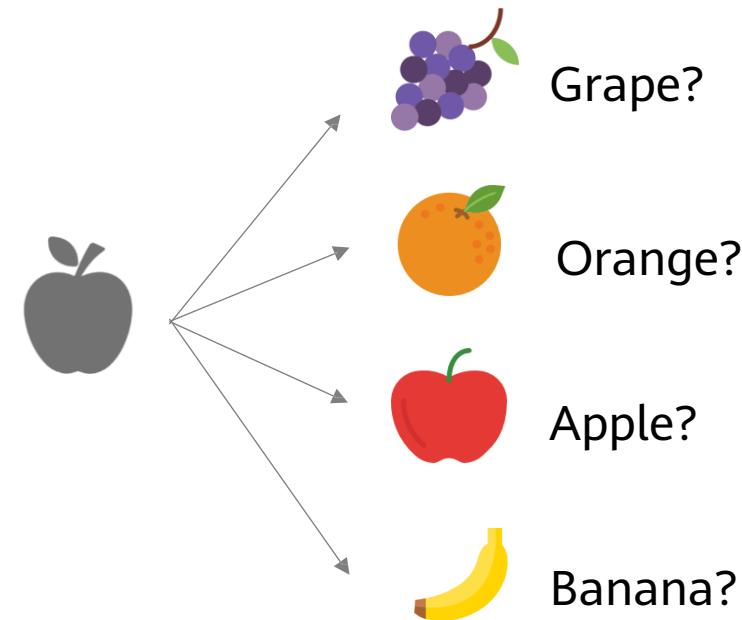
# Logistic Regression Extension - Softmax Function (1)

- Logistic regression applies only to binary classification problems. For multi-class classification problems, use the Softmax function.

**Binary classification problem**



**Multi-class classification problem**



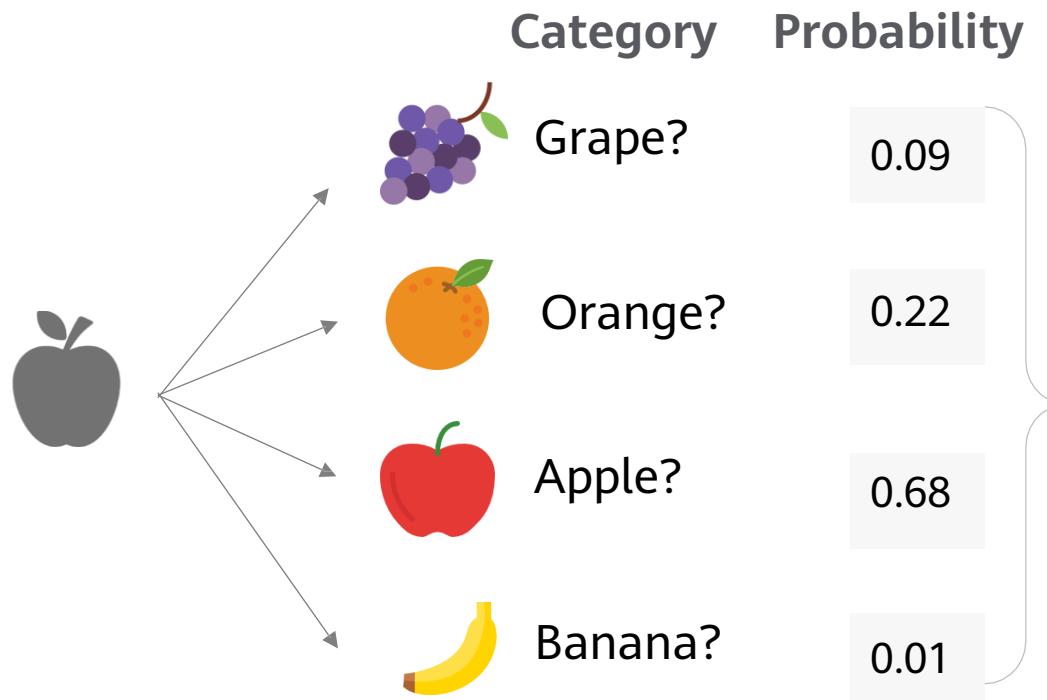
# Logistic Regression Extension - Softmax Function (2)

- Softmax regression is a generalization of logistic regression that we can use for K-class classification.
- The Softmax function is used to map a K-dimensional vector of arbitrary real values to another K-dimensional vector of real values, where each vector element is in the interval (0, 1).
- The regression probability function of Softmax is as follows:

$$p(y = k \mid x; w) = \frac{e^{w_k^T x}}{\sum_{l=1}^K e^{w_l^T x}}, k = 1, 2, \dots, K$$

# Logistic Regression Extension - Softmax Function (3)

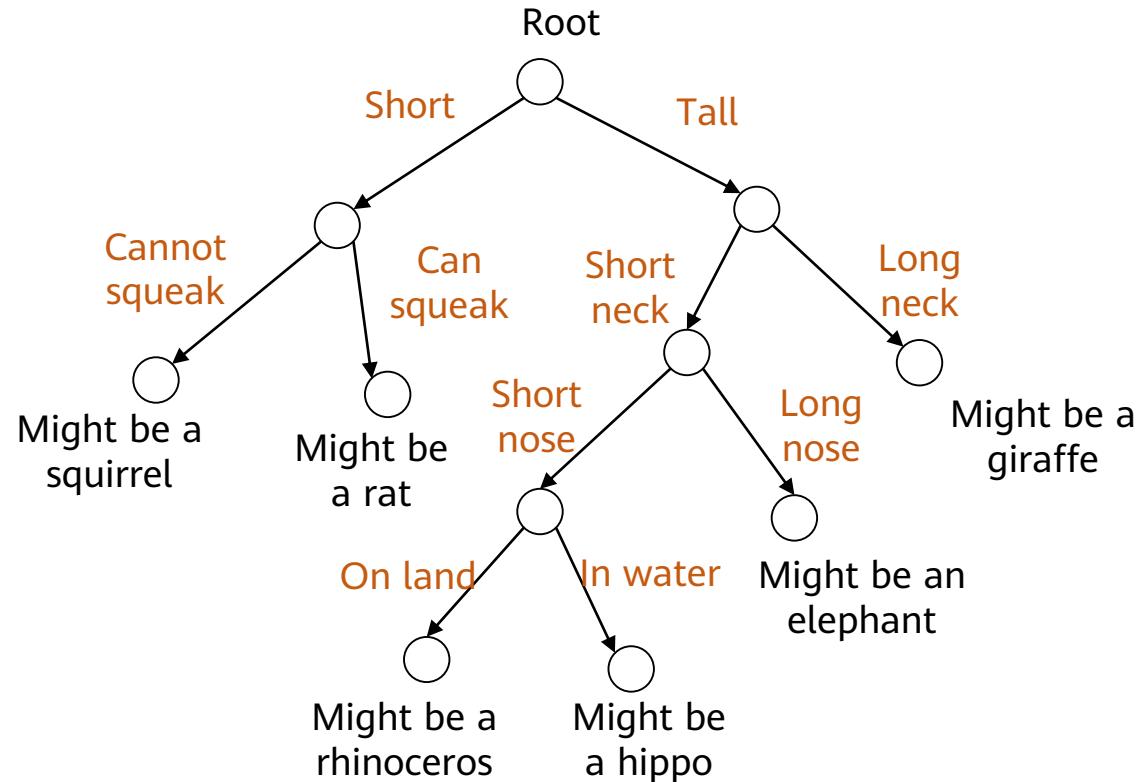
- Softmax assigns a probability to each class in a multi-class problem. These probabilities must add up to 1.
  - Softmax may produce a form belonging to a particular class. Example:



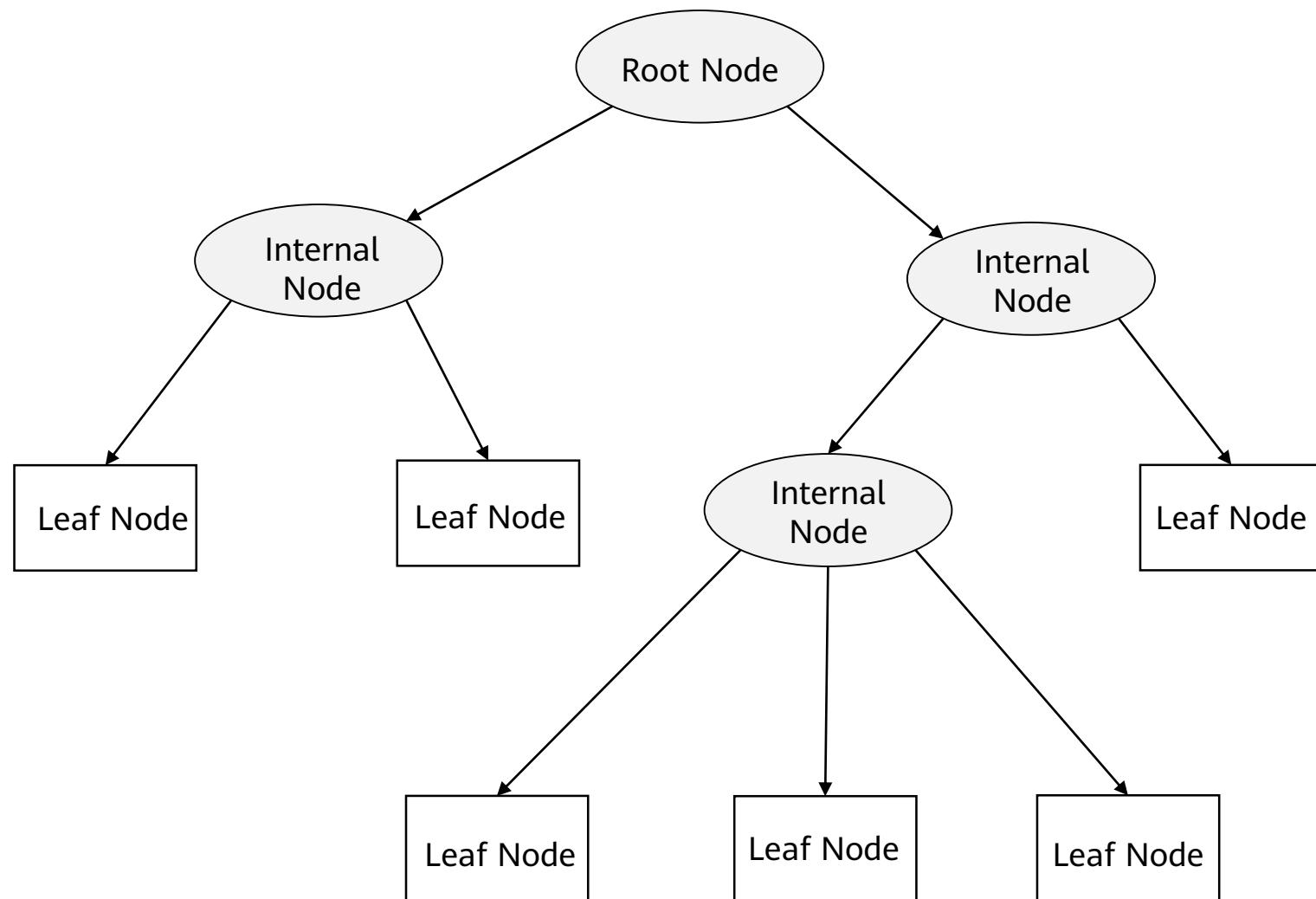
- **Sum of all probabilities:**
  - $0.09 + 0.22 + 0.68 + 0.01 = 1$
  - Most probably, this picture is an **apple**.

# Decision Tree

- A decision tree is a tree structure (a binary tree or a non-binary tree). Each non-leaf node represents a test on a feature attribute. Each branch represents the output of a feature attribute in a certain value range, and each leaf node stores a category. To use the decision tree, start from the root node, test the feature attributes of the items to be classified, select the output branches, and use the category stored on the leaf node as the final result.



# Decision Tree Structure



# Key Points of Decision Tree Construction

- To create a decision tree, we need to select attributes and determine the tree structure between feature attributes. The key step of constructing a decision tree is to divide data of all feature attributes, compare the result sets in terms of 'purity', and select the attribute with the highest 'purity' as the data point for dataset division.
- The metrics to quantify the 'purity' include the information entropy and GINI Index. The formula is as follows:

$$H(X) = -\sum_{k=1}^K p_k \log_2(p_k) \quad Gini = 1 - \sum_{k=1}^K p_k^2$$

- where  $p_k$  indicates the probability that the sample belongs to class k (there are K classes in total). A greater difference between purity before segmentation and that after segmentation indicates a better decision tree.
- Common decision tree algorithms include ID3, C4.5, and CART.

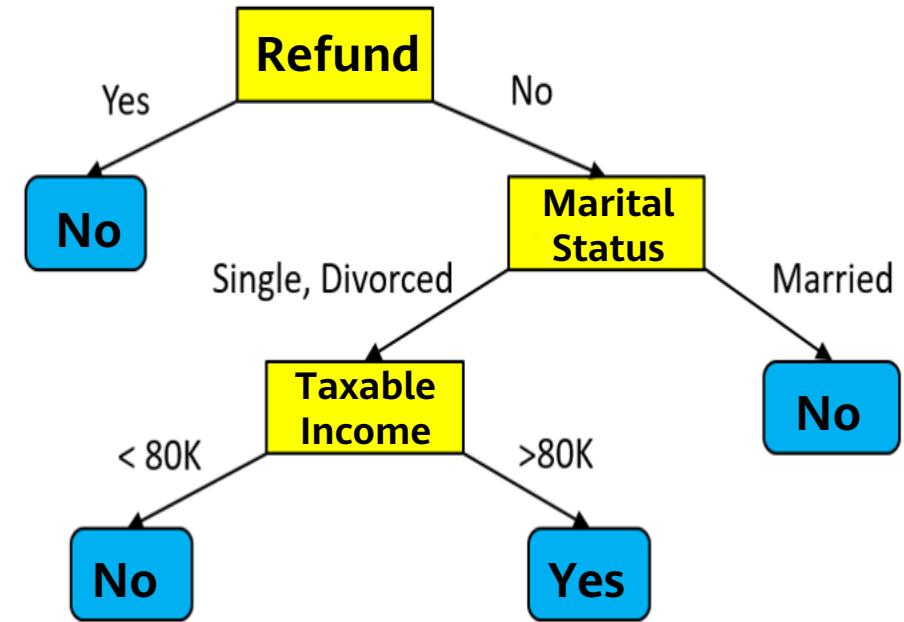
# Decision Tree Construction Process

- **Feature selection:** Select a feature from the features of the training data as the split standard of the current node. (Different standards generate different decision tree algorithms.)
- **Decision tree generation:** Generate internal node upside down based on the selected features and stop until the dataset can no longer be split.
- **Pruning:** The decision tree may easily become overfitting unless necessary pruning (including pre-pruning and post-pruning) is performed to reduce the tree size and optimize its node structure.

# Decision Tree Example

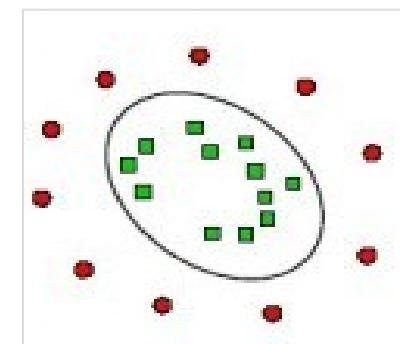
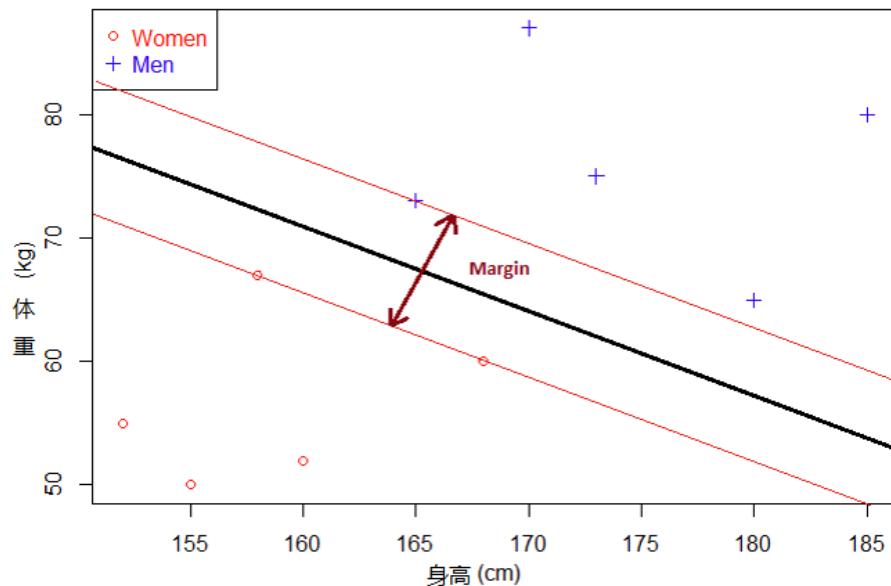
- The following figure shows a classification when a decision tree is used. The classification result is impacted by three attributes: Refund, Marital Status, and Taxable Income.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125,000	No
2	No	Married	100,000	No
3	No	Single	70,000	No
4	Yes	Married	120,000	No
5	No	Divorced	95,000	Yes
6	No	Married	60,000	No
7	Yes	Divorced	220,000	No
8	No	Single	85,000	Yes
9	No	Married	75,000	No
10	No	Single	90,000	Yes

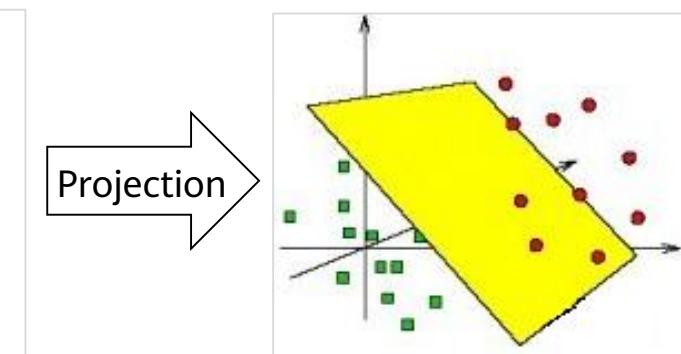


# SVM

- SVM is a binary classification model whose basic model is a linear classifier defined in the eigenspace with the largest interval. SVMs also include kernel tricks that make them nonlinear classifiers. The SVM learning algorithm is the optimal solution to convex quadratic programming.



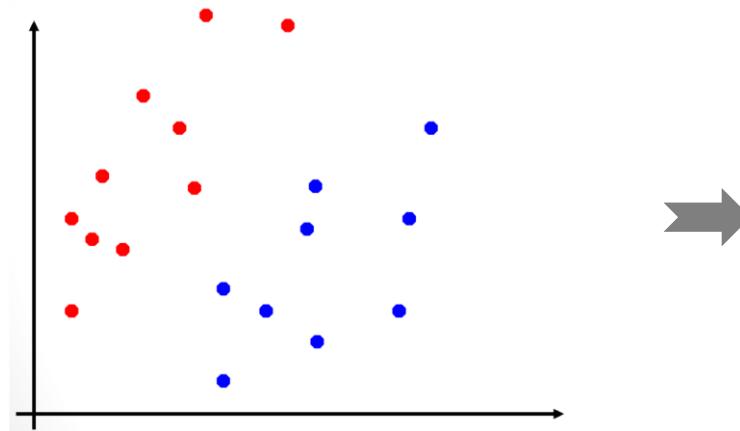
Complex  
segmentation in low-  
dimensional space



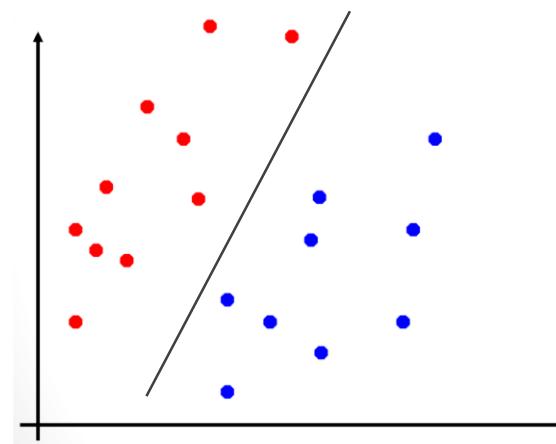
Easy segmentation in  
high-dimensional space

# Linear SVM (1)

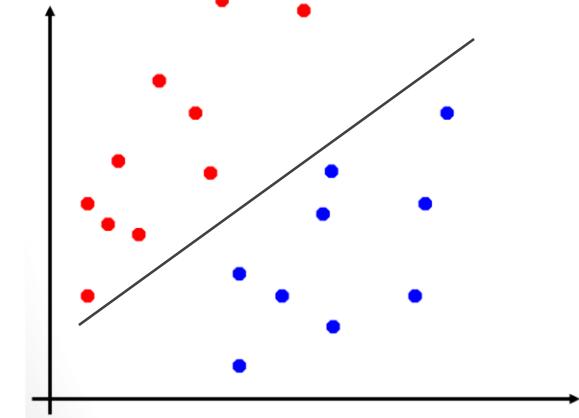
- How do we split the red and blue datasets by a straight line?



With binary classification  
Two-dimensional dataset



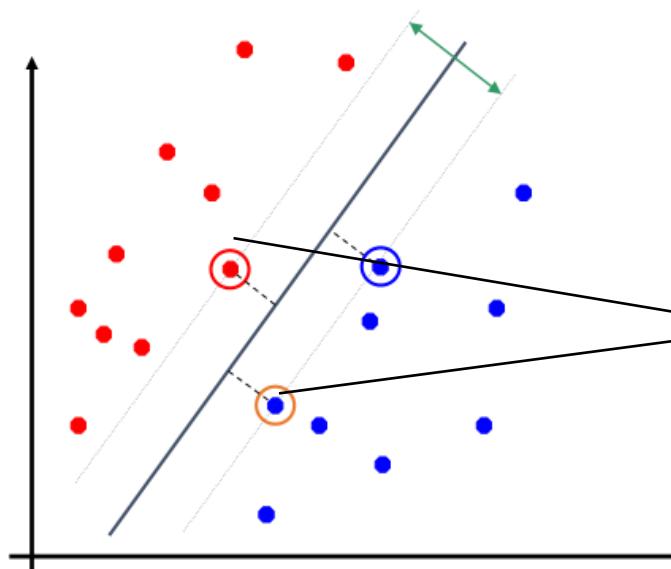
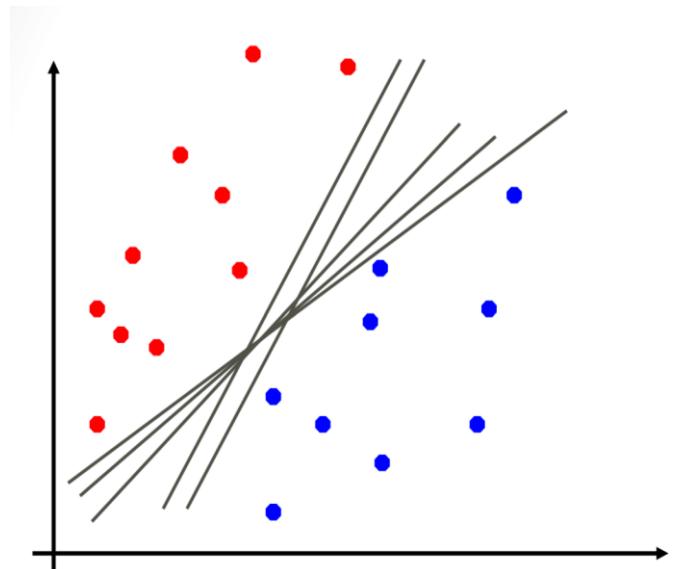
or



Both the left and right methods can be used to divide datasets. Which of them is correct?

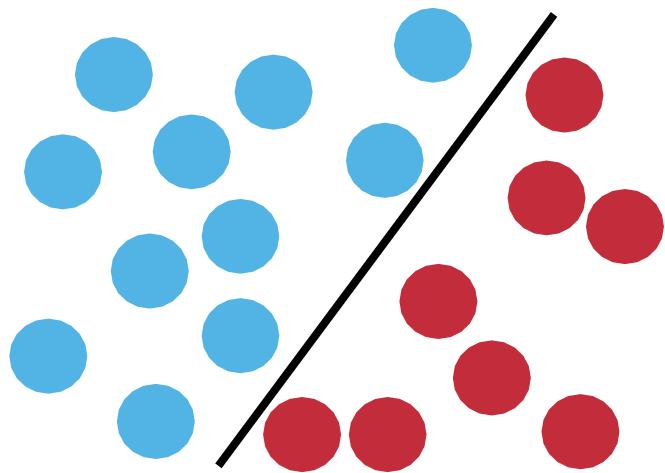
# Linear SVM (2)

- Straight lines are used to divide data into different classes. Actually, we can use multiple straight lines to divide data. The core idea of the SVM is to find a straight line and keep the point close to the straight line as **far** as possible from the straight line. This can enable strong generalization capability of the model. These points are called **support vectors**.
- In two-dimensional space, we use straight lines for segmentation. In high-dimensional space, we use **hyperplanes** for segmentation.

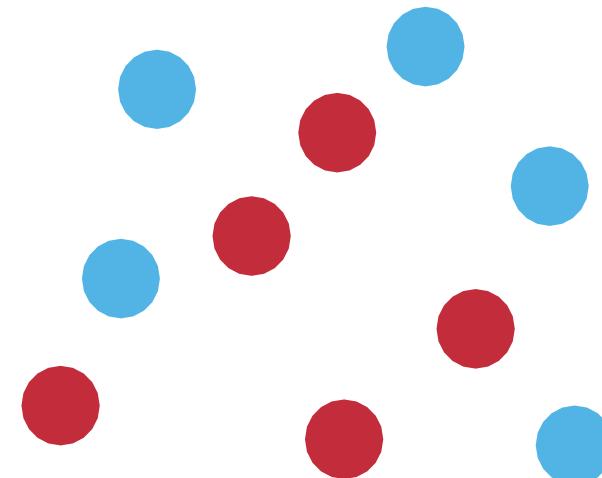


# Nonlinear SVM (1)

- How do we classify a nonlinear separable dataset?



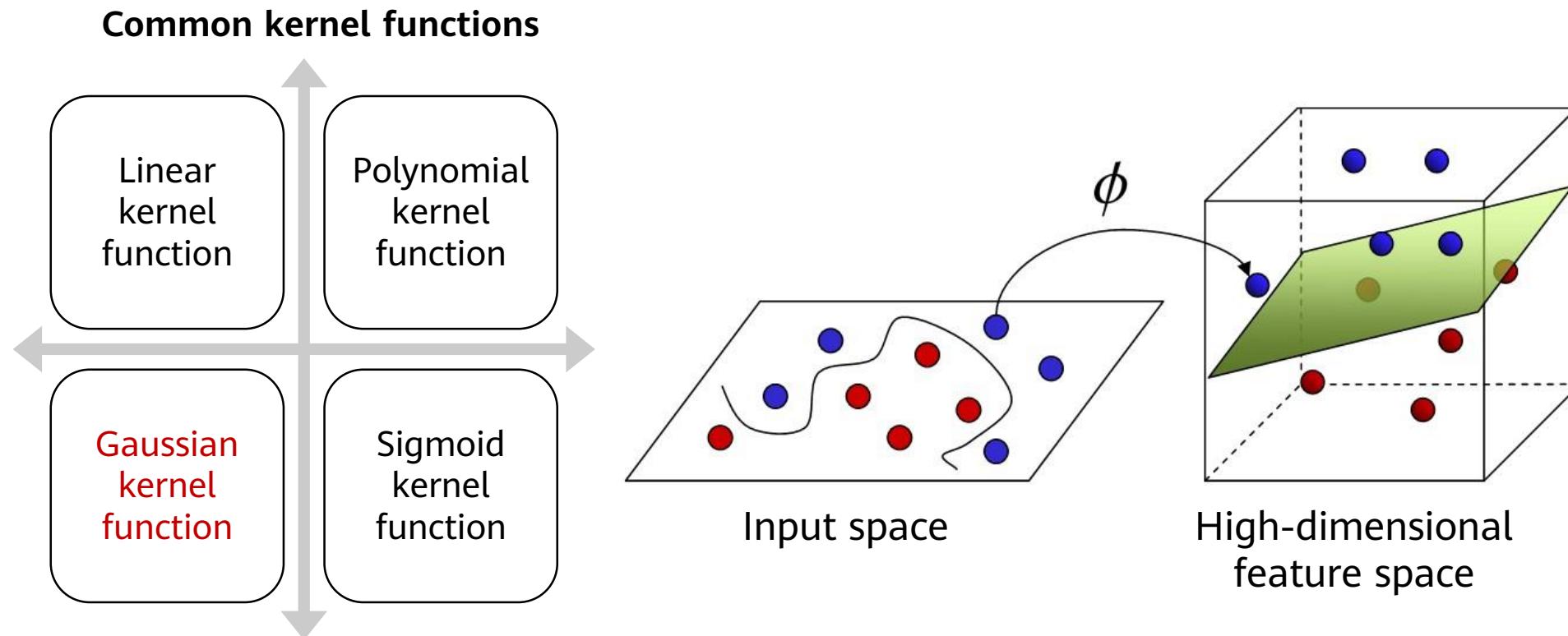
Linear SVM can function well  
for linear separable datasets.



Nonlinear datasets cannot be  
split with straight lines.

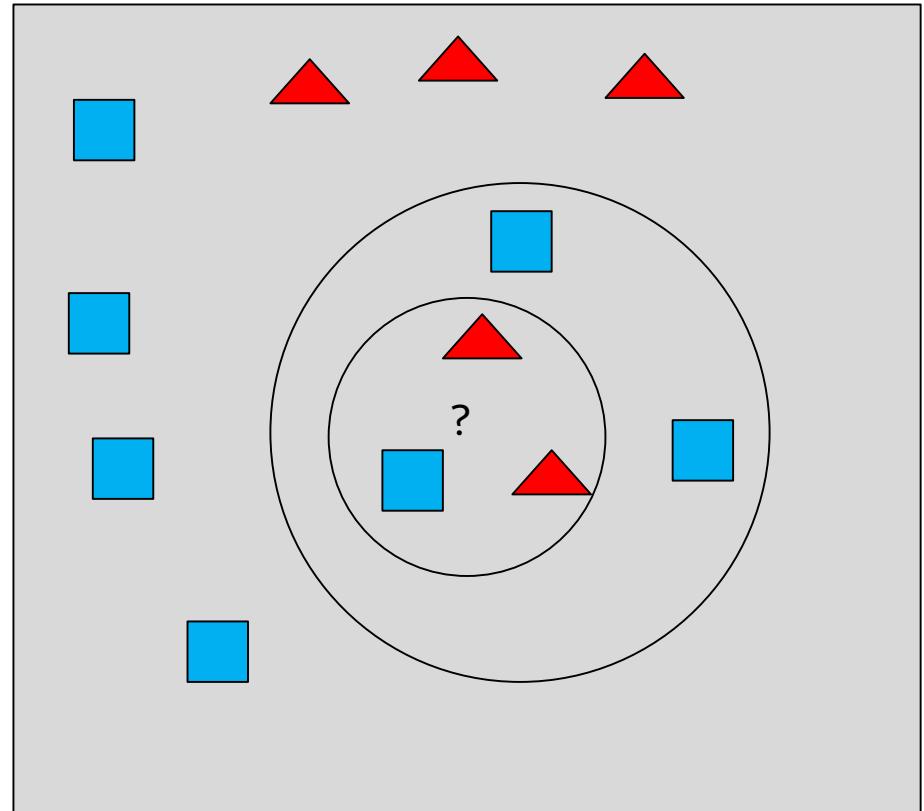
# Nonlinear SVM (2)

- Kernel functions are used to construct nonlinear SVMs.
- Kernel functions allow algorithms to fit the largest hyperplane in a transformed high-dimensional feature space.



# KNN Algorithm (1)

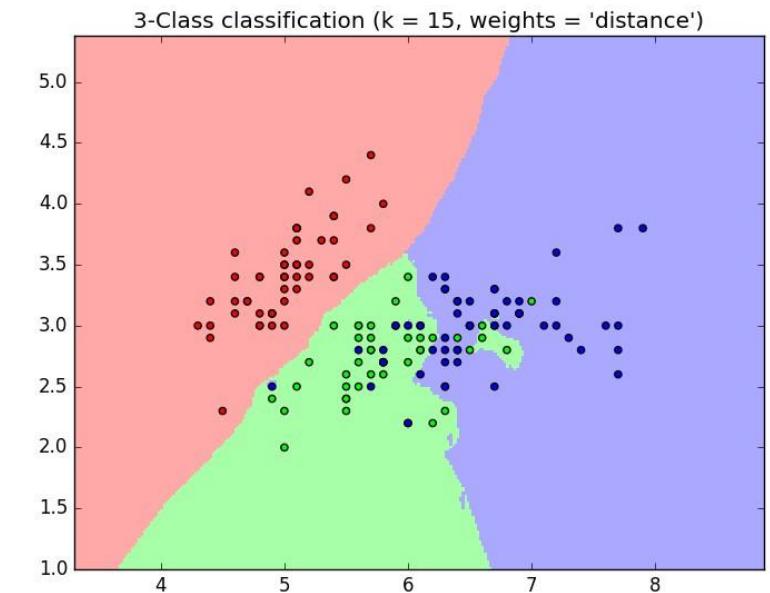
- The KNN classification algorithm is a theoretically mature method and one of the simplest machine learning algorithms. According to this method, if the majority of  $k$  samples most similar to one sample (nearest neighbors in the eigenspace) belong to a specific category, this sample also belongs to this category.



The target category of point ? varies with the number of the most adjacent nodes.

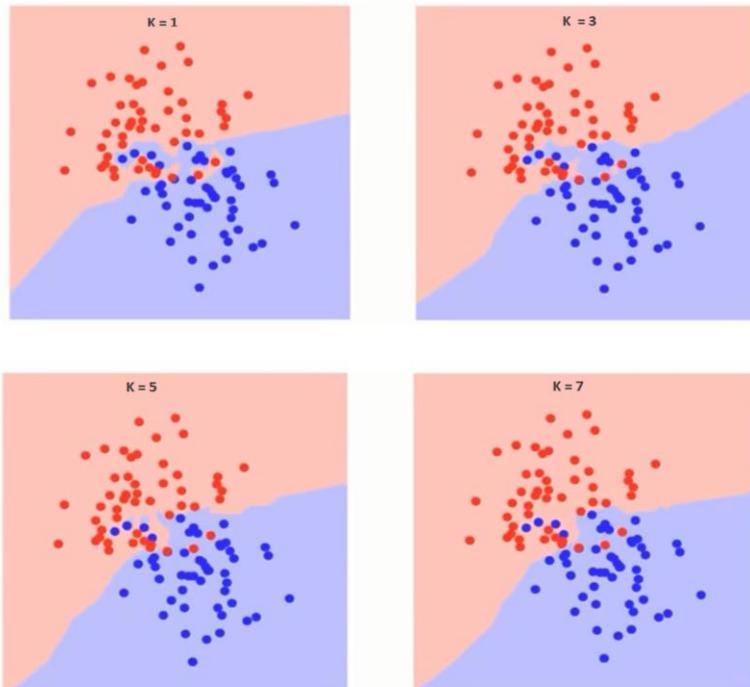
# KNN Algorithm (2)

- As the prediction result is determined based on the number and weights of neighbors in the training set, the KNN algorithm has a simple logic.
- KNN is a non-parametric method which is usually used in datasets with irregular decision boundaries.
  - The KNN algorithm generally adopts the majority voting method for classification prediction and the average value method for regression prediction.
- KNN requires a huge number of computations.



# KNN Algorithm (3)

- Generally, a larger k value reduces the impact of noise on classification, but obscures the boundary between classes.
  - A larger k value means a higher probability of underfitting because the segmentation is too rough. A smaller k value means a higher probability of overfitting because the segmentation is too refined.



- The boundary becomes smoother as the value of k increases.
- As the k value increases to infinity, all data points will eventually become all blue or all red.

# Naive Bayes (1)

- Naive Bayes algorithm: a simple multi-class classification algorithm based on the Bayes theorem. It assumes that features are independent of each other. For a given sample feature  $X$ , the probability that a sample belongs to a category  $H$  is:

$$P(C_k | X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n | C_k) P(C_k)}{P(X_1, \dots, X_n)}$$

- $X_1, \dots, X_n$  are data features, which are usually described by measurement values of  $m$  attribute sets.
  - For example, the color feature may have three attributes: red, yellow, and blue.
- $C_k$  indicates that the data belongs to a specific category  $C$
- $P(C_k | X_1, \dots, X_n)$  is a posterior probability, or a posterior probability of under condition  $C_k$ .
- $P(C_k)$  is a prior probability that is independent of  $X_1, \dots, X_n$
- $P(X_1, \dots, X_n)$  is the priori probability of  $X$ .

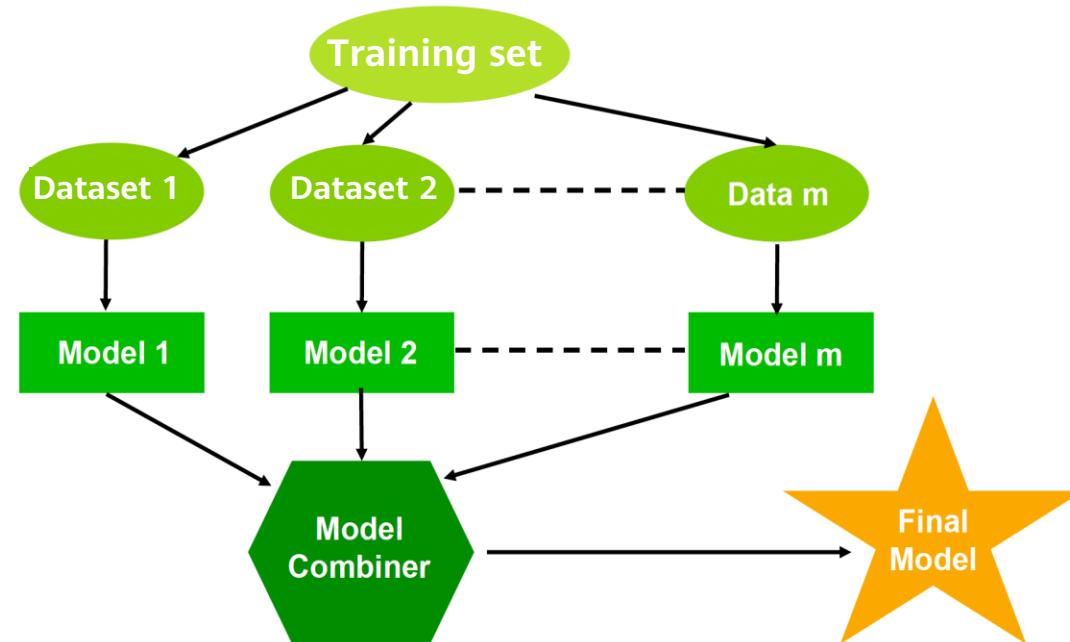
# Naive Bayes (2)

- Independent assumption of features.
  - For example, if a fruit is red, round, and about 10 cm (3.94 in.) in diameter, it can be considered an apple.
  - A Naive Bayes classifier considers that each feature independently contributes to the probability that the fruit is an apple, regardless of any possible correlation between the color, roundness, and diameter.

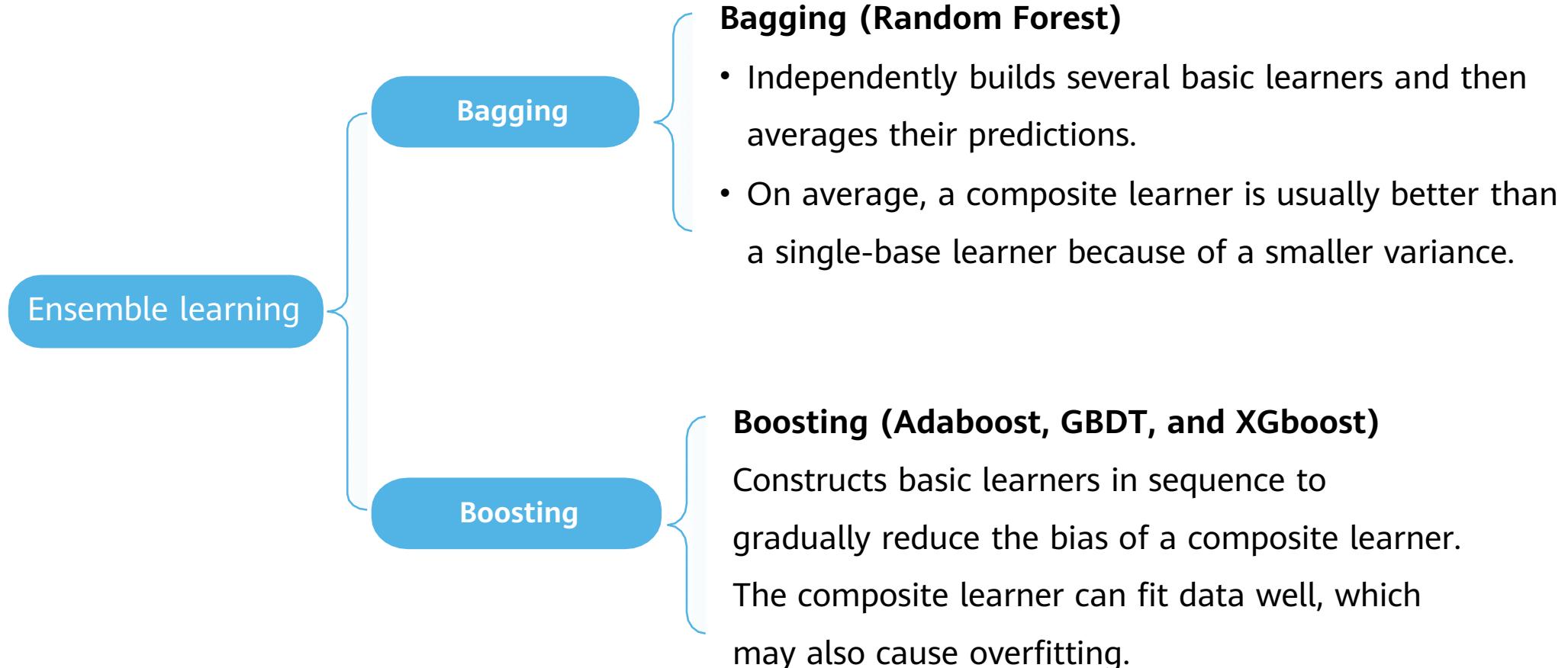


# Ensemble Learning

- Ensemble learning is a machine learning paradigm in which multiple learners are trained and combined to solve the same problem. When multiple learners are used, the integrated generalization capability can be much stronger than that of a single learner.
- If you ask a complex question to thousands of people at random and then summarize their answers, the summarized answer is better than an expert's answer in most cases. This is the wisdom of the masses.

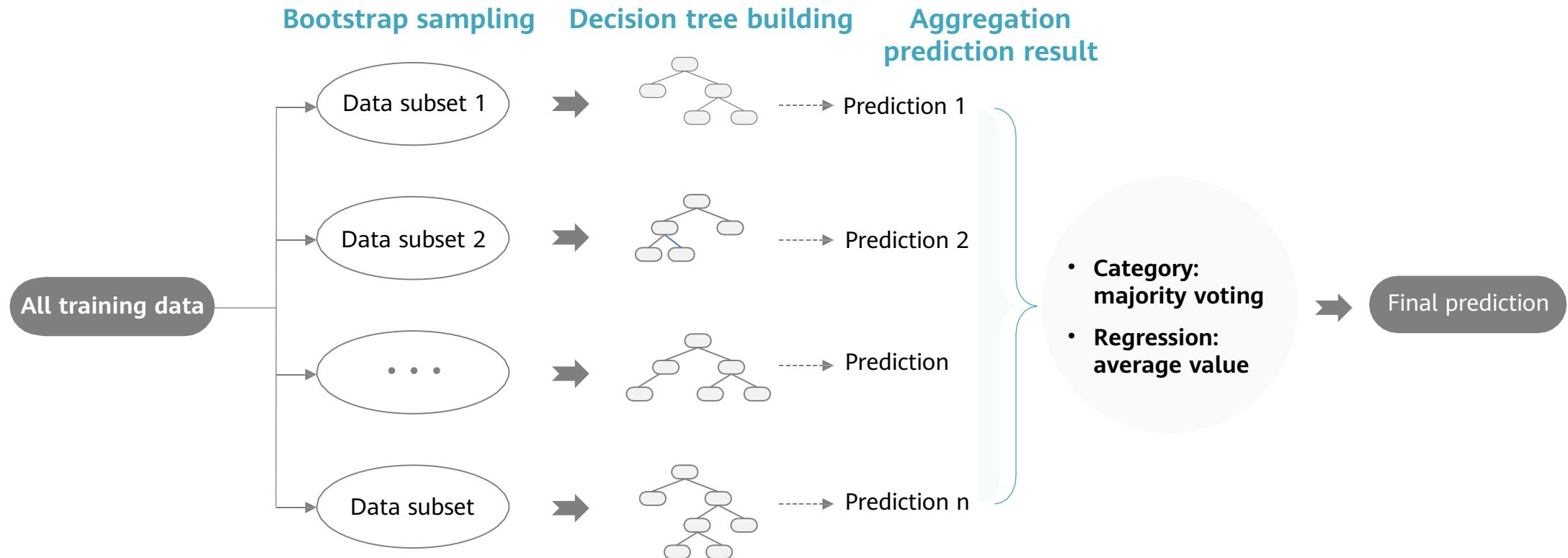


# Classification of Ensemble Learning



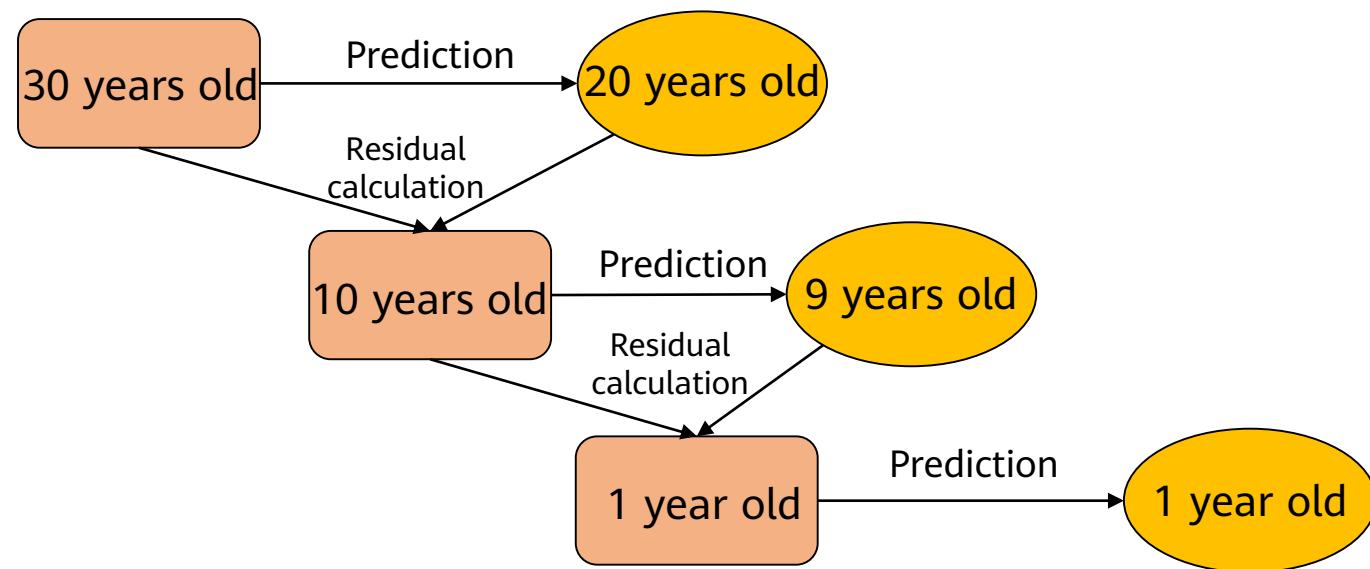
# Ensemble Methods in Machine Learning (1)

- Random forest = Bagging + CART decision tree
- Random forests build multiple decision trees and merge them together to make predictions more accurate and stable.
  - Random forests can be used for classification and regression problems.



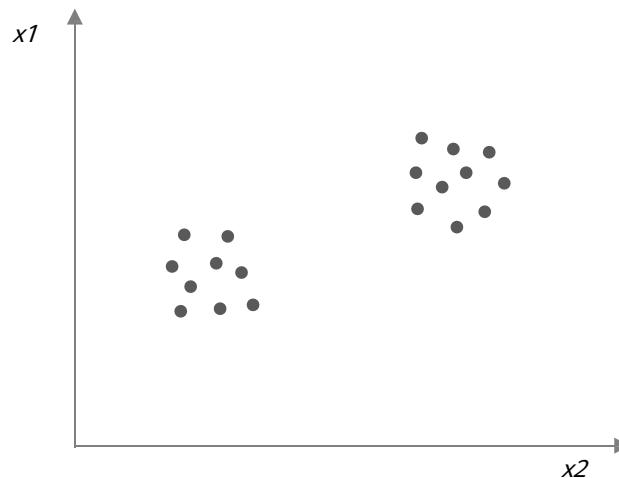
# Ensemble Methods in Machine Learning (2)

- GBDT is a type of boosting algorithm.
- For an aggregative mode, the sum of the results of all the basic learners equals the predicted value. In essence, the residual of the error function to the predicted value is fit by the next basic learner. (The residual is the error between the predicted value and the actual value.)
- During model training, GBDT requires that the sample loss for model prediction be as small as possible.

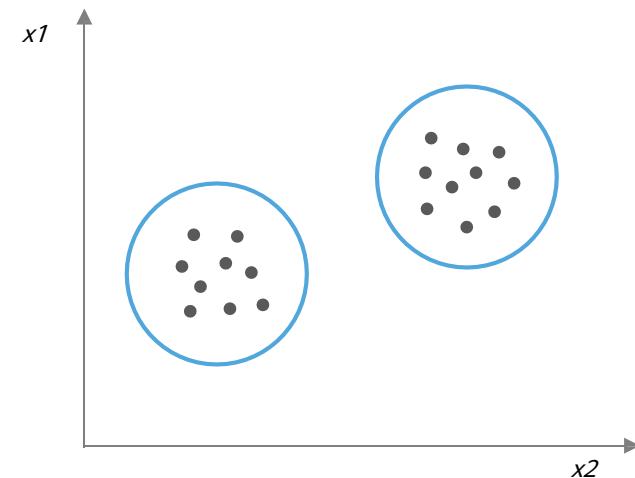


# Unsupervised Learning - K-means

- K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
- For the k-means algorithm, specify the final number of clusters (k). Then, divide n data objects into k clusters. The clusters obtained meet the following conditions: (1) Objects in the same cluster are highly similar. (2) The similarity of objects in different clusters is small.

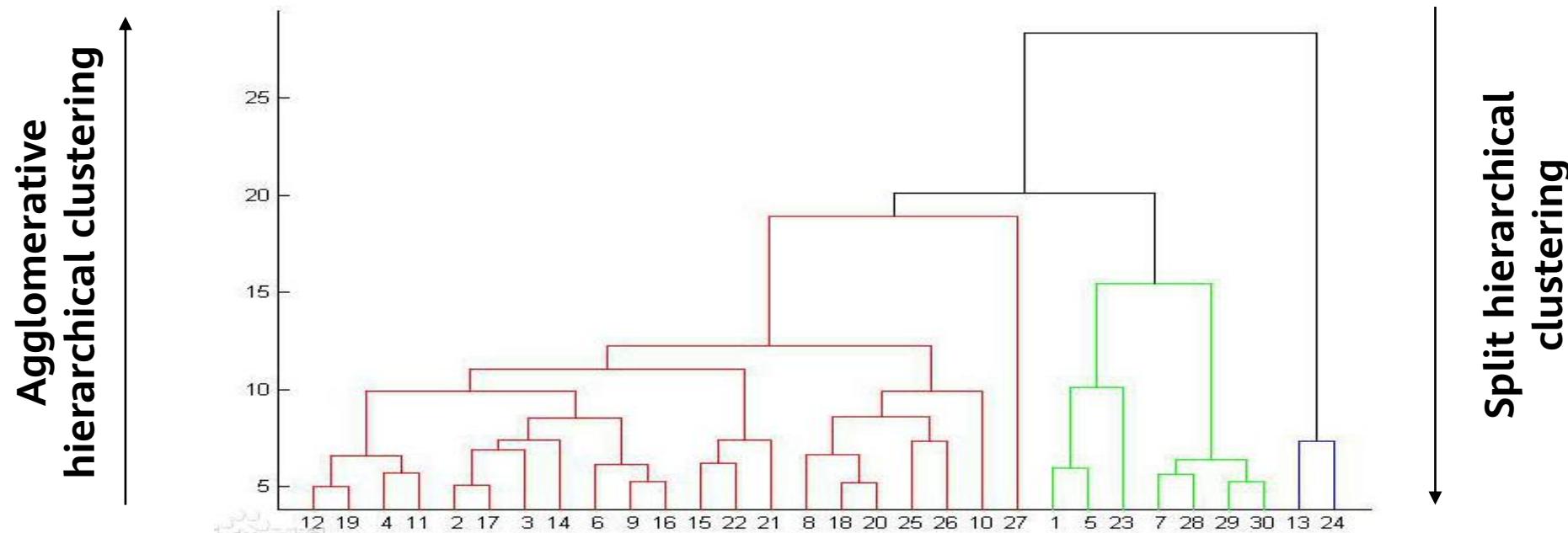


K-means clustering  
→  
The data is **not tagged**. K-means clustering can automatically classify datasets.



# Unsupervised Learning - Hierarchical Clustering

- Hierarchical clustering divides a dataset at different layers and forms a tree-like clustering structure. The dataset division may use a "bottom-up" aggregation policy, or a "top-down" splitting policy. The hierarchy of clustering is represented in a tree graph. The root is the unique cluster of all samples, and the leaves are the cluster of only a sample.



# Contents

1. Machine Learning Definition
2. Machine Learning Types
3. Machine Learning Process
4. Other Key Machine Learning Methods
5. Common Machine Learning Algorithms
- 6. Case study**

# Comprehensive Case

- Assume that there is a dataset containing the house areas and prices of 21,613 housing units sold in a city. Based on this data, we can predict the prices of other houses in the city.

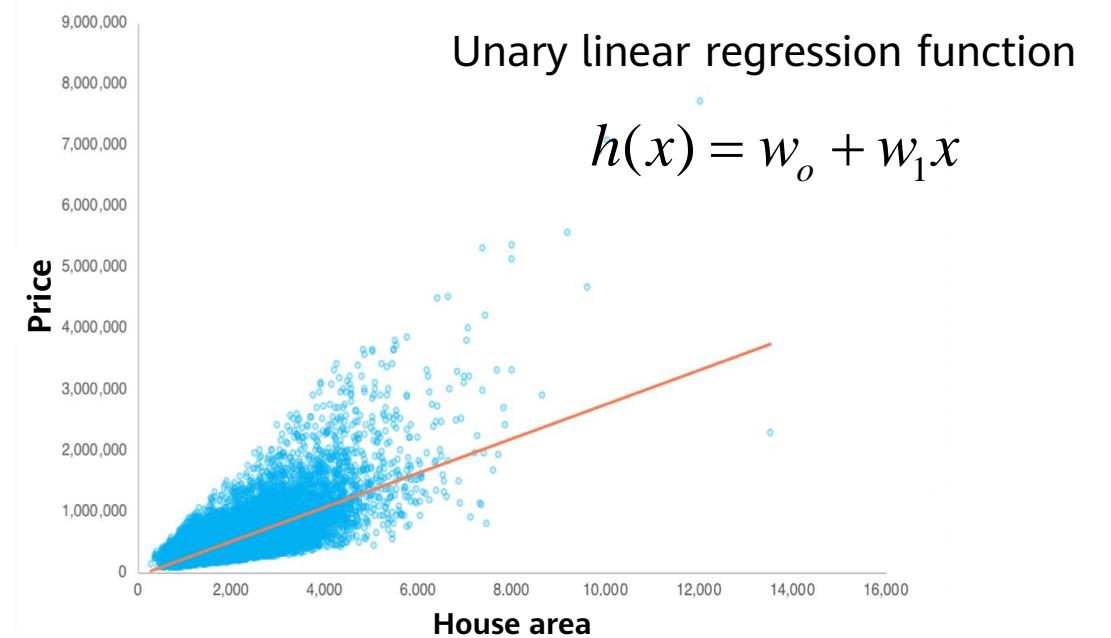
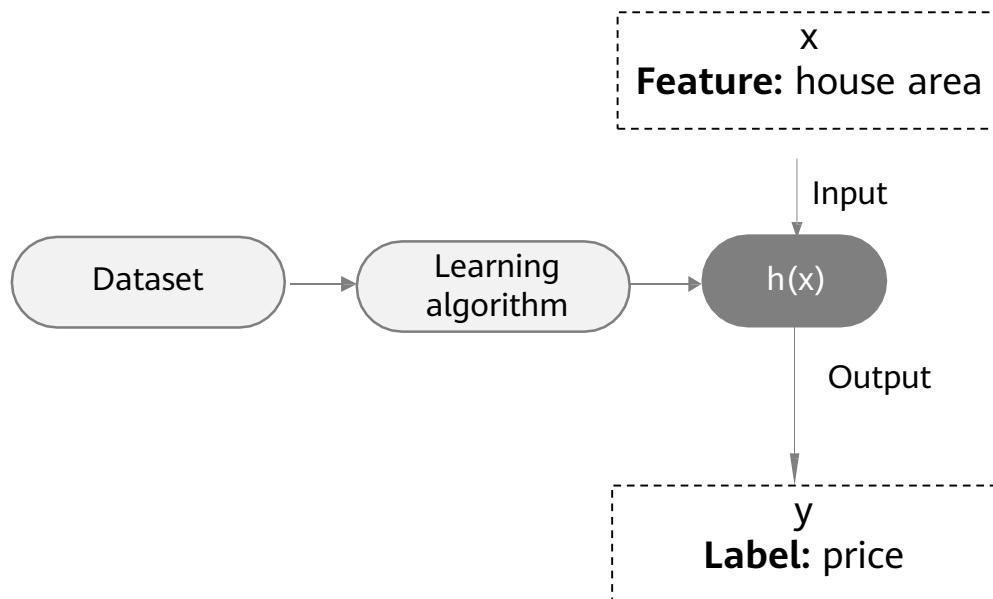
House Area	Price
1,180	221,900
2,570	538,000
770	180,000
1,960	604,000
1,680	510,000
5,420	1,225,000
1,715	257,500
1,060	291,850
1,160	468,000
1,430	310,000
1,370	400,000
1,810	530,000
...	...

Dataset



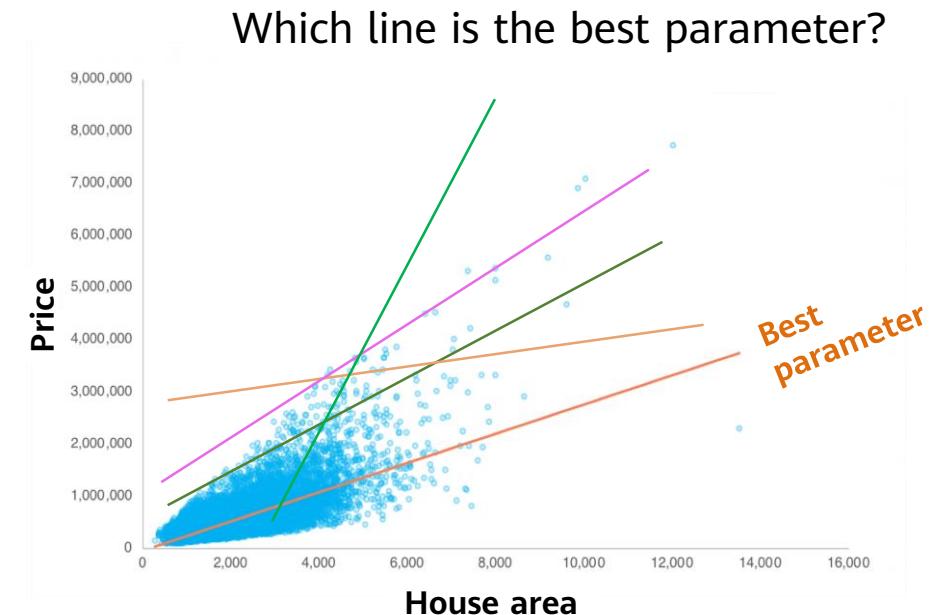
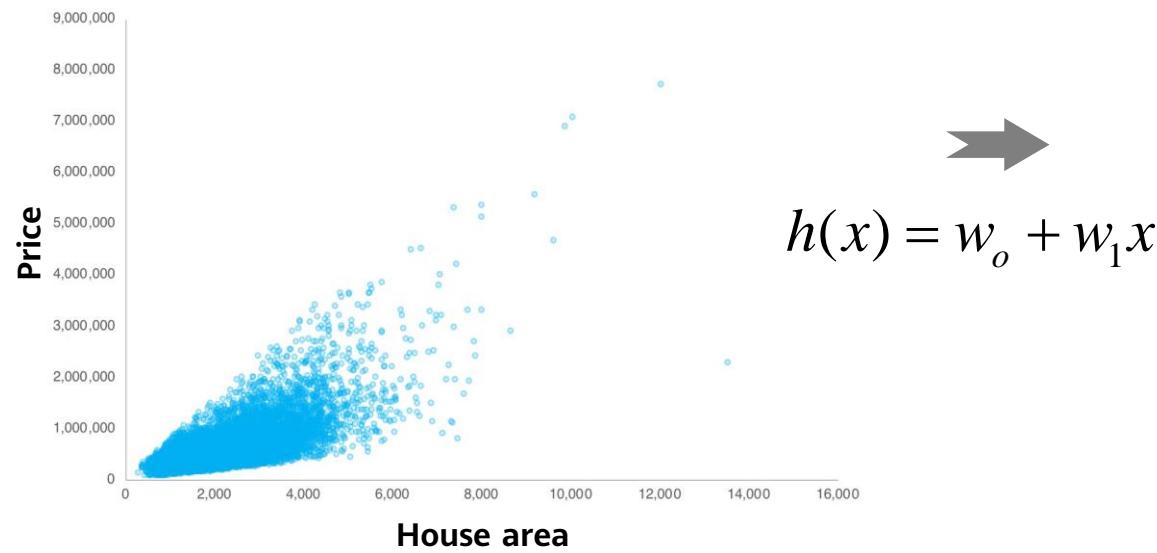
# Problem Analysis

- This case contains a large amount of data, including input  $x$  (house area), and output  $y$  (price), which is a continuous value. We can use **regression** of **supervised learning**. Draw a scatter chart based on the data and use **linear regression**.
- Our goal is to build a model function  $h(x)$  that infinitely approximates the function that expresses true distribution of the dataset.
- Then, use the model to predict unknown price data.



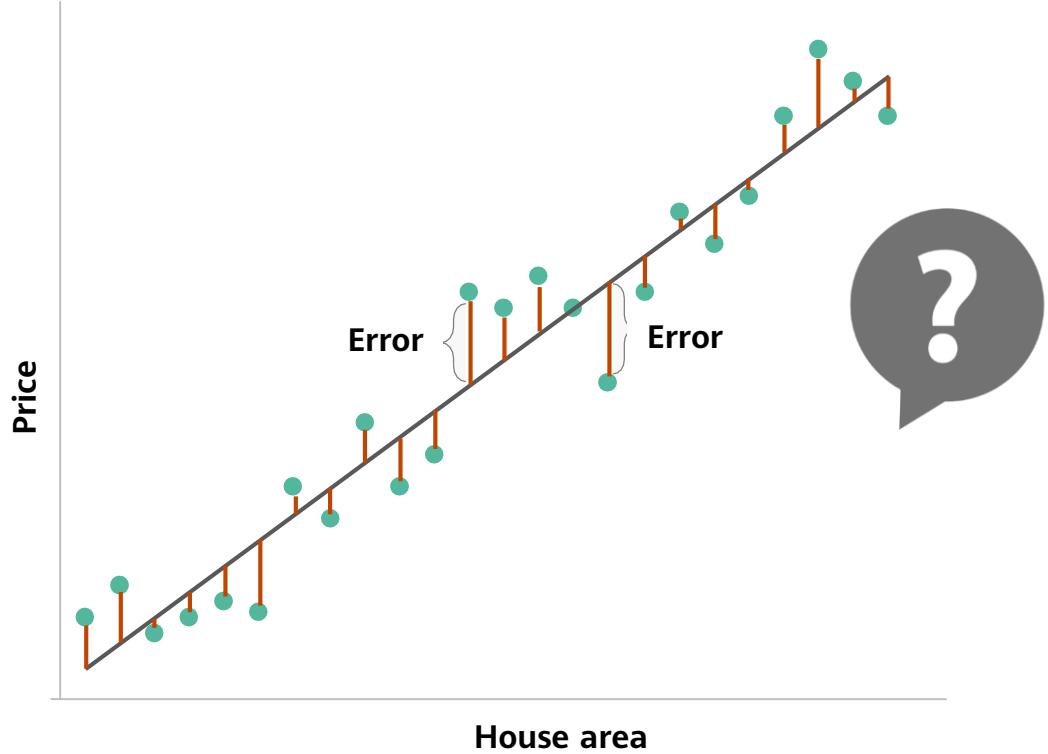
# Goal of Linear Regression

- Linear regression aims to find a straight line that best fits the dataset.
- Linear regression is a parameter-based model. Here, we need learning parameters  $w_0$  and  $w_1$ . When these two parameters are found, the best model appears.



# Loss Function of Linear Regression

- To find the optimal parameter, construct a loss function and find the parameter values when the loss function becomes the minimum.



Loss function of linear regression:

$$J(w) = \frac{1}{2m} \sum (h(x) - y)^2$$

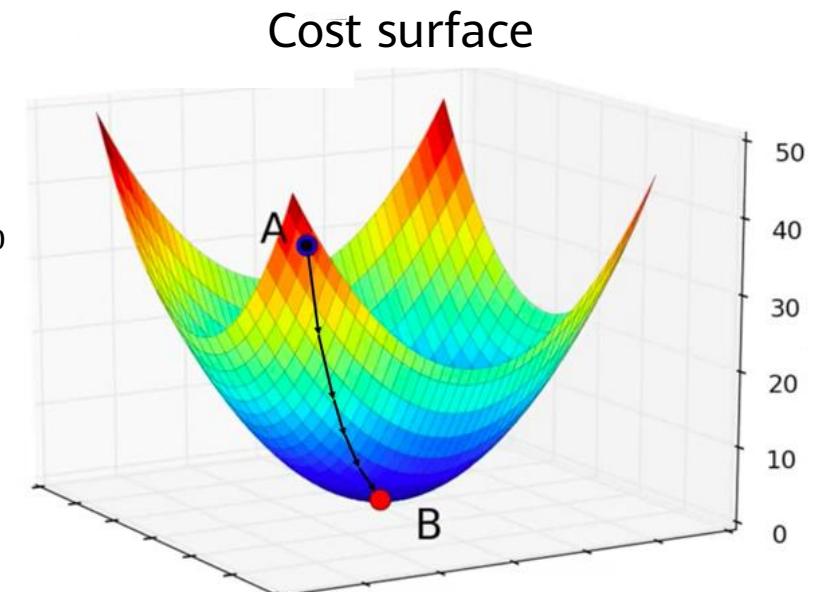
Goal:

$$\arg \min_w J(w) = \frac{1}{2m} \sum (h(x) - y)^2$$

- where,  $m$  indicates the number of samples,
- $h(x)$  indicates the predicted value, and  $y$  indicates the actual value.

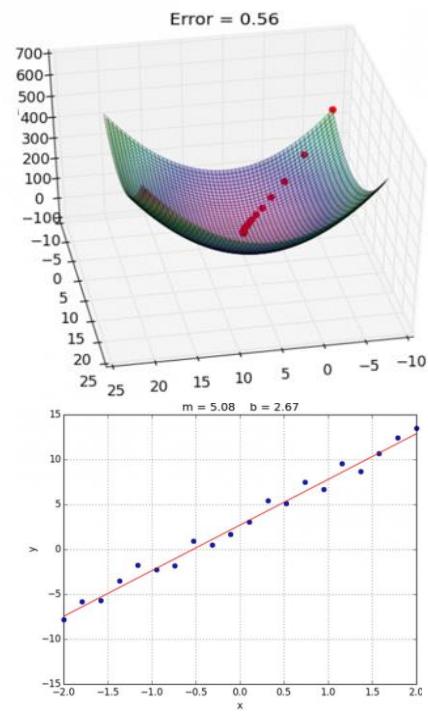
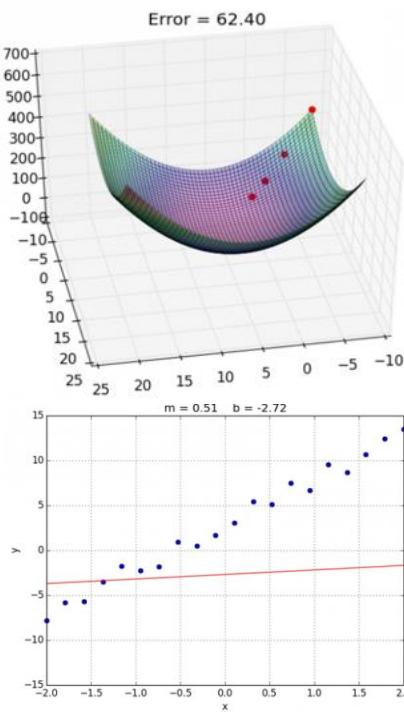
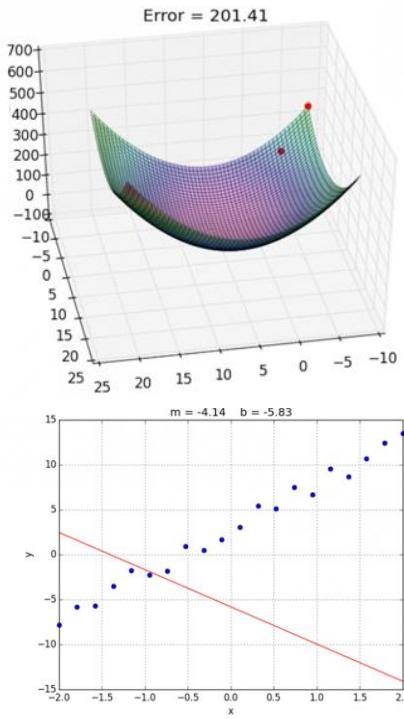
# Gradient Descent Method

- The gradient descent algorithm finds the minimum value of a function through iteration.
- It aims to randomize an initial point on the loss function, and then find the global minimum value of the loss function based on the negative gradient direction. Such parameter value is the optimal parameter value.
  - **Point A:** the position of  $w_0$  and  $w_1$  after random initialization.  
 $w_0$  and  $w_1$  are the required **parameters**.
  - **A-B connection line:** a track formed based on descents in a negative gradient direction. Upon each descent, values  $w_0$  and  $w_1$  **change**, and the regression line also changes.
  - **Point B:** global minimum value of the loss function.  
Final values of  $w_0$  and  $w_1$  are also found.



# Iteration Example

- The following is an example of a gradient descent iteration. We can see that as red points on the loss function surface gradually approach a lowest point, fitting of the linear regression red line with data becomes better and better. At this time, we can get the best parameters.

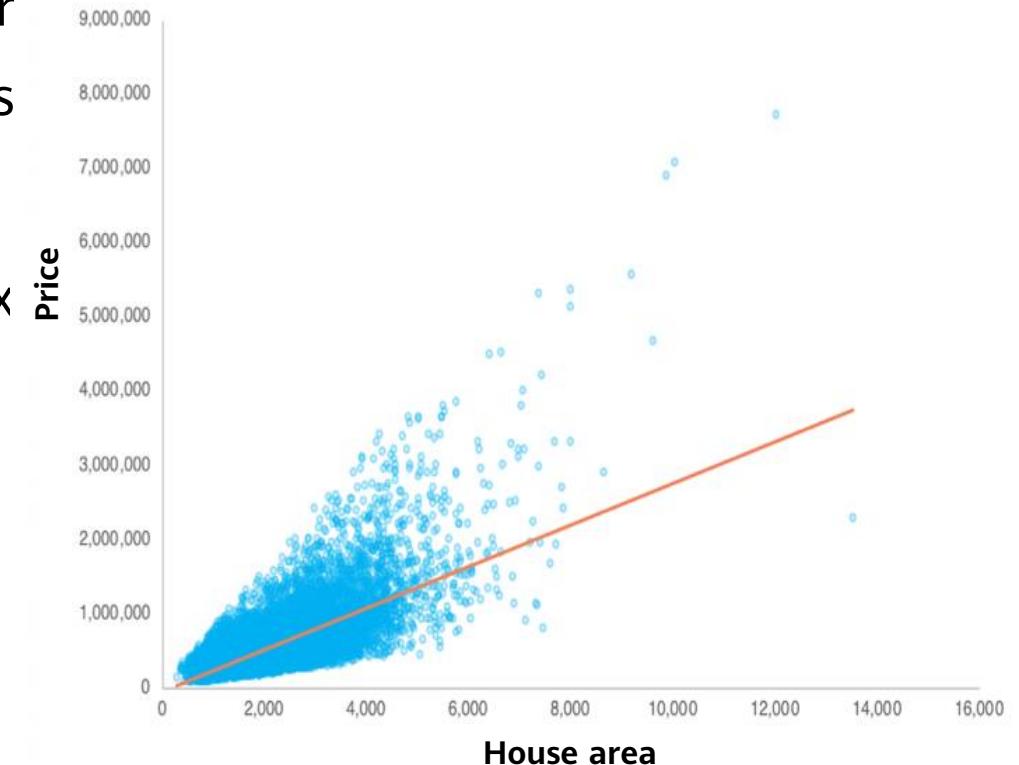


# Model Debugging and Application

- After the model is trained, test it with the test set to ensure the generalization capability.
- If overfitting occurs, use Lasso regression or Ridge regression with regularization terms and tune the hyperparameters.
- If underfitting occurs, use a more complex regression model, such as GBDT.
- Note:
  - For real data, pay attention to the functions of data cleansing and feature engineering.

The final model result is as follows:

$$h(x) = 280.62x - 43581$$



# Summary

- First, this course describes the definition and classification of machine learning, as well as problems machine learning solves. Then, it introduces key knowledge points of machine learning, including the overall procedure (data collection, data cleansing, feature extraction, model training, model training and evaluation, and model deployment), common algorithms (linear regression, logistic regression, decision tree, SVM, naive Bayes, KNN, ensemble learning, K-means, etc.), gradient descent algorithm, parameters and hyper-parameters.
- Finally, a complete machine learning process is presented by a case of using linear regression to predict house prices.

# Quiz

1. (True or false) Gradient descent iteration is the only method of machine learning algorithms. ( )
  - A. True
  - B. False
2. (Single-answer question) Which of the following algorithms is not supervised learning ? ( )
  - A. Linear regression
  - B. Decision tree
  - C. KNN
  - D. K-means

# Recommendations

---

- Online learning website
  - <https://e.huawei.com/en/talent/#/>
- Huawei Knowledge Base
  - <https://support.huawei.com/enterprise/en/knowledge?lang=en>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

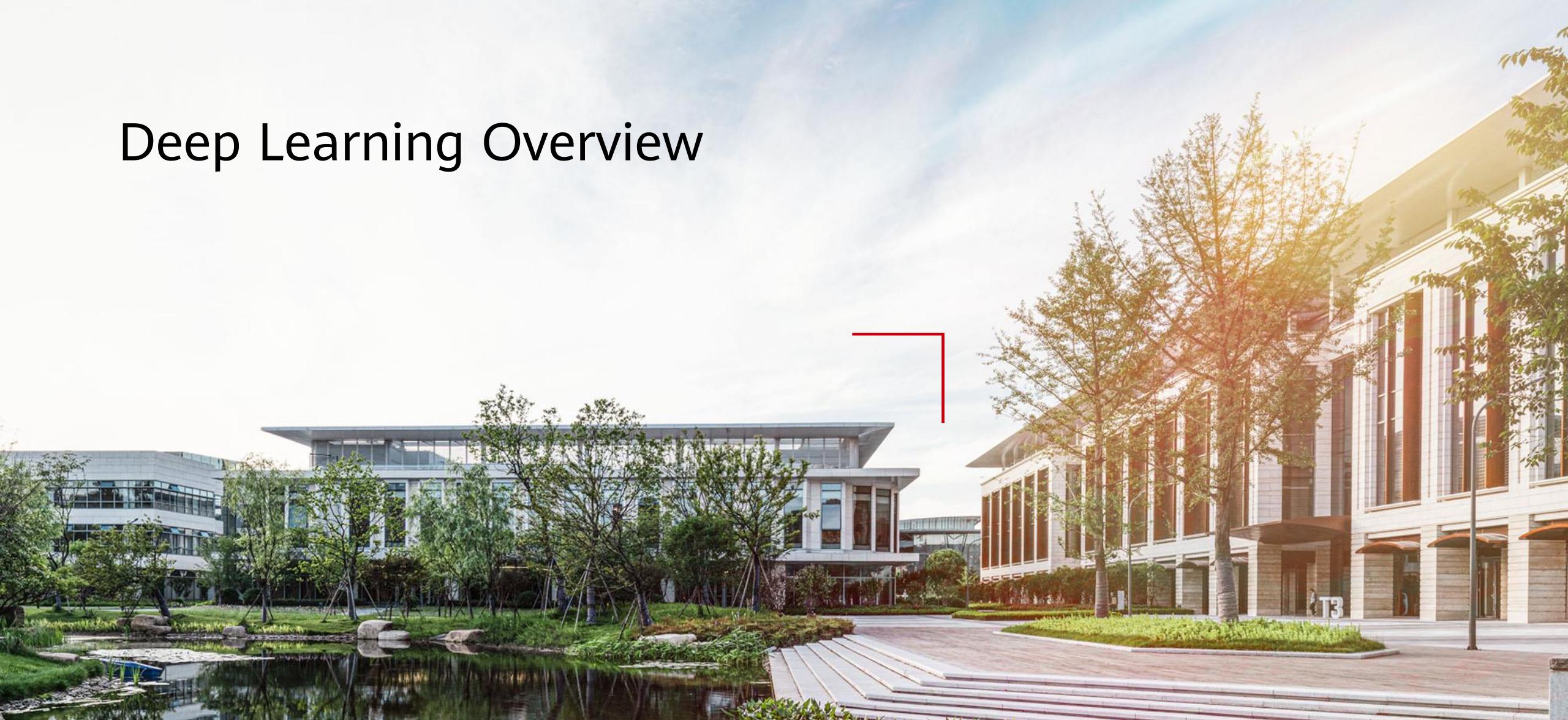
Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Deep Learning Overview



# Foreword

---

- The chapter describes the basic knowledge of deep learning, including the development history of deep learning, components and types of deep learning neural networks, and common problems in deep learning projects.

# Objectives

On completion of this course, you will be able to:

- Describe the definition and development of neural networks.
- Learn about the important components of deep learning neural networks.
- Understand training and optimization of neural networks.
- Describe common problems in deep learning.

# Contents

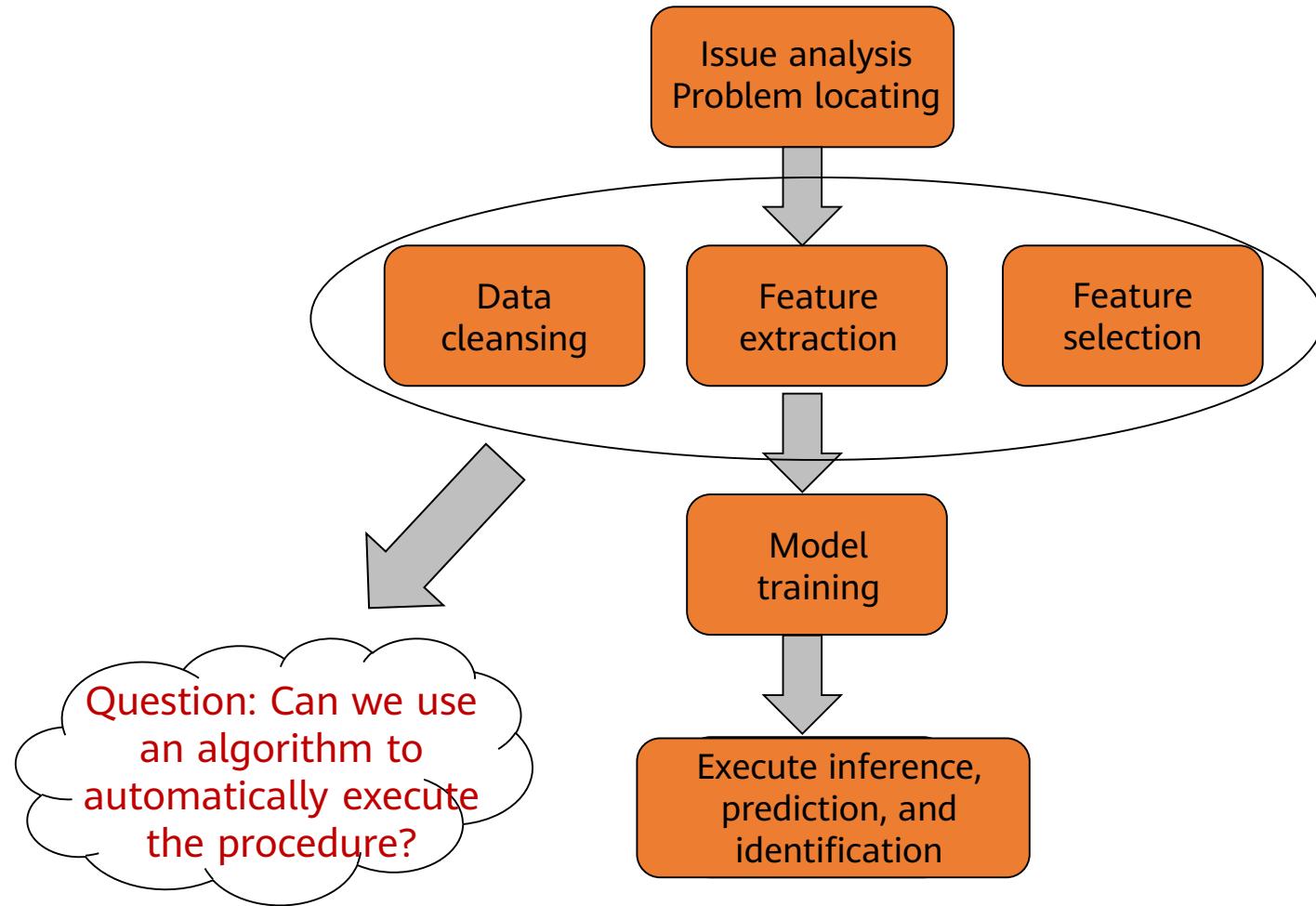
- 1. Deep Learning Summary**
2. Training Rules
3. Activation Function
4. Normalizer
5. Optimizer
6. Types of Neural Networks
7. Common Problems

# Traditional Machine Learning and Deep Learning

- As a model based on unsupervised feature learning and feature hierarchy learning, deep learning has great advantages in fields such as computer vision, speech recognition, and natural language processing.

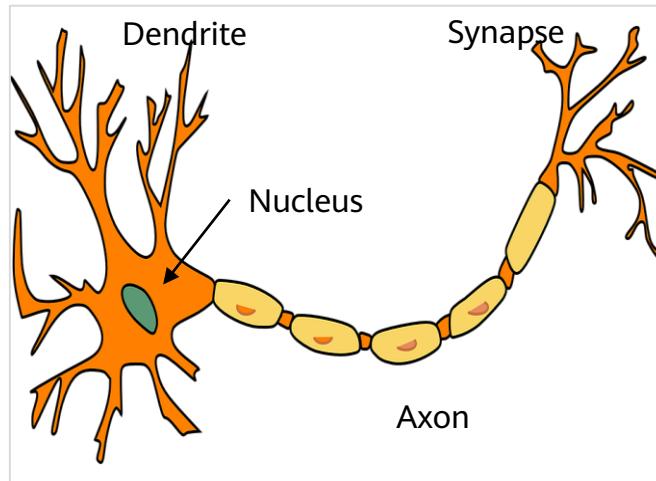
Traditional Machine Learning	Deep Learning
Low hardware requirements on the computer: Given the limited computing amount, the computer does not need a GPU for parallel computing generally.	Higher hardware requirements on the computer: To execute matrix operations on massive data, the computer needs a GPU to perform parallel computing.
Applicable to training under a small data amount and whose performance cannot be improved continuously as the data amount increases.	The performance can be high when high-dimensional weight parameters and massive training data are provided.
Level-by-level problem breakdown	E2E learning
Manual feature selection	Algorithm-based automatic feature extraction
Easy-to-explain features	Hard-to-explain features

# Traditional Machine Learning

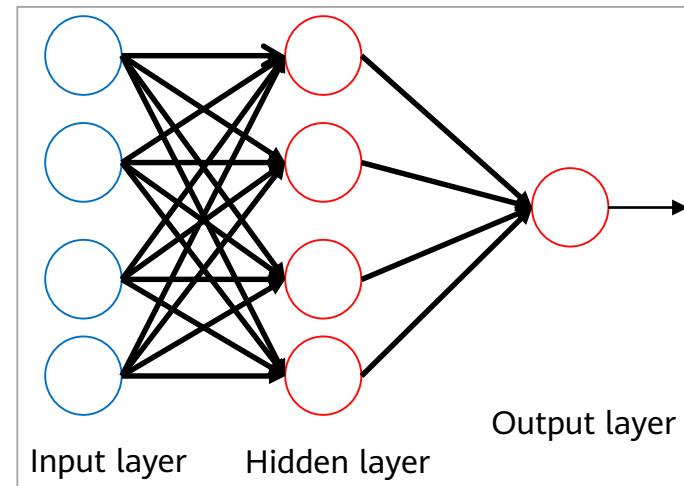


# Deep Learning

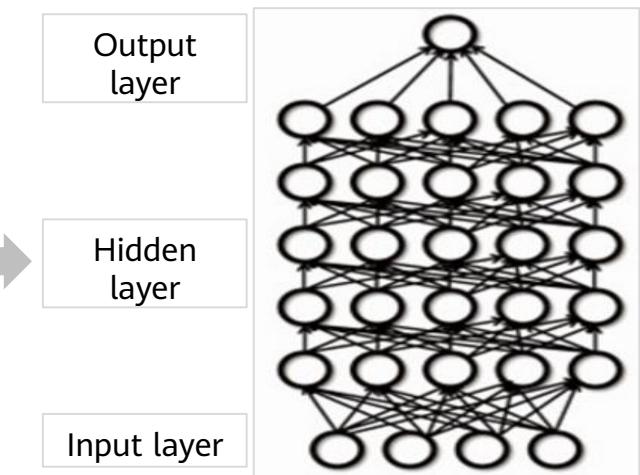
- Generally, the deep learning architecture is a deep neural network. "Deep" in "deep learning" refers to the number of layers of the neural network.



Human neural network



Perceptron

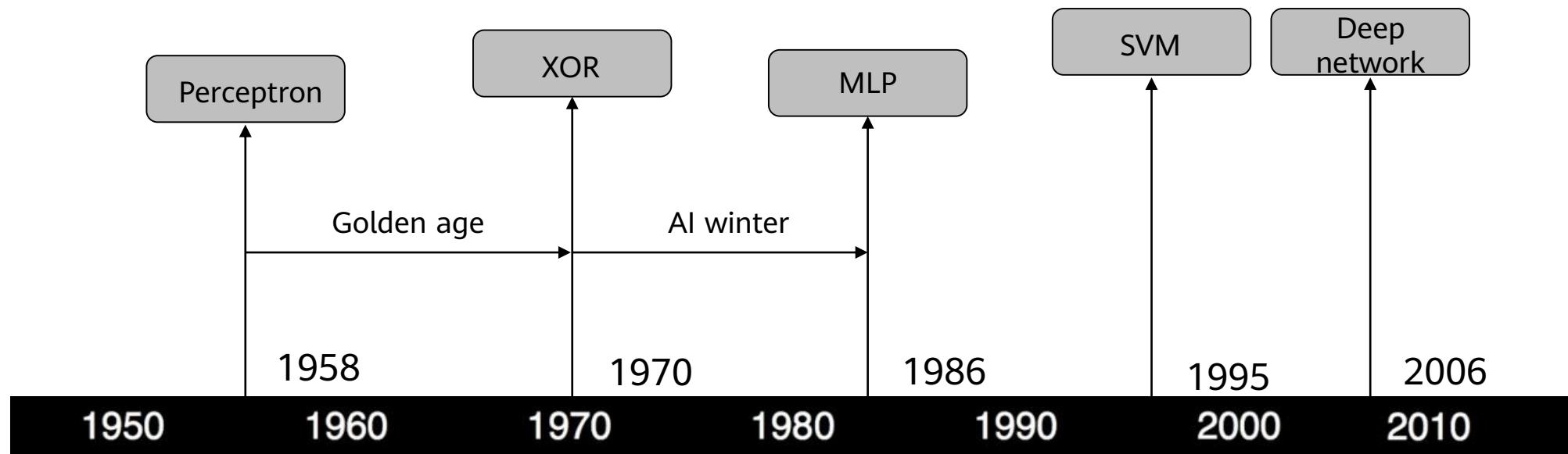


Deep neural network

# Neural Network

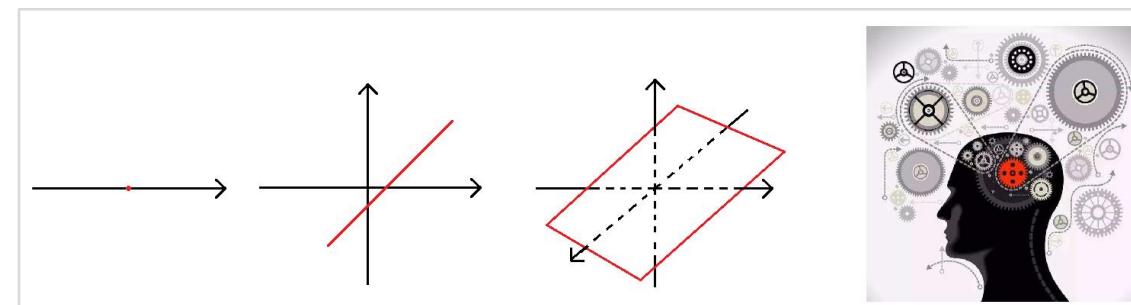
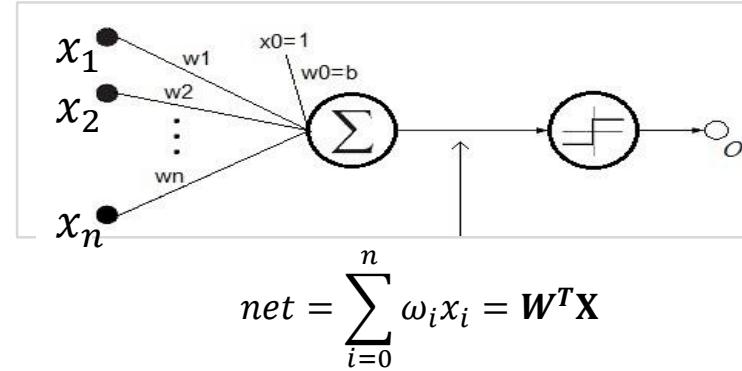
- Currently, the definition of the neural network has not been determined yet. Hecht Nielsen, a neural network researcher in the U.S., defines a neural network as a computer system composed of simple and highly interconnected processing elements, which process information by dynamic response to external inputs.
- A neural network can be simply expressed as an information processing system designed to imitate the human brain structure and functions based on its source, features, and explanations.
- Artificial neural network (neural network): Formed by artificial neurons connected to each other, the neural network extracts and simplifies the human brain's microstructure and functions. It is an important approach to simulate human intelligence and reflect several basic features of human brain functions, such as concurrent information processing, learning, association, model classification, and memory.

# Development History of Neural Networks



# Single-Layer Perceptron

- Input vector:  $X = [x_0, x_1, \dots, x_n]^T$
- Weight:  $W = [\omega_0, \omega_1, \dots, \omega_n]^T$ , in which  $\omega_0$  is the offset.
- Activation function:  $O = sign(net) = \begin{cases} 1, & net > 0, \\ -1, & otherwise. \end{cases}$
- The preceding perceptron is equivalent to a classifier. It uses the high-dimensional  $X$  vector as the input and performs binary classification on input samples in the high-dimensional space. When  $W^T X > 0$ ,  $O = 1$ . In this case, the samples are classified into a type. Otherwise,  $O = -1$ . In this case, the samples are classified into the other type. The boundary of these two types is  $W^T X = 0$ , which is a high-dimensional hyperplane.



Classification point  
 $Ax + B = 0$

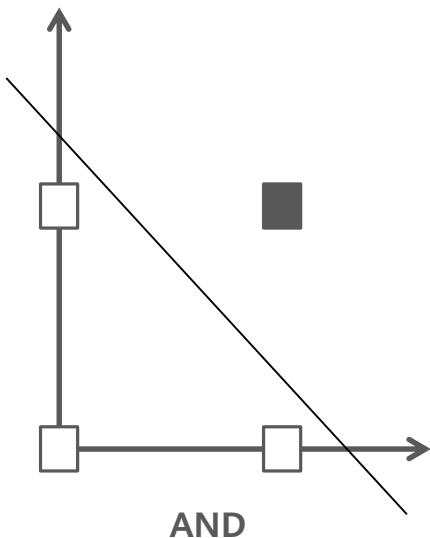
Classification line  
 $Ax + By + C = 0$

Classification plane  
 $Ax + By + Cz + D = 0$

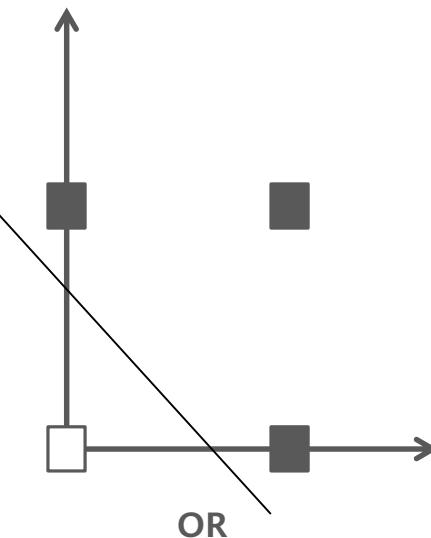
Classification hyperplane  
 $W^T X + b = 0$

# XOR Problem

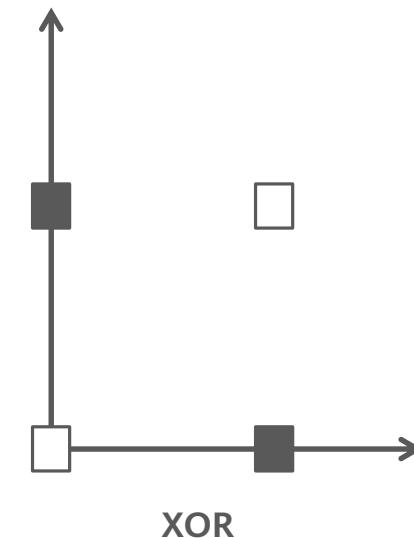
- In 1969, Minsky, an American mathematician and AI pioneer, proved that a perceptron is essentially a linear model that can only deal with linear classification problems, but cannot process non-linear data.



AND

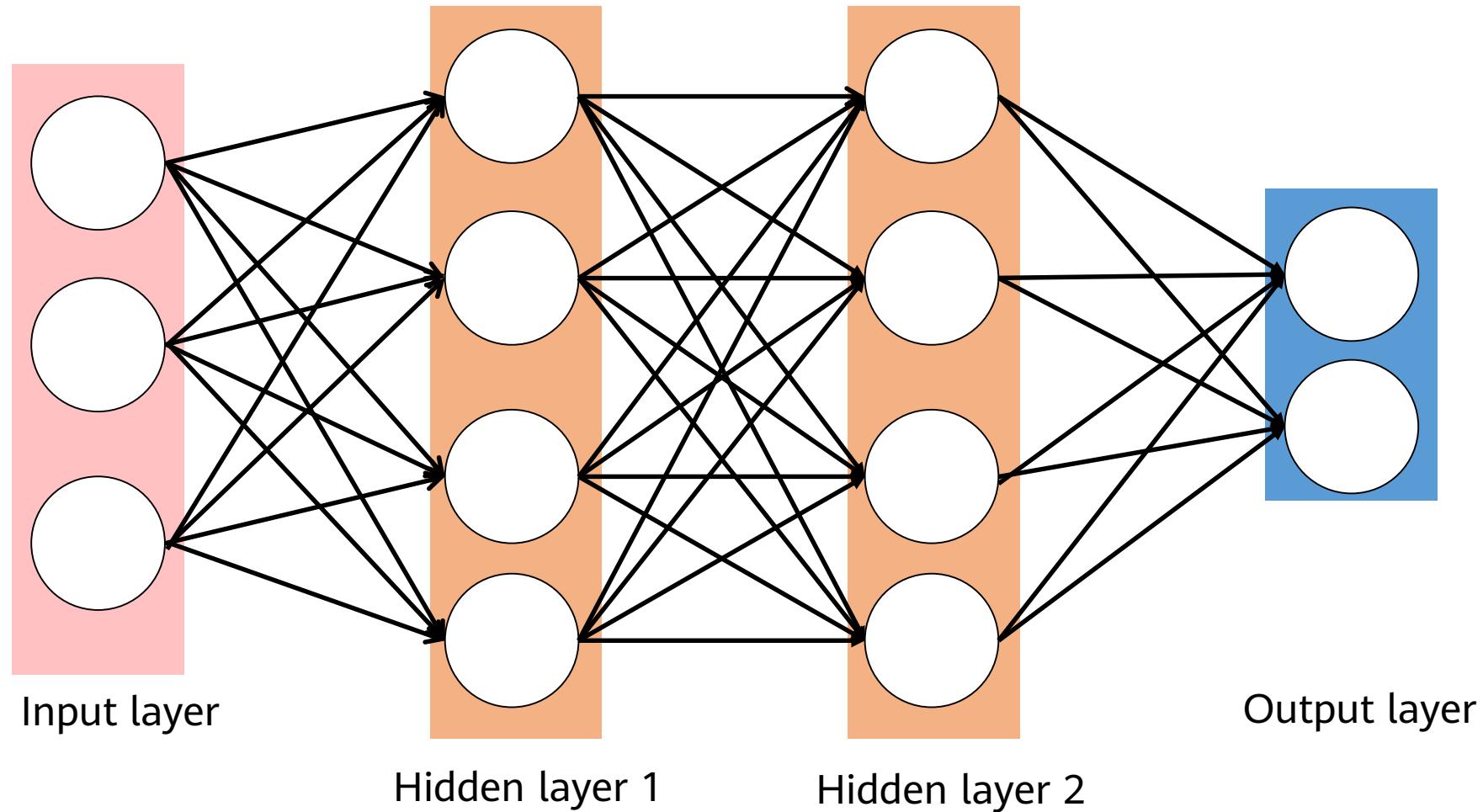


OR

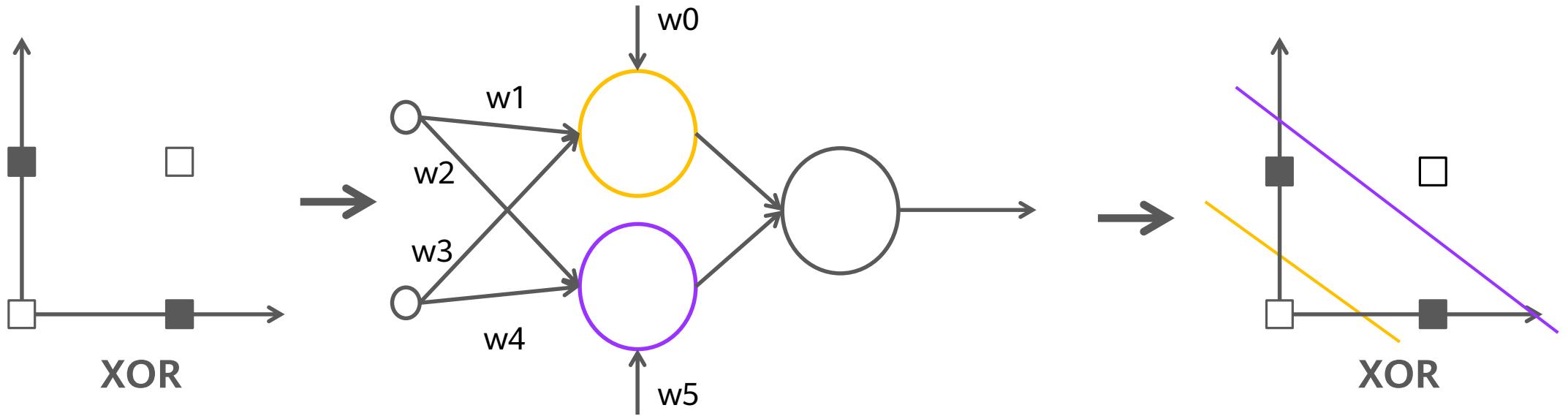


XOR

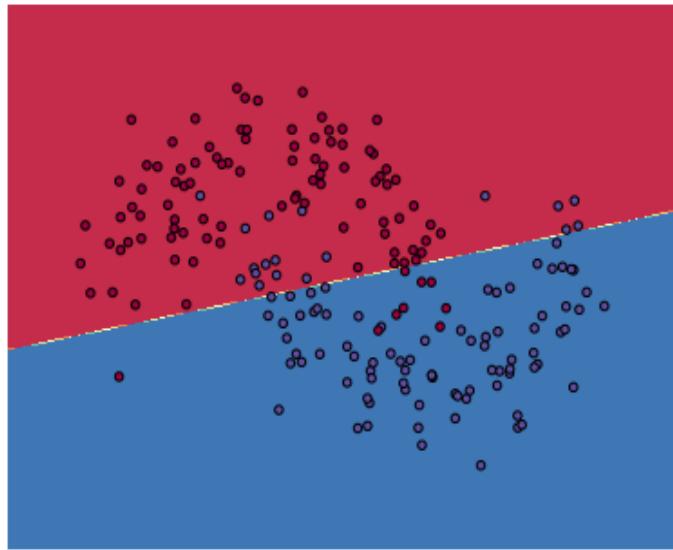
# Feedforward Neural Network



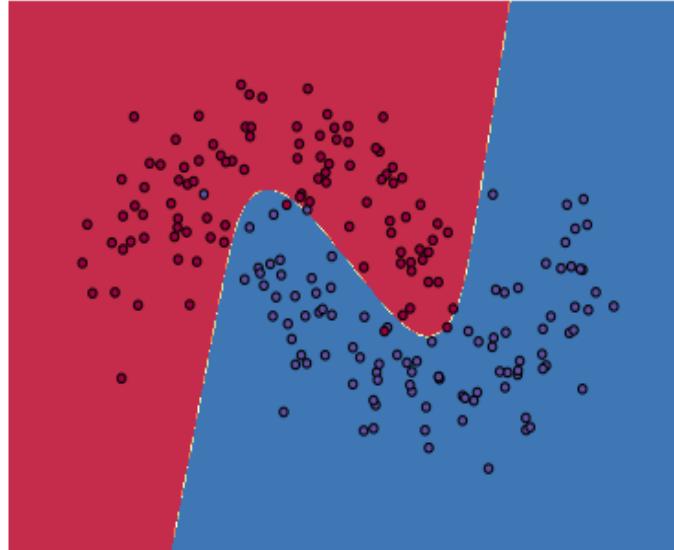
# Solution of XOR



# Impacts of Hidden Layers on A Neural Network



**0 hidden layers**



**3 hidden layers**



**20 hidden layers**

# Contents

1. Deep Learning Summary
- 2. Training Rules**
3. Activation Function
4. Normalizer
5. Optimizer
6. Types of Neural Networks
7. Common Problems

# Gradient Descent and Loss Function

- The gradient of the multivariate function  $o = f(x) = f(x_0, x_1, \dots, x_n)$  at  $X' = [x_0', x_1', \dots, x_n']^T$  is shown as follows:

$$\nabla f(x_0', x_1', \dots, x_n') = \left[ \frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right]^T |_{X=X'},$$

The direction of the gradient vector is the fastest growing direction of the function. As a result, the direction of the negative gradient vector  $-\nabla f$  is the fastest descent direction of the function.

- During the training of the deep learning network, target classification errors must be parameterized. A **loss function (error function)** is used, which reflects the error between the target output and actual output of the perceptron. For a single training sample  $x$ , the most common error function is the **Quadratic cost function**.

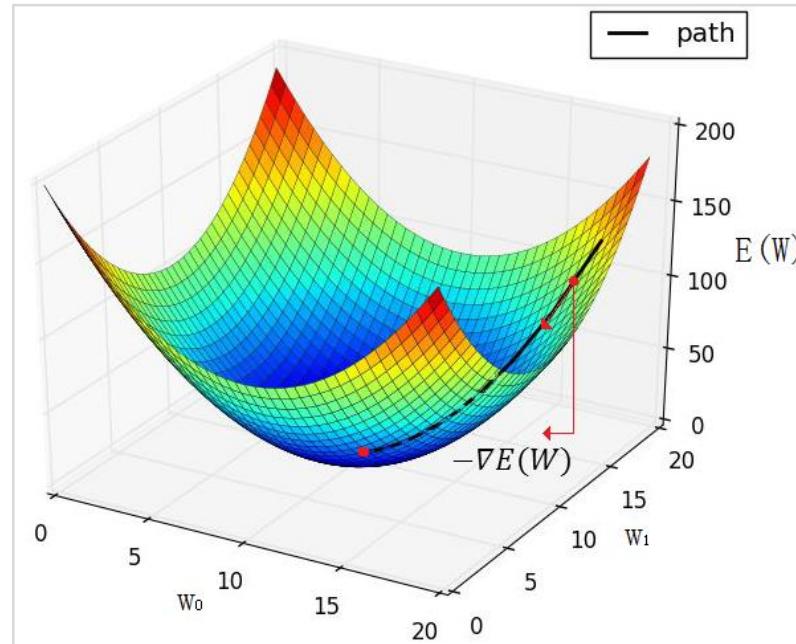
$$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2,$$

In the preceding function,  $d$  is one neuron in the output layer,  $D$  is all the neurons in the output layer,  $t_d$  is the target output, and  $o_d$  is the actual output.

- The gradient descent method enables the loss function to search along the negative gradient direction and update the parameters iteratively, finally minimizing the loss function.

# Extrema of the Loss Function

- Purpose: The loss function  $E(W)$  is defined on the weight space. The objective is to search for the weight vector  $W$  that can minimize  $E(W)$ .
- Limitation: No effective method can solve the extremum in mathematics on the complex high-dimensional surface of  $E(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$ .



Example of gradient descent of  
binary paraboloid

# Common Loss Functions in Deep Learning

- Quadratic cost function:

$$E(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- Cross entropy error function:

$$E(W) = -\frac{1}{n} \sum_x \sum_{d \in D} [t_d \ln o_d + (1 - t_d) \ln(1 - o_d)]$$

- The cross entropy error function depicts the distance between two probability distributions, which is a widely used loss function for classification problems.
- Generally, the mean square error function is used to solve the regression problem, while the cross entropy error function is used to solve the classification problem.

# Batch Gradient Descent Algorithm (BGD)

- In the training sample set  $D$ , each sample is recorded as  $\langle X, t \rangle$ , in which  $X$  is the input vector,  $t$  the target output,  $o$  the actual output, and  $\eta$  the learning rate.
  - Initializes each  $w_i$  to a random value with a smaller absolute value.
  - Before the end condition is met:
    - Initializes each  $\Delta w_i$  to zero.
    - For each  $\langle X, t \rangle$  in  $D$ :
      - Input  $X$  to this unit and calculate the output  $o$ .
      - For each  $w_i$  in this unit:  $\Delta w_i += -\eta \frac{1}{n} \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$ .
    - For each  $w_i$  in this unit:  $w_i += \Delta w_i$ .
- The gradient descent algorithm of this version is not commonly used because:
  - The convergence process is very slow as all training samples need to be calculated every time the weight is updated.

# Stochastic Gradient Descent Algorithm (SGD)

- To address the BGD algorithm defect, a common variant called Incremental Gradient Descent algorithm is used, which is also called the Stochastic Gradient Descent (SGD) algorithm. One implementation is called Online Learning, which updates the gradient based on each sample:

$$\Delta w_i = -\eta \frac{1}{n} \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i} \Rightarrow \Delta w_i = -\eta \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}.$$

- ONLINE-GRADIENT-DESCENT**
  - Initializes each  $w_i$  to a random value with a smaller absolute value.
  - Before the end condition is met:
    - Generates a random  $\langle X, t \rangle$  from  $D$  and does the following calculation:
      - Input  $X$  to this unit and calculate the output  $o$ .
      - For each  $w_i$  in this unit:  $w_i += -\eta \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$ .

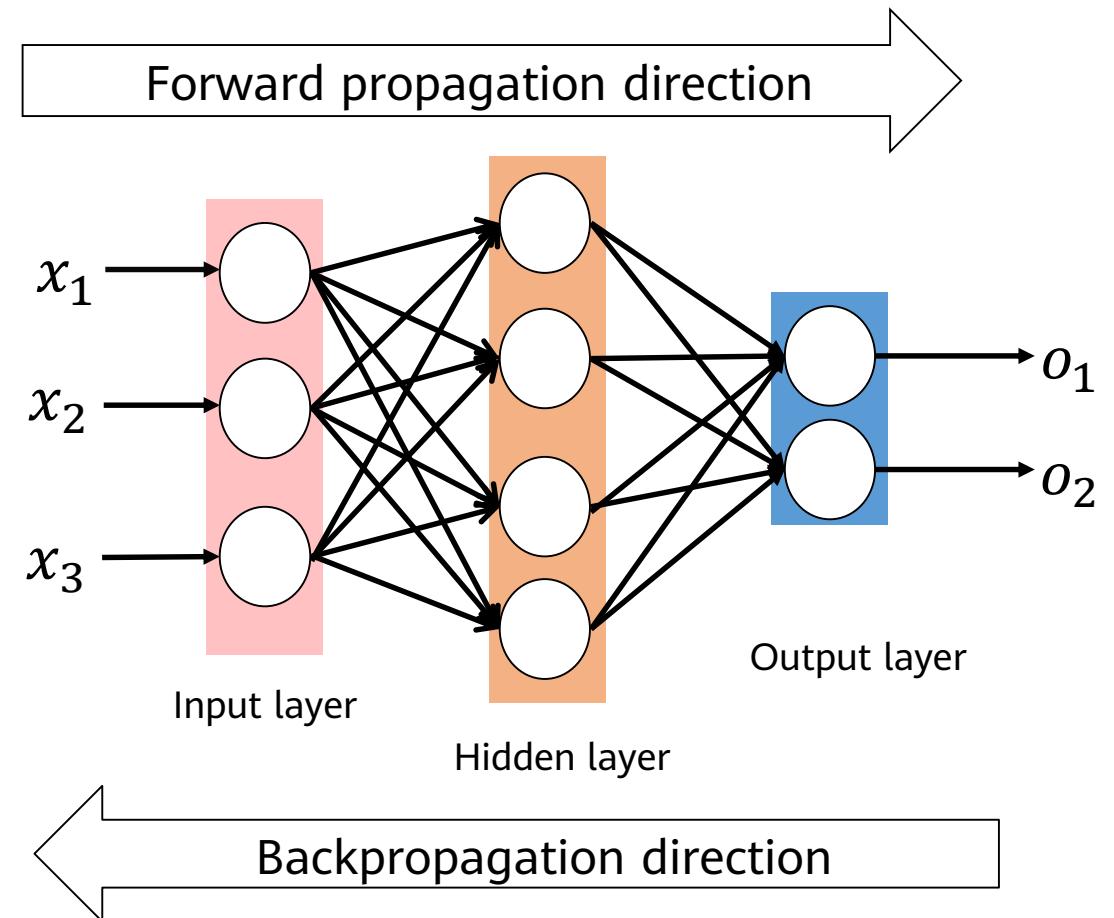
# Mini-Batch Gradient Descent Algorithm (MBGD)

- To address the defects of the previous two gradient descent algorithms, the Mini-batch Gradient Descent Algorithm (MBGD) was proposed and has been most widely used. A small number of Batch Size (BS) samples are used at a time to calculate  $\Delta w_i$ , and then the weight is updated accordingly.
- Batch-gradient-descent
  - Initializes each  $w_i$  to a random value with a smaller absolute value.
  - Before the end condition is met:
    - Initializes each  $\Delta w_i$  to zero.
    - For each  $\langle X, t \rangle$  in the BS samples in the next batch in  $D$ :
      - Input  $X$  to this unit and calculate the output  $o$ .
      - For each  $w_i$  in this unit:  $\Delta w_i += -\eta \frac{1}{n} \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$
    - For each  $w_i$  in this unit:  $w_i += \Delta w_i$
    - For the last batch, the training samples are mixed up in a random order.

# Backpropagation Algorithm (1)

- Signals are propagated in forward direction, and errors are propagated in backward direction.
- In the training sample set D, each sample is recorded as  $\langle X, t \rangle$ , in which X is the input vector, t the target output, o the actual output, and w the weight coefficient.
- Loss function:

$$E(w) = \frac{1}{2} \sum_{(d \in D)} (t_d - o_d)^2$$



# Backpropagation Algorithm (2)

- According to the following formulas, errors in the input, hidden, and output layers are accumulated to generate the error in the loss function.
- wc is the weight coefficient between the hidden layer and the output layer, while wb is the weight coefficient between the input layer and the hidden layer.  $f$  is the activation function, D is the output layer set, and C and B are the hidden layer set and input layer set respectively. Assume that the loss function is a quadratic cost function:

- Output layer error: 
$$E = \frac{1}{2} \sum_{(d \in D)} (t_d - o_d)^2$$

- Expanded hidden layer error: 
$$E = \frac{1}{2} \sum_{(d \in D)} [t_d - f(\text{net}_d)]^2 = \frac{1}{2} \sum_{(d \in D)} \left[ t_d - f\left(\sum_{(c \in C)} w_c y_c\right) \right]^2$$

- Expanded input layer error: 
$$E = \frac{1}{2} \sum_{(d \in D)} \left[ t_d - f\left(\sum_{(c \in C)} w_c f(\text{net}_c)\right) \right]^2 = \frac{1}{2} \sum_{(d \in D)} \left[ t_d - f\left(\sum_{(c \in C)} w_c f\left(\sum_{b \in B} w_b x_b\right)\right) \right]^2$$

# Backpropagation Algorithm (3)

- To minimize error E, the gradient descent iterative calculation can be used to solve wc and wb, that is, calculating wc and wb to minimize error E.
- Formula:

$$\Delta w_c = -\eta \frac{\partial E}{\partial w_c}, c \in C$$

$$\Delta w_b = -\eta \frac{\partial E}{\partial w_b}, b \in B$$

- If there are multiple hidden layers, chain rules are used to take a derivative for each layer to obtain the optimized parameters by iteration.

# Backpropagation Algorithm (4)

- For a neural network with any number of layers, the arranged formula for training is as follows:

$$\Delta w_{jk}^l = -\eta \delta_k^{l+1} f_j(z_j^l)$$

$$\delta_j^l = \begin{cases} f_j'(z_j^l)(t_j - f_j(z_j^l)), & l \in outputs, (1) \\ \sum_k \delta_k^{l+1} w_{jk}^l f_j'(z_j^l), & otherwise, (2) \end{cases}$$

- The BP algorithm is used to train the network as follows:
  - Takes out the next training sample  $\langle X, T \rangle$ , inputs  $X$  to the network, and obtains the actual output  $o$ .
  - Calculates output layer  $\delta$  according to the output layer error formula (1).
  - Calculates  $\delta$  of each hidden layer from output to input by iteration according to the hidden layer error propagation formula (2).
  - According to the  $\delta$  of each layer, the weight values of all the layer are updated.

# Contents

1. Deep Learning Summary
2. Training Rules
- 3. Activation Function**
4. Normalizer
5. Optimizer
6. Types of Neural Networks
7. Common Problems

# Activation Function

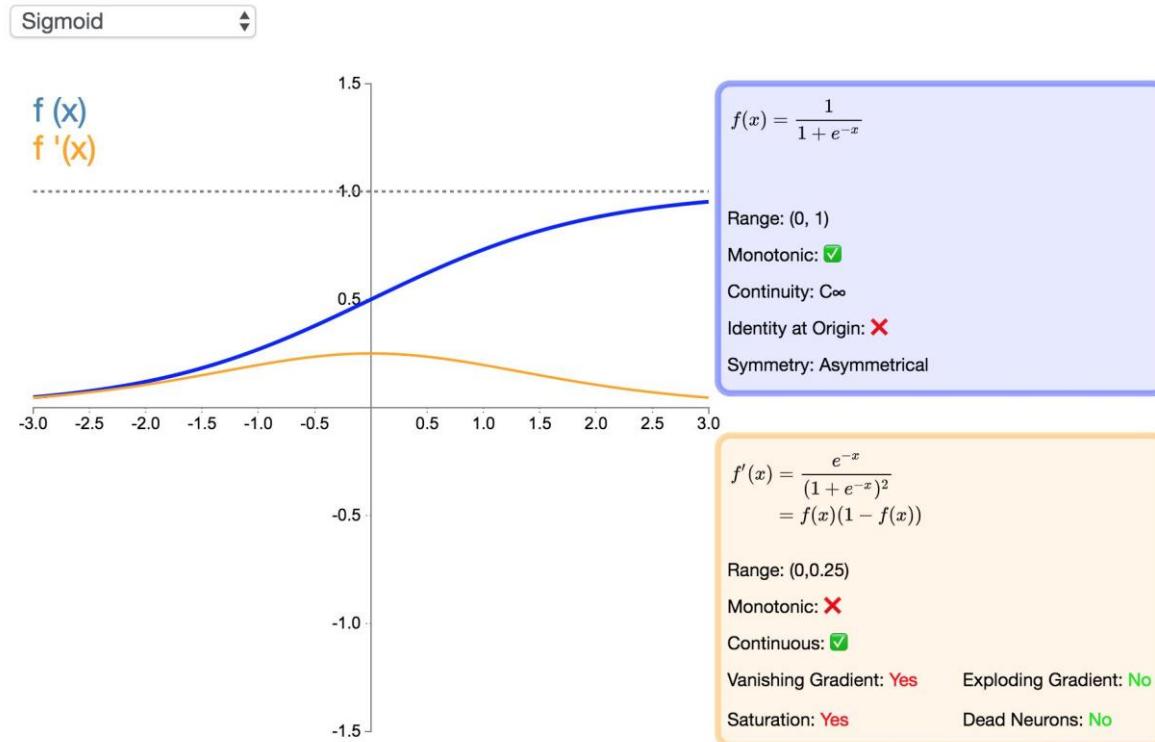
- Activation functions are important for the neural network model to learn and understand complex non-linear functions. They allow introduction of non-linear features to the network.
- Without activation functions, output signals are only simple linear functions. The complexity of linear functions is limited, and the capability of learning complex function mappings from data is low.

Activation Function

$$output = f(w_1x_1 + w_2x_2 + w_3x_3 \dots) = f(W^t \bullet X)$$

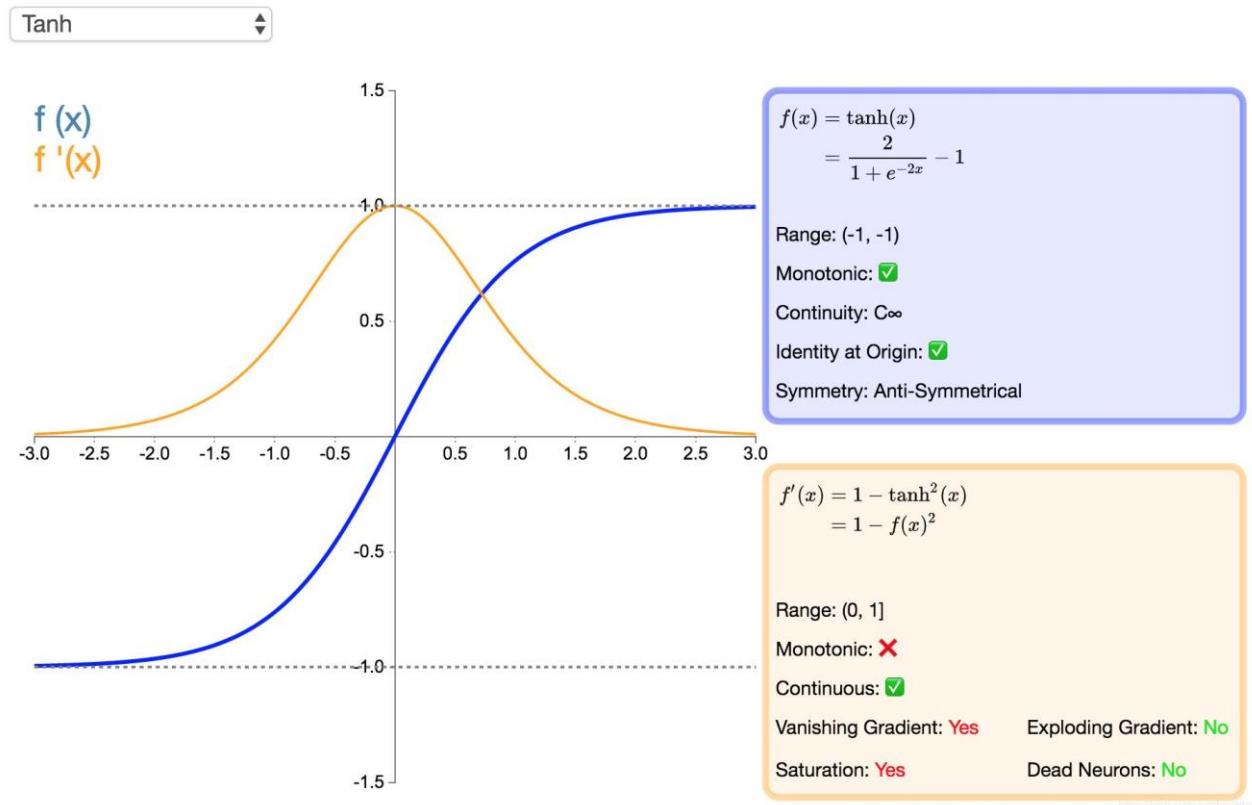
# Sigmoid

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



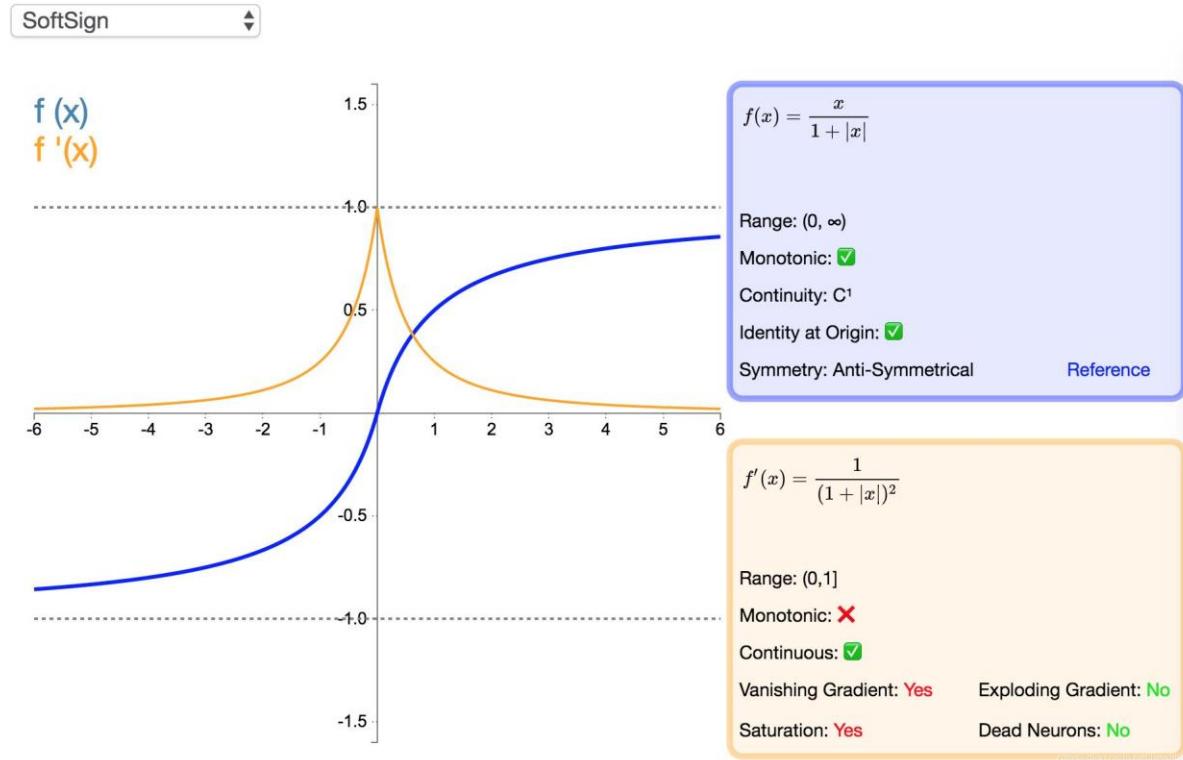
# Tanh

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



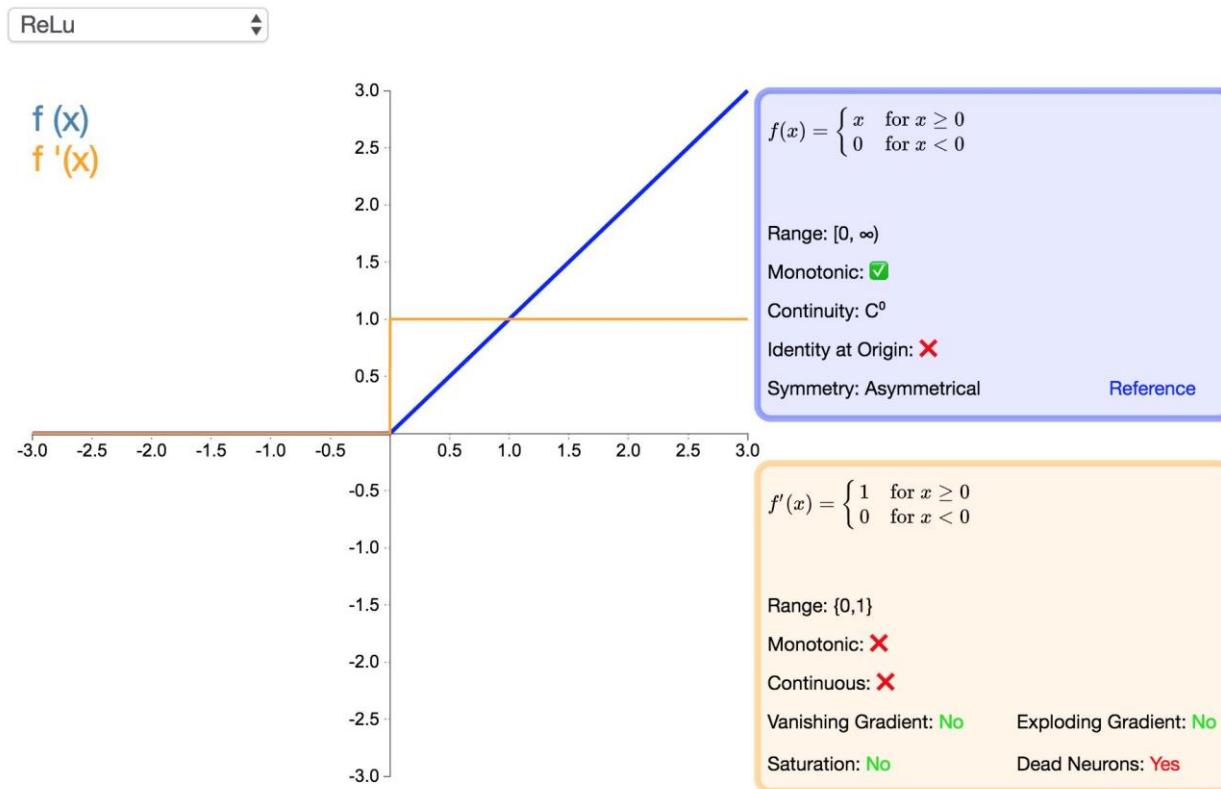
# Softsign

$$f(x) = \frac{x}{|x| + 1}$$



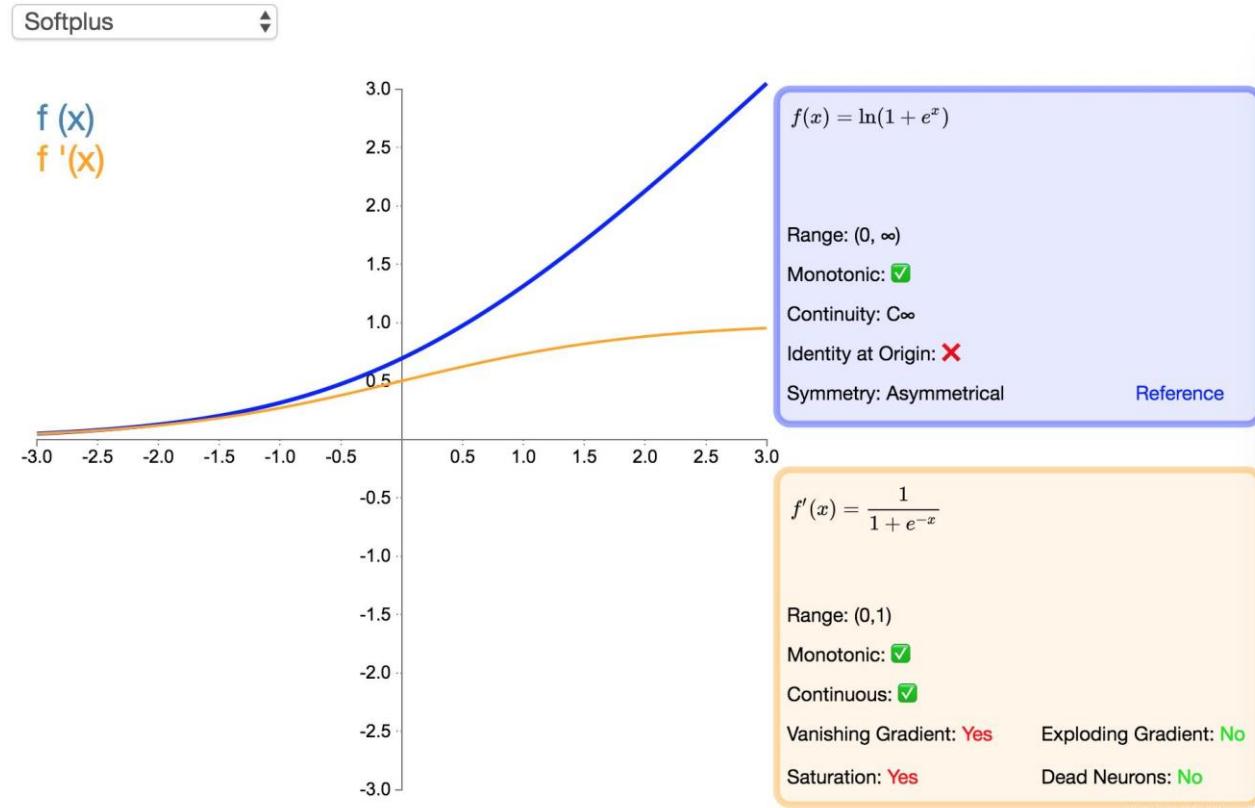
# Rectified Linear Unit (ReLU)

$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



# Softplus

$$f(x) = \ln(e^x + 1)$$



# Softmax

- Softmax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

- The Softmax function is used to map a K-dimensional vector of arbitrary real values to another K-dimensional vector of real values, where each vector element is in the interval (0, 1). All the elements add up to 1.
- The Softmax function is often used as the output layer of a multiclass classification task.

# Contents

1. Deep Learning Summary
2. Training Rules
3. Activation Function
- 4. Normalizer**
5. Optimizer
6. Types of Neural Networks
7. Common Problems

# Normalizer

- Regularization is an important and effective technology to reduce generalization errors in machine learning. It is especially useful for deep learning models that tend to be over-fit due to a large number of parameters. Therefore, researchers have proposed many effective technologies to prevent over-fitting, including:
  - Adding constraints to parameters, such as  $L_1$  and  $L_2$  norms
  - Expanding the training set, such as adding noise and transforming data
  - Dropout
  - Early stopping

# Penalty Parameters

- Many regularization methods restrict the learning capability of models by adding a penalty parameter  $\Omega(\theta)$  to the objective function  $J$ . Assume that the target function after regularization is  $\tilde{J}$ .

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta),$$

- Where  $\alpha \in [0, \infty)$  is a hyperparameter that weights the relative contribution of the norm penalty term  $\Omega$  and the standard objective function  $J(X; \theta)$ . If  $\alpha$  is set to 0, no regularization is performed. The penalty in regularization increases with  $\alpha$ .

# $L_1$ Regularization

- Add  $L_1$  norm constraint to model parameters, that is,

$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \|w\|_1,$$

- If a gradient method is used to resolve the value, the parameter gradient is

$$\nabla \tilde{J}(w) = \alpha sign(w) + \nabla J(w).$$

# $L_2$ Regularization

- Add norm penalty term  $L_2$  to prevent overfitting.

$$\tilde{J}(w; X, y) = J(w; X, y) + \frac{1}{2} \alpha \|w\|_2^2,$$

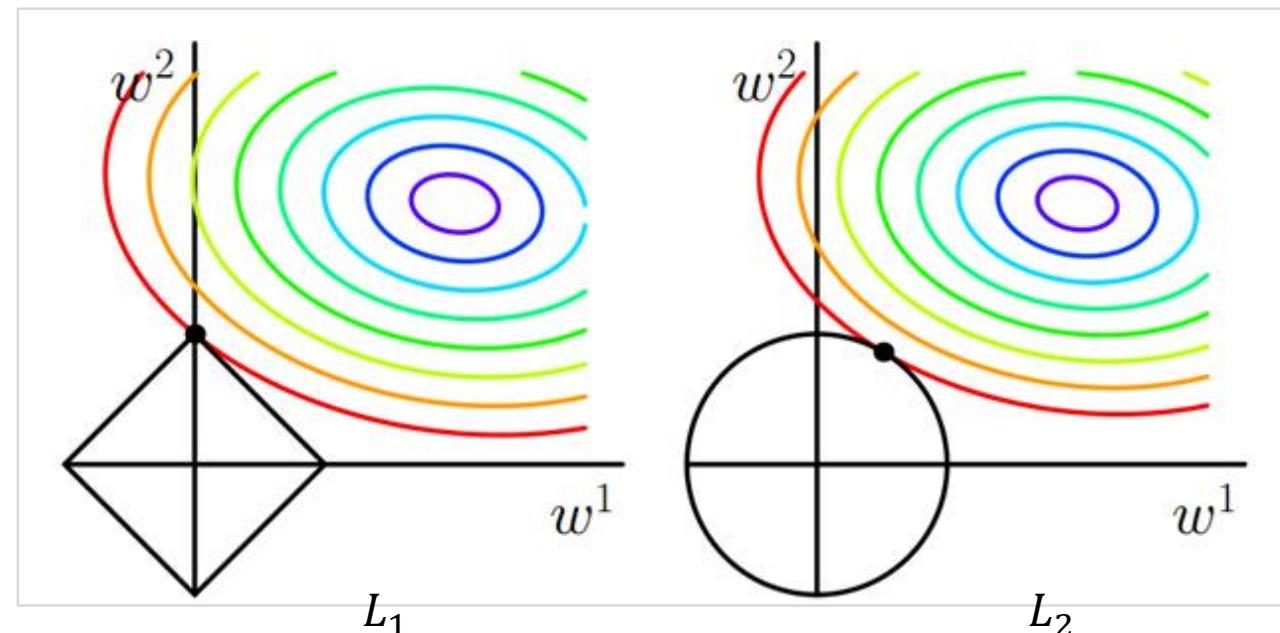
- A parameter optimization method can be inferred using an optimization technology (such as a gradient method):

$$w = (1 - \varepsilon \alpha) \omega - \varepsilon \nabla J(w),$$

- where  $\varepsilon$  is the learning rate. Compared with a common gradient optimization formula, this formula multiplies the parameter by a reduction factor.

# $L_1$ v.s. $L_2$

- The major differences between  $L_2$  and  $L_1$ :
  - According to the preceding analysis,  $L_1$  can generate a more sparse model than  $L_2$ . When the value of parameter  $w$  is small,  $L_1$  regularization can directly reduce the parameter value to 0, which can be used for feature selection.
  - From the perspective of probability, many norm constraints are equivalent to adding prior probability distribution to parameters. In  $L_2$  regularization, the parameter value complies with the Gaussian distribution rule. In  $L_1$  regularization, the parameter value complies with the Laplace distribution rule.

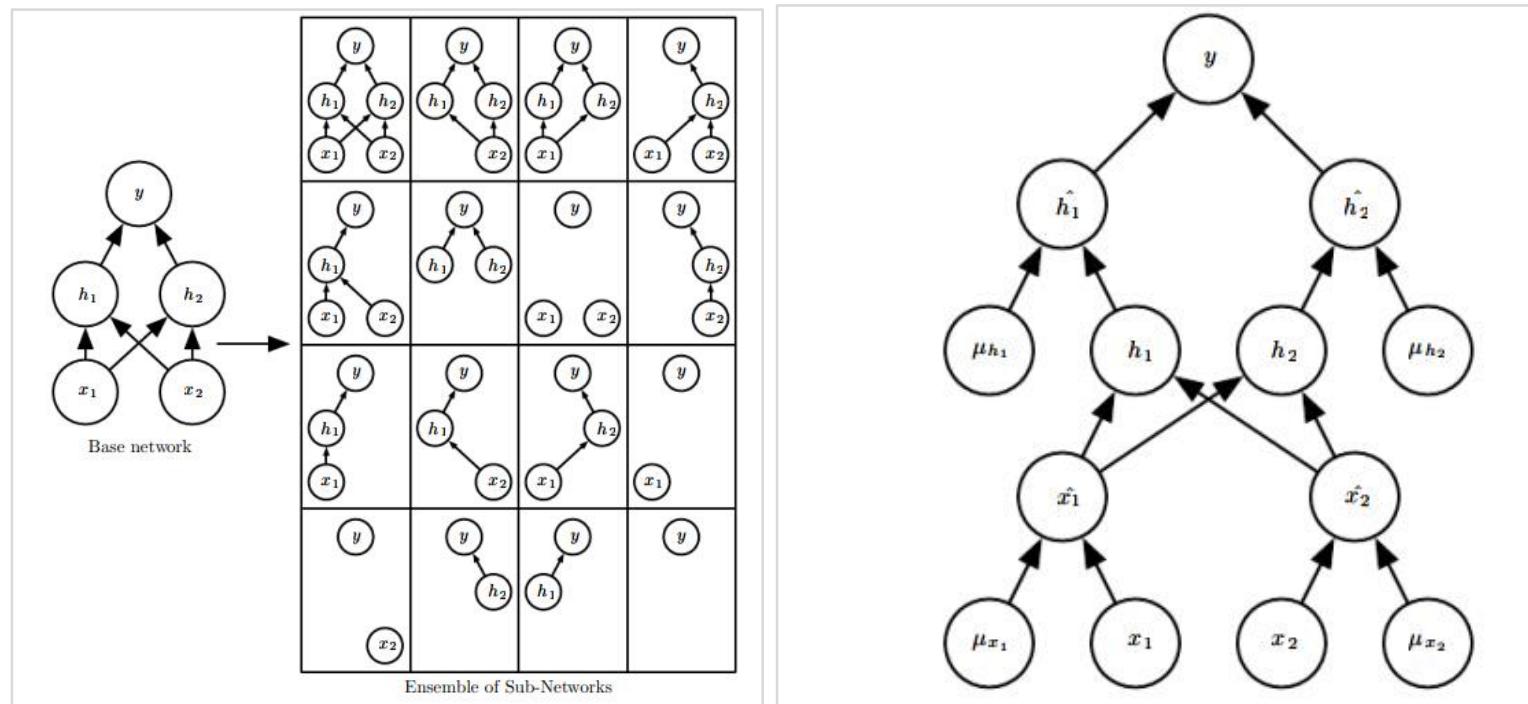


# Dataset Expansion

- The most effective way to prevent over-fitting is to add a training set. A larger training set has a smaller over-fitting probability. Dataset expansion is a time-saving method, but it varies in different fields.
  - A common method in the object recognition field is to rotate or scale images. (The prerequisite to image transformation is that the type of the image cannot be changed through transformation. For example, for handwriting digit recognition, categories 6 and 9 can be easily changed after rotation).
  - Random noise is added to the input data in speech recognition.
  - A common practice of natural language processing (NLP) is replacing words with their synonyms.
  - Noise injection can add noise to the input or to the hidden layer or output layer. For example, for Softmax classification, noise can be added using the label smoothing technology. If noise is added to categories 0 and 1, the corresponding probabilities are changed to  $\frac{\varepsilon}{k}$  and  $1 - \frac{k-1}{k}\varepsilon$  respectively.

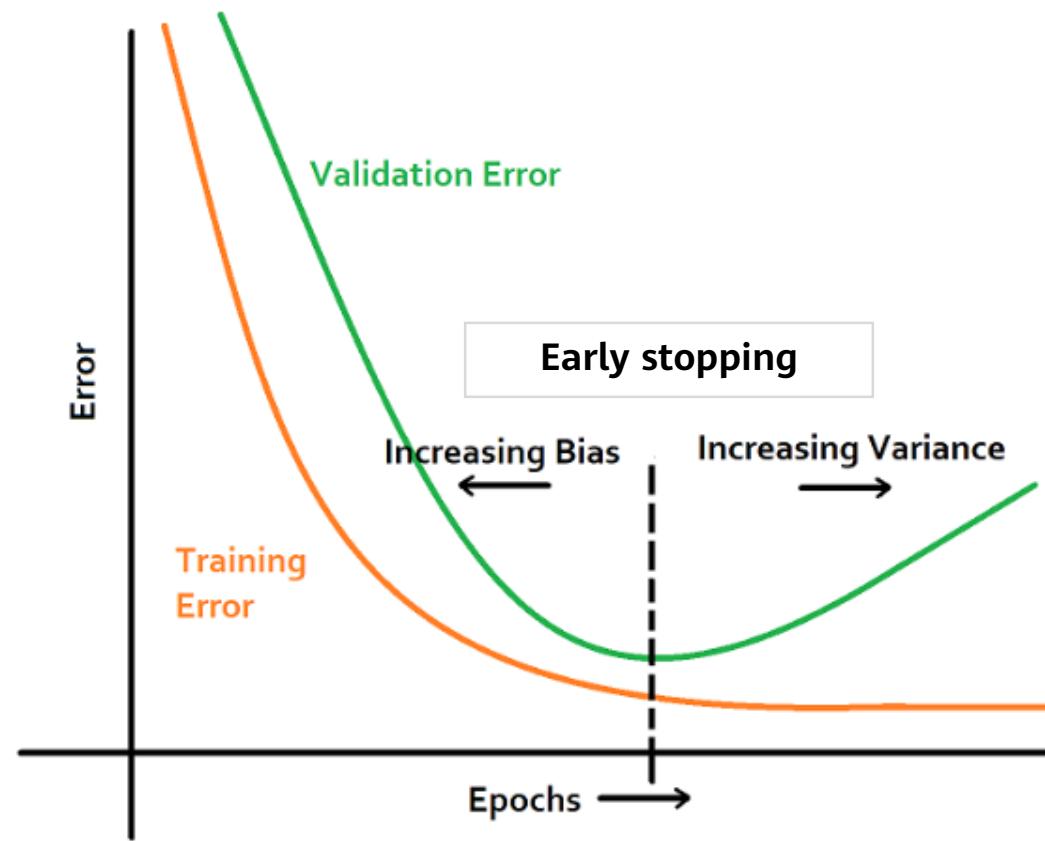
# Dropout

- Dropout is a common and simple regularization method, which has been widely used since 2014. Simply put, Dropout randomly discards some inputs during the training process. In this case, the parameters corresponding to the discarded inputs are not updated. As an integration method, Dropout combines all sub-network results and obtains sub-networks by randomly dropping inputs. See the figures below:



# Early Stopping

- A test on data of the validation set can be inserted during the training. When the data loss of the verification set increases, perform early stopping.



# Contents

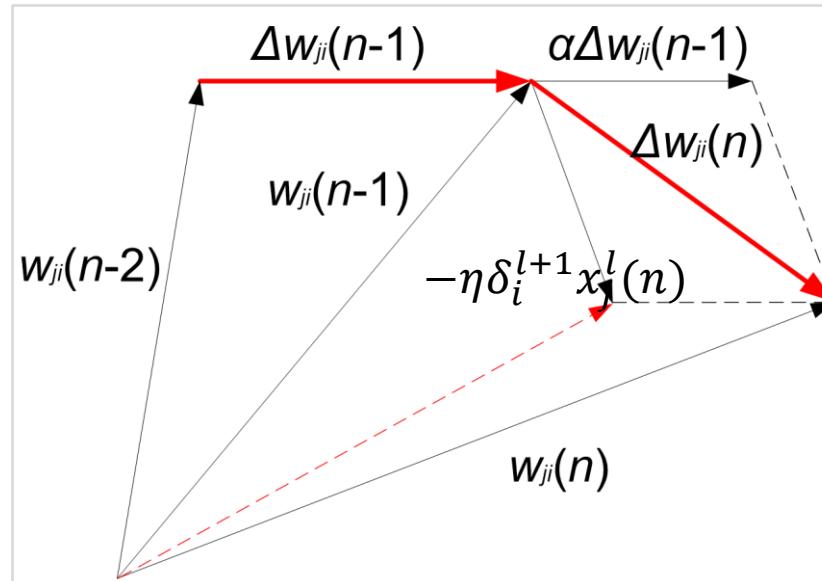
1. Deep Learning Summary
2. Training Rules
3. Activation Function
4. Normalizer
- 5. Optimizer**
6. Types of Neural Networks
7. Common Problems

# Optimizer

- There are various optimized versions of gradient descent algorithms. In object-oriented language implementation, different gradient descent algorithms are often encapsulated into objects called optimizers.
- Purposes of the algorithm optimization include but are not limited to:
  - Accelerating algorithm convergence.
  - Preventing or jumping out of local extreme values.
  - Simplifying manual parameter setting, especially the learning rate (LR).
- Common optimizers: common GD optimizer, momentum optimizer, Nesterov, AdaGrad, AdaDelta, RMSProp, Adam, AdaMax, and Nadam.

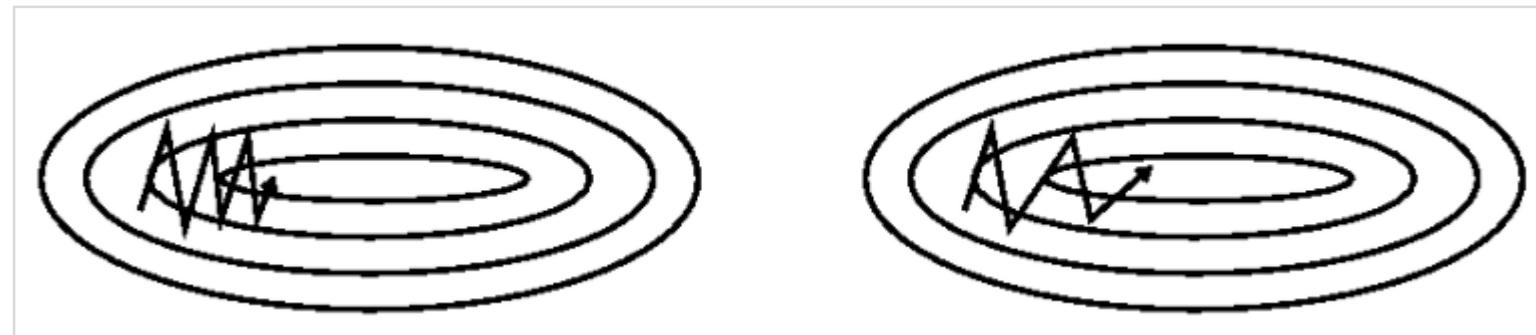
# Momentum Optimizer

- A most basic improvement is to add momentum terms for  $\Delta w_{ji}$ . Assume that the weight correction of the  $n$ -th iteration is  $\Delta w_{ji}(n)$ . The weight correction rule is:
- $\Delta w_{ji}^l(n) = -\eta \delta_i^{l+1} x_j^l(n) + \alpha \Delta w_{ji}^l(n-1)$
- where  $\alpha$  is a constant ( $0 \leq \alpha < 1$ ) called Momentum Coefficient and  $\alpha \Delta w_{ji}(n-1)$  is a momentum term.
- Imagine a small ball rolls down from a random point on the error surface. The introduction of the momentum term is equivalent to giving the small ball inertia.



# Advantages and Disadvantages of Momentum Optimizer

- Advantages:
  - Enhances the stability of the gradient correction direction and reduces mutations.
  - In areas where the gradient direction is stable, the ball rolls faster and faster (there is a speed upper limit because  $\alpha < 1$ ), which helps the ball quickly overshoot the flat area and accelerates convergence.
  - A small ball with inertia is more likely to roll over some narrow local extrema.
- Disadvantages:
  - The learning rate  $\eta$  and momentum  $\alpha$  need to be manually set, which often requires more experiments to determine the appropriate value.



# AdaGrad Optimizer (1)

- The common feature of the random gradient descent algorithm (SGD), small-batch gradient descent algorithm (MBGD), and momentum optimizer is that each parameter is updated with the same LR.
- According to the approach of AdaGrad, different learning rates need to be set for different parameters.

$$g_t = \frac{\partial C(t, o)}{\partial w_t}$$

Gradient calculation

$$r_t = r_{t-1} + g_t^2$$

Square gradient accumulation

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t$$

Computing update

$$w_{t+1} = w_t + \Delta w_t$$

Application update

- $g_t$  indicates the  $t$ -th gradient,  $r$  is a gradient accumulation variable, and the initial value of  $r$  is 0, which **increases continuously**.  $\eta$  indicates the global LR, which needs to be set manually.  $\varepsilon$  is a small constant, and is set to about  $10^{-7}$  for numerical stability.

# AdaGrad Optimizer (2)

- The AdaGrad optimization algorithm shows that the  $r$  continues increasing while the overall learning rate keeps decreasing as the algorithm iterates. This is because we hope LR to decrease as the number of updates increases. In the initial learning phase, we are far away from the optimal solution to the loss function. As the number of updates increases, we are closer to the optimal solution, and therefore LR can decrease.
- Pros:
  - The learning rate is automatically updated. As the number of updates increases, the learning rate decreases.
- Cons:
  - The denominator keeps accumulating so that the learning rate will eventually become very small, and the algorithm will become ineffective.

# RMSProp Optimizer

- The RMSProp optimizer is an improved AdaGrad optimizer. It introduces an attenuation coefficient to ensure a certain attenuation ratio for  $r$  in each round.
- The RMSProp optimizer solves the problem that the AdaGrad optimizer ends the optimization process too early. It is suitable for non-stable target handling and has good effects on the RNN.

$$g_t = \frac{\partial C(t, o)}{\partial w_t}$$

Gradient calculation

$$r_t = \beta r_{t-1} + (1 - \beta) g_t^2$$

Square gradient accumulation

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t$$

Computing update

$$w_{t+1} = w_t + \Delta w_t$$

Application update

- $g_t$  indicates the  $t$ -th gradient,  $r$  is a gradient accumulation variable, and the initial value of  $r$  is 0, which **may not increase and needs to be adjusted using a parameter**.  $\beta$  is the attenuation factor,  $\eta$  indicates the global LR, which needs to be set manually.  $\varepsilon$  is a small constant, and is set to about  $10^{-7}$  for numerical stability.

# Adam Optimizer (1)

- Adaptive Moment Estimation (Adam): Developed based on AdaGrad and AdaDelta, Adam maintains two additional variables  $m_t$  and  $\nu_t$  for each variable to be trained:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$\nu_t = \beta_2 \nu_{t-1} + (1 - \beta_2) g_t^2$$

- Where  $t$  represents the  $t$ -th iteration and  $g_t$  is the calculated gradient.  $m_t$  and  $\nu_t$  are moving averages of the gradient and square gradient. From the statistical perspective,  $m_t$  and  $\nu_t$  are estimates of the first moment (the average value) and the second moment (the uncentered variance) of the gradients respectively, which also explains why the method is so named.

# Adam Optimizer (2)

- If  $m_t$  and  $v_t$  are initialized using the zero vector,  $m_t$  and  $v_t$  are close to 0 during the initial iterations, especially when  $\beta_1$  and  $\beta_2$  are close to 1. To solve this problem, we use  $\hat{m}_t$  and  $\hat{v}_t$ :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

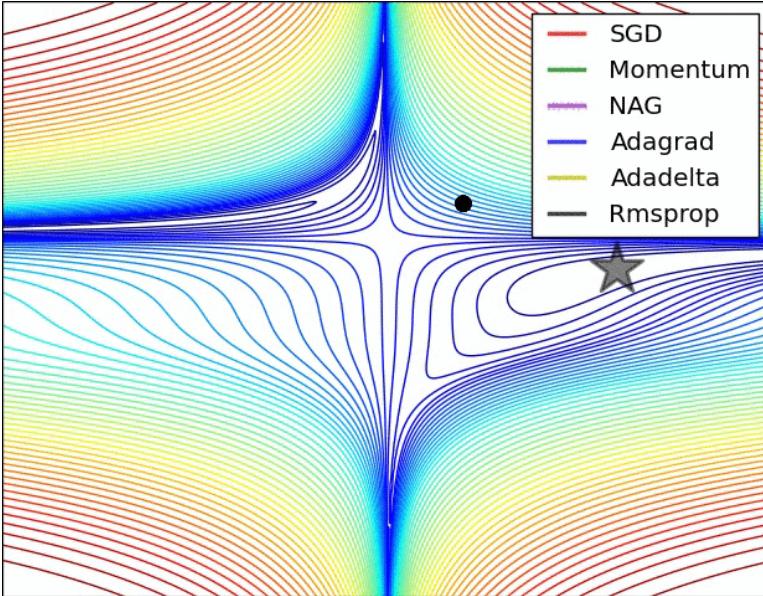
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- The weight update rule of Adam is as follows:

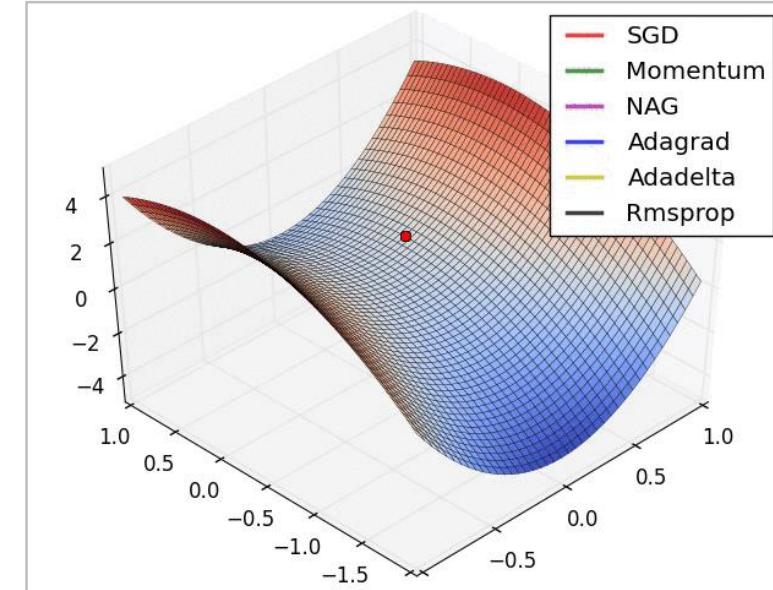
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- Although the rule involves manual setting of  $\eta$ ,  $\beta_1$ , and  $\beta_2$ , the setting is much simpler. According to experiments, the default settings are  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ , and  $\eta = 0.001$ . In practice, Adam will converge quickly. When convergence saturation is reached,  $\eta$  can be reduced. After several times of reduction, a satisfying local extremum will be obtained. Other parameters do not need to be adjusted.

# Optimizer Performance Comparison



Comparison of optimization  
algorithms in contour maps of  
loss functions



Comparison of optimization  
algorithms at the saddle point

# Contents

1. Deep Learning Summary
2. Training Rules
3. Activation Function
4. Normalizer
5. Optimizer
- 6. Types of Neural Networks**
7. Common Problems

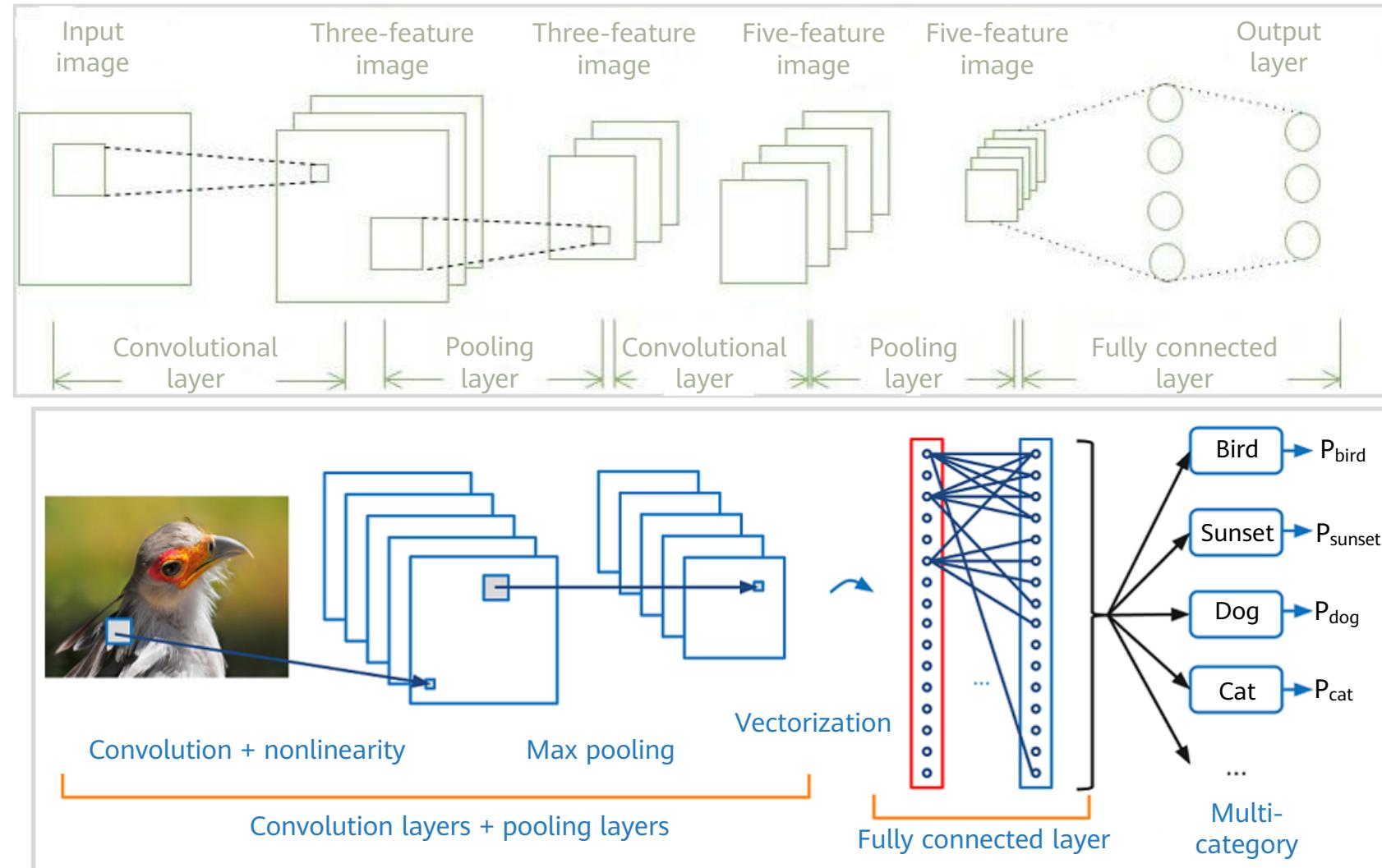
# Convolutional Neural Network

- A convolutional neural network (CNN) is a feedforward neural network. Its artificial neurons may respond to **surrounding units within the coverage range**. CNN excels at image processing. It includes a **convolutional layer**, a **pooling layer**, and a **fully connected layer**.
- In the 1960s, Hubel and Wiesel studied cats' cortex neurons used for local sensitivity and direction selection and found that their unique network structure could simplify feedback neural networks. They then proposed the CNN.
- Now, CNN has become one of the research hotspots in many scientific fields, especially in the pattern classification field. The network is widely used because it can avoid complex pre-processing of images and directly input original images.

# Main Concepts of CNN

- **Local receptive field:** It is generally considered that human perception of the outside world is from local to global. **Spatial correlations among local pixels of an image are closer than those among distant pixels.** Therefore, each neuron does not need to know the global image. It only needs to know the local image. The local information is combined at a higher level to generate global information.
- **Parameter sharing:** One or more filters/kernels may be used to scan input images. Parameters carried by the filters are weights. In a layer scanned by filters, each filter uses the same parameters during weighted computation. Weight sharing means that when each filter scans an entire image, parameters of the filter are fixed.

# Architecture of Convolutional Neural Network



# Single-Filter Calculation (1)

- Description of convolution calculation

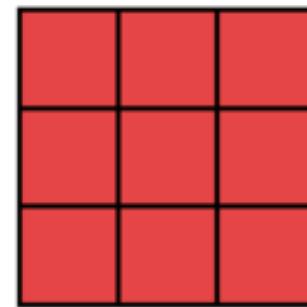
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

image 5\*5

1	0	1
0	1	0
1	0	1

bias=0

filter 3\*3



feature map 3\*3

# Single-Filter Calculation (2)

- Demonstration of the convolution calculation

1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	0	0
0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1	0
0 <small><math>\times 1</math></small>	0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

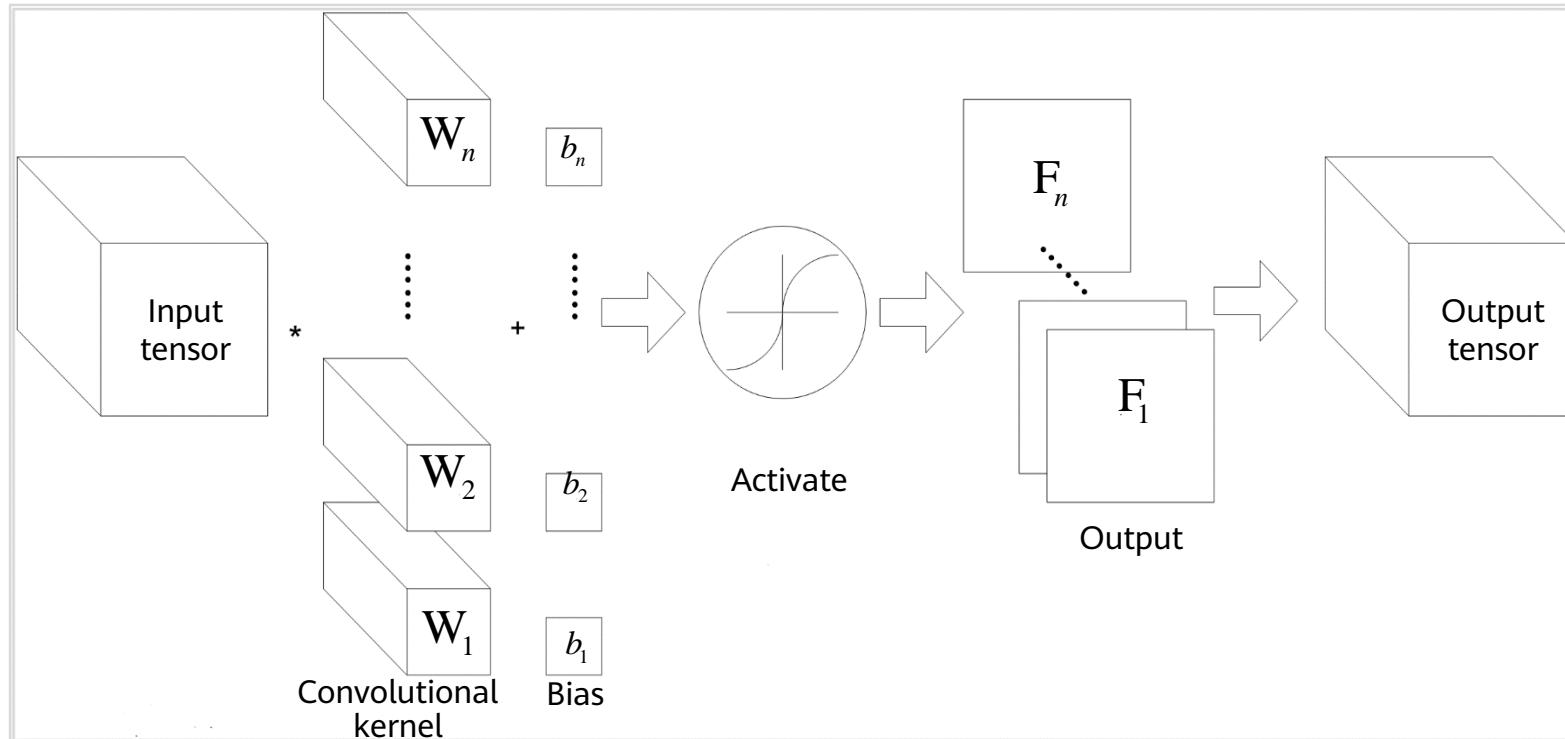
4		

Convolved  
Feature

Han Bingtao, 2017, Convolutional Neural Network

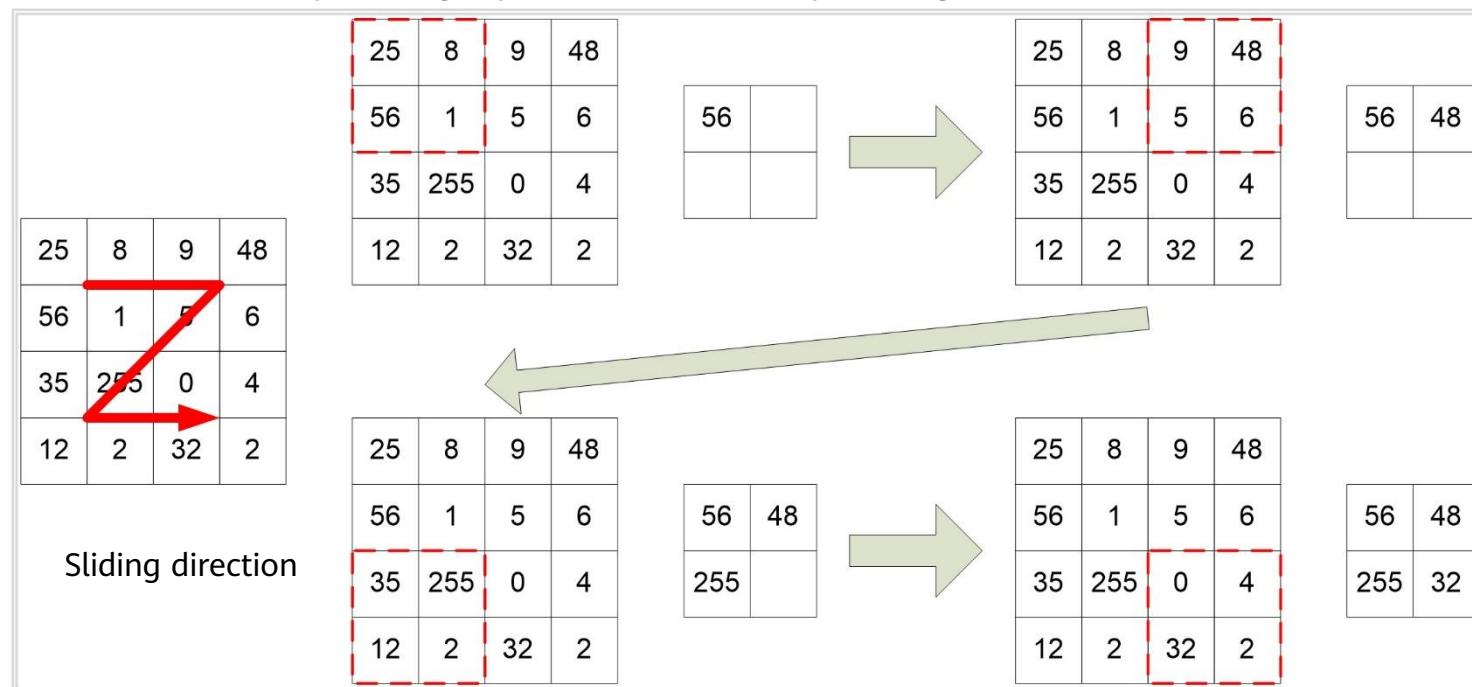
# Convolutional Layer

- The basic architecture of a CNN is multi-channel convolution consisting of multiple single convolutions. The output of the previous layer (or the original image of the first layer) is used as the input of the current layer. It is then convolved with the filter in the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learned. Similar to FCN, after the convolution is complete, the result should be biased and activated through activation functions before being input to the next layer.



# Pooling Layer

- Pooling combines nearby units to reduce the size of the input on the next layer, reducing dimensions. Common pooling includes max pooling and average pooling. When max pooling is used, the maximum value in a small square area is selected as the representative of this area, while the mean value is selected as the representative when average pooling is used. The side of this small area is the pool window size. The following figure shows the max pooling operation whose pooling window size is 2.



# Fully Connected Layer

- The fully connected layer is essentially a classifier. The features extracted on the convolutional layer and pooling layer are straightened and placed at the fully connected layer to output and classify results.
- Generally, the Softmax function is used as the activation function of the final fully connected output layer to combine all local features into global features and calculate the score of each type.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

# Recurrent Neural Network

- The recurrent neural network (RNN) is a neural network that captures dynamic information in sequential data through periodical connections of hidden layer nodes. It can classify sequential data.
- Unlike other forward neural networks, the RNN can keep a context state and even store, learn, and express related information in context windows of any length. Different from traditional neural networks, it is not limited to the space boundary, but also supports time sequences. In other words, there is a side between the hidden layer of the current moment and the hidden layer of the next moment.
- The RNN is widely used in scenarios related to sequences, such as videos consisting of image frames, audio consisting of clips, and sentences consisting of words.

# Recurrent Neural Network Architecture (1)

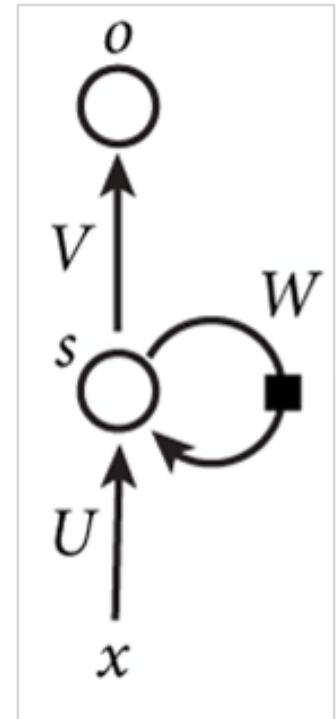
- $X_t$  is the input of the input sequence at time t.
- $S_t$  is the memory unit of the sequence at time t and caches previous information.

$$S_t = \tanh(UX_t + WS_{t-1}).$$

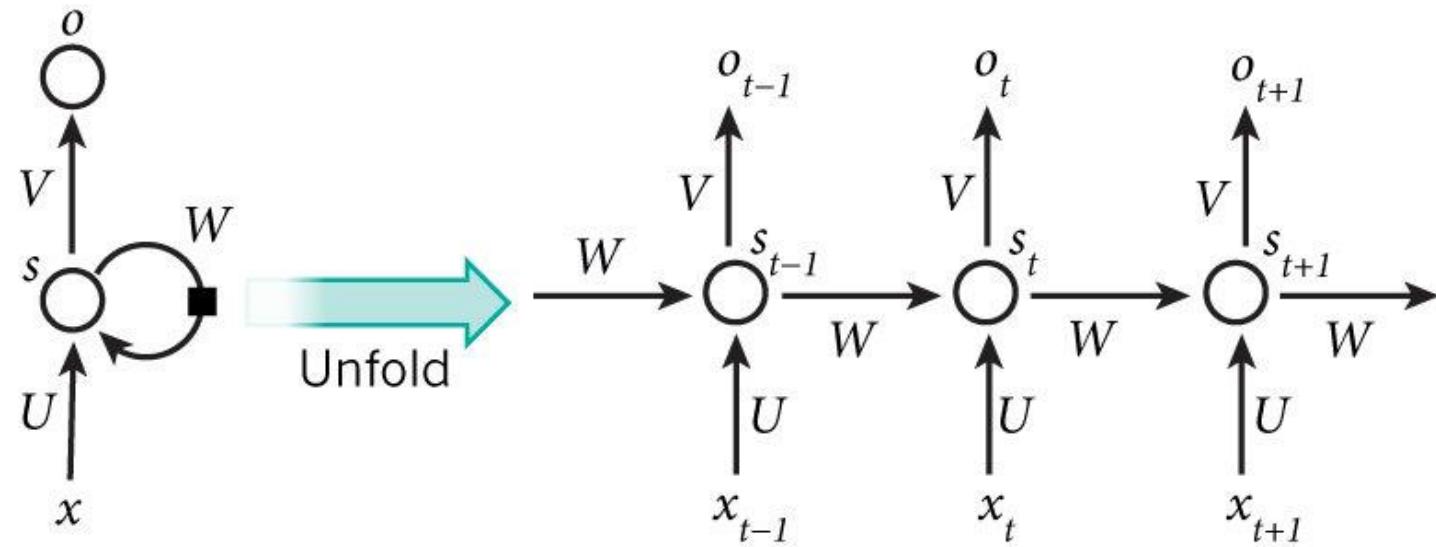
- $O_t$  is the output of the hidden layer of the sequence at time t.

$$O_t = \tanh(VS_t)$$

- $O_t$  after through multiple hidden layers, it can get the final output of the sequence at time t.

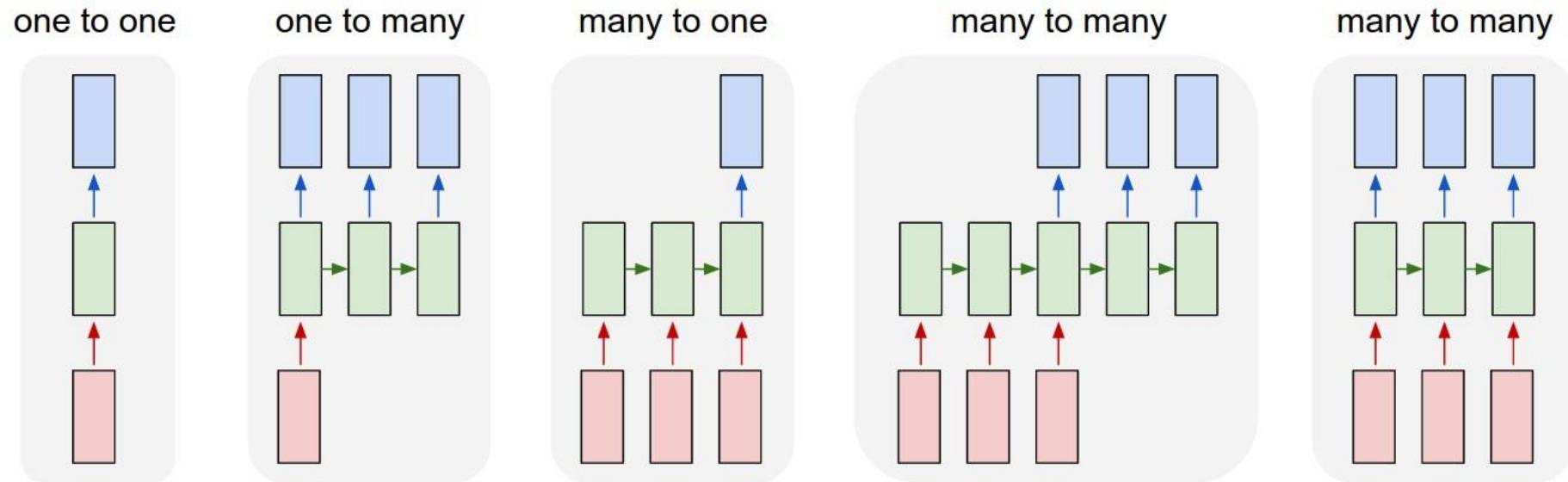


# Recurrent Neural Network Architecture (2)



LeCun, Bengio, and G. Hinton, 2015, A Recurrent Neural Network and the Unfolding in Time of the Computation Involved in Its Forward Computation

# Types of Recurrent Neural Networks



Andrej Karpathy, 2015, The Unreasonable Effectiveness of Recurrent Neural Networks

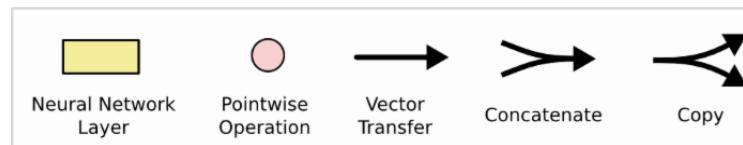
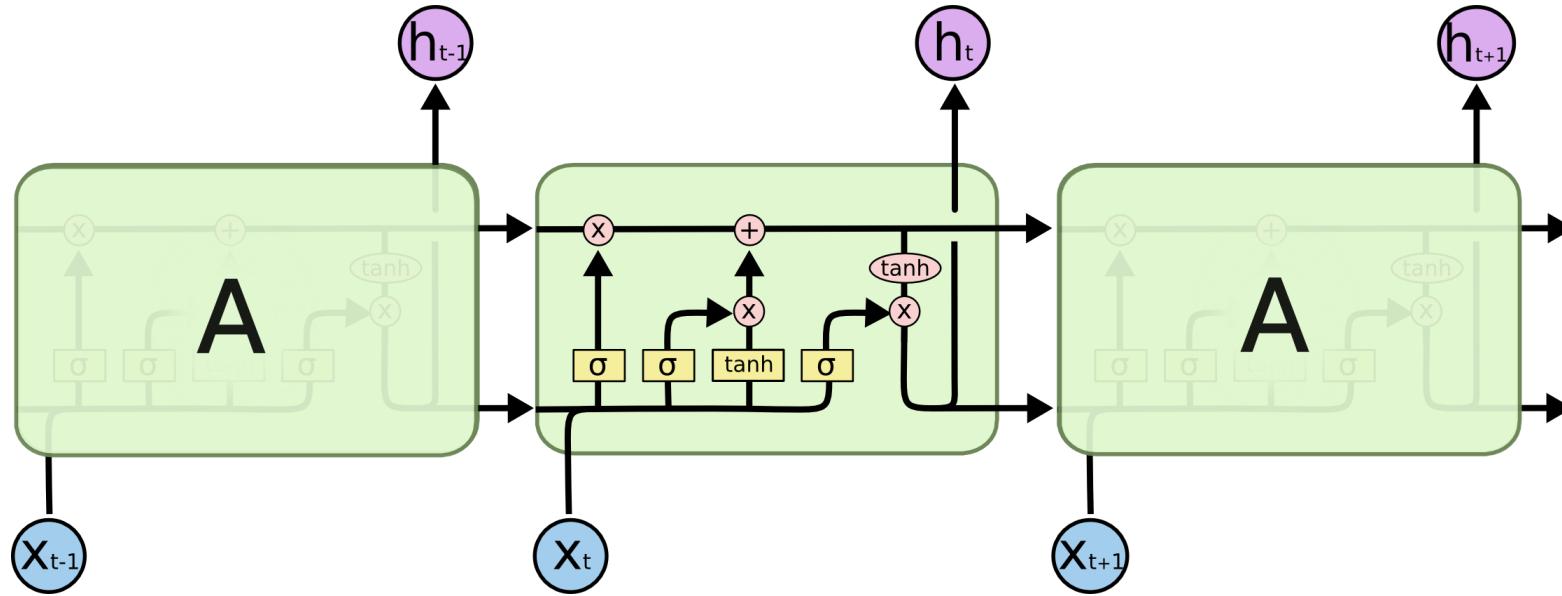
# Backpropagation Through Time (BPTT)

- BPTT:
  - Traditional backpropagation is the extension on the time sequence.
  - There are two sources of errors in the sequence at time of memory unit: first is from the hidden layer output error at time sequence; the second is the error from the memory cell at the next time sequence  $t + 1$ .
  - The longer the time sequence, the more likely the loss of the last time sequence to the gradient of  $w$  in the first time sequence causes the vanishing gradient or exploding gradient problem.
  - The total gradient of weight  $w$  is the accumulation of the gradient of the weight at all time sequence.
- Three steps of BPTT:
  - Computing the output value of each neuron through forward propagation.
  - Computing the error value of each neuron through backpropagation  $\delta_j$ .
  - Computing the gradient of each weight.
- Updating weights using the SGD algorithm.

# Recurrent Neural Network Problem

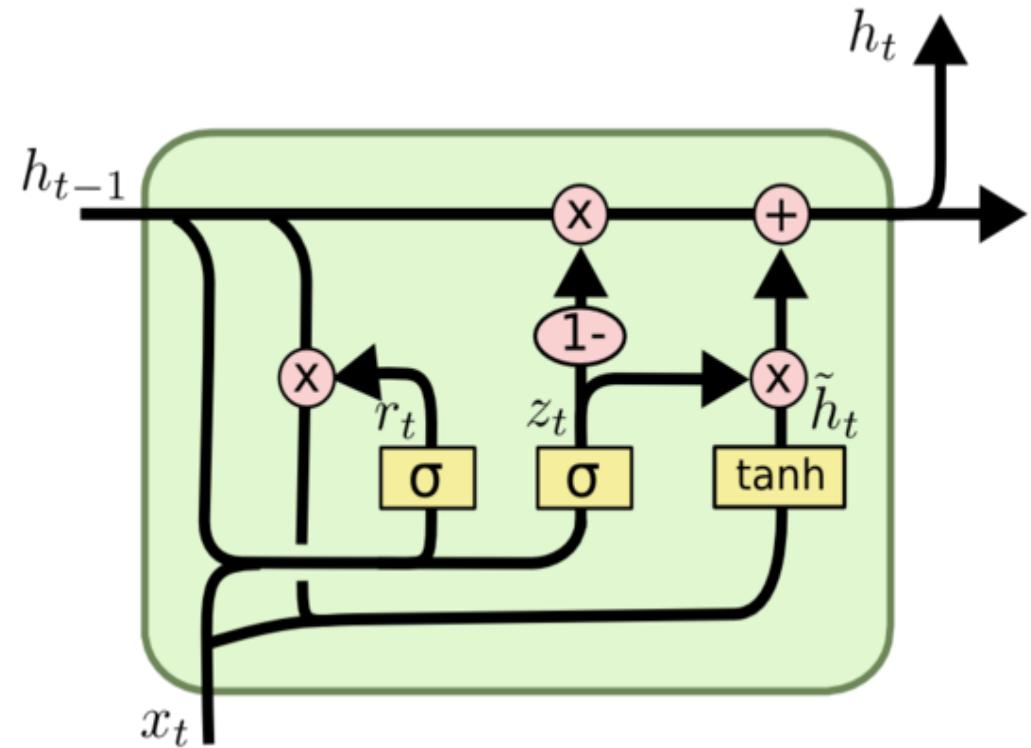
- $S_t = \sigma(UX_t + WS_{t-1})$  is extended on the time sequence.
- $S_t = \sigma\left( UX_t + W\left( \sigma\left( UX_{t-1} + W\left( \sigma\left( UX_{t-2} + W(\dots) \right) \right) \right) \right) \right)$
- Despite that the standard RNN structure solves the problem of information memory, the information attenuates during long-term memory.
- Information needs to be saved long time in many tasks. For example, a hint at the beginning of a speculative fiction may not be answered until the end.
- The RNN may not be able to save information for long due to the limited memory unit capacity.
- We expect that memory units can remember key information.

# Long Short-term Memory Network



Colah, 2015, Understanding LSTMs Networks

# Gated Recurrent Unit (GRU)



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

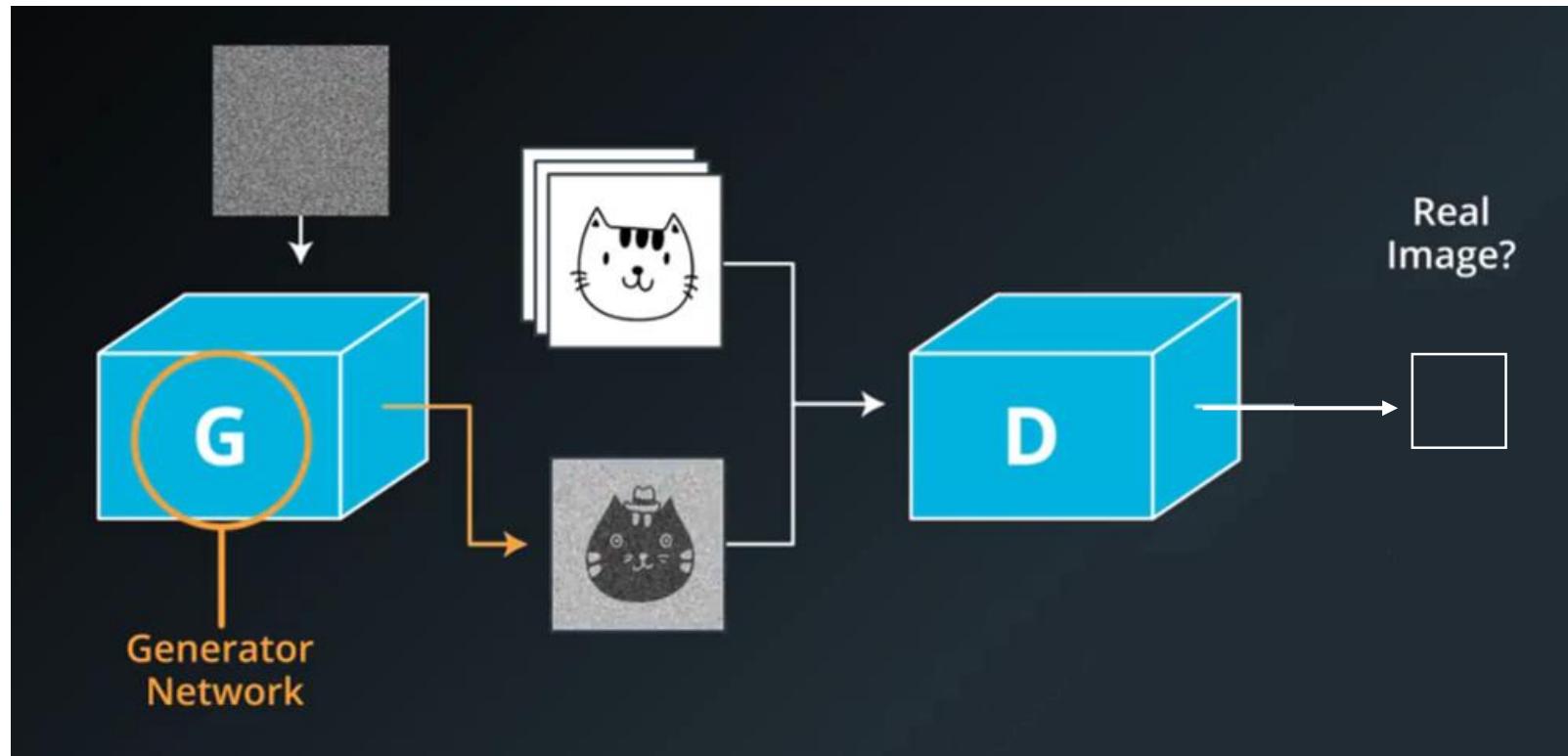
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

# Generative Adversarial Network (GAN)

- Generative Adversarial Network is a framework that trains generator G and discriminator D through the adversarial process. Through the adversarial process, the discriminator can tell whether the sample from the generator is fake or real. GAN adopts a mature BP algorithm.
- (1) Generator G: The input is noise  $z$ , which complies with manually selected prior probability distribution, such as even distribution and Gaussian distribution. The generator adopts the network structure of the multilayer perceptron (MLP), uses maximum likelihood estimation (MLE) parameters to represent the derivable mapping  $G(z)$ , and maps the input space to the sample space.
- (2) Discriminator D: The input is the real sample  $x$  and the fake sample  $G(z)$ , which are tagged as real and fake respectively. The network of the discriminator can use the MLP carrying parameters. The output is the probability  $D(G(z))$  that determines whether the sample is a real or fake sample.
- GAN can be applied to scenarios such as image generation, text generation, speech enhancement, image super-resolution.

# GAN Architecture

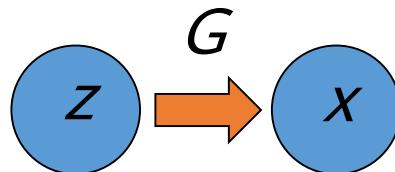
- Generator/Discriminator



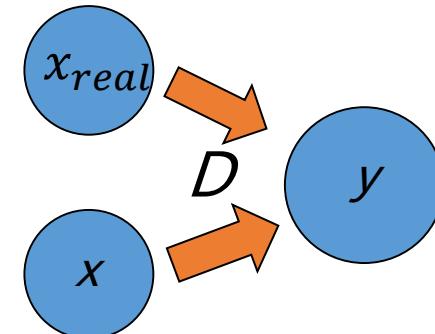
# Generative Model and Discriminative Model

- Generative network
  - Generates sample data
    - Input: Gaussian white noise vector  $z$
    - Output: sample data vector  $x$
- Discriminator network
  - Determines whether sample data is real
    - Input: real sample data  $x_{real}$  and generated sample data  $x = G(z)$
    - Output: probability that determines whether the sample is real

$$x = G(z; \theta^G)$$



$$y = D(x; \theta^D)$$



# Training Rules of GAN

- Optimization objective:
  - Value function

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

- In the early training stage, when the outcome of G is very poor, D determines that the generated sample is fake with high confidence, because the sample is obviously different from training data. In this case,  $\log(1 - D(G(z)))$  is saturated (where the gradient is 0, and iteration cannot be performed). Therefore, we choose to train G only by minimizing  $[-\log(D(G(z)))]$ .

# Contents

1. Deep Learning Summary
2. Training Rules
3. Activation Function
4. Normalizer
5. Optimizer
6. Types of Neural Networks
- 7. Common Problems**

# Data Imbalance (1)

- Problem description: In the dataset consisting of various task categories, the number of samples varies greatly from one category to another. One or more categories in the predicted categories contain very few samples.
- For example, in an image recognition experiment, more than 2,000 categories among a total of 4251 training images contain just one image each. Some of the others have 2-5 images.
- Impacts:
  - Due to the unbalanced number of samples, we cannot get the optimal real-time result because model/algorithm never examines categories with very few samples adequately.
  - Since few observation objects may not be representative for a class, we may fail to obtain adequate samples for verification and test.

# Data Imbalance (2)

## Random undersampling

- Deleting redundant samples in a category

## Random oversampling

- Copying samples

## Synthetic Minority Oversampling Technique

- Sampling
- Merging samples

# Vanishing Gradient and Exploding Gradient Problem (1)

- Vanishing gradient: As network layers increase, the derivative value of backpropagation decreases, which causes a vanishing gradient problem.
- Exploding gradient: As network layers increase, the derivative value of backpropagation increases, which causes an exploding gradient problem.
- Cause:  $y_i = \sigma(z_i) = \sigma(w_i x_i + b_i)$  Where  $\sigma$  is sigmoid function.

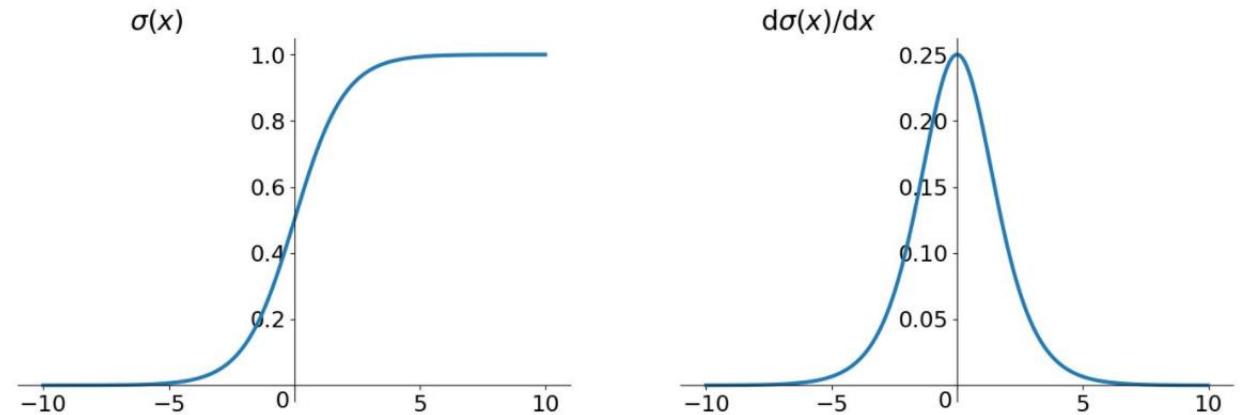


- Backpropagation can be deduced as follows:

$$\begin{aligned}\frac{\partial C}{\partial b_1} &= \frac{\partial C}{\partial y_4} \frac{\partial y_4}{\partial z_4} \frac{\partial z_4}{\partial x_4} \frac{\partial x_4}{\partial z_3} \frac{\partial z_3}{\partial x_3} \frac{\partial x_3}{\partial z_2} \frac{\partial z_2}{\partial x_2} \frac{\partial x_2}{\partial z_1} \frac{\partial z_1}{\partial b_1} \\ &= \frac{\partial C}{\partial y_4} \sigma'(z_4) w_4 \sigma'(z_3) w_3 \sigma'(z_2) w_2 \sigma'(z_1) x\end{aligned}$$

# Vanishing Gradient and Exploding Gradient Problem (2)

- The maximum value of  $\sigma'(x)$  is  $\frac{1}{4}$ :



- However, the network weight  $|w|$  is usually smaller than 1. Therefore,  $|\sigma'(z)w| \leq \frac{1}{4}$ . According to the chain rule, as layers increase, the derivation result  $\frac{\partial C}{\partial b_1}$  decreases, resulting in the vanishing gradient problem.
- When the network weight  $|w|$  is large, resulting in  $|\sigma'(z)w| > 1$ , the exploding gradient problem occurs.
- Solution: For example, gradient clipping is used to alleviate the exploding gradient problem, ReLU activation function and LSTM are used to alleviate the vanishing gradient problem.

# Overfitting

- Problem description: The model performs well in the training set, but badly in the test set.
- Root cause: There are too many feature dimensions, model assumptions, and parameters, too much noise, but very few training data. As a result, the fitting function perfectly predicts the training set, while the prediction result of the test set of new data is poor. Training data is over-fitted without considering generalization capabilities.
- Solution: For example, data augmentation, regularization, early stopping, and dropout

# Summary

- This chapter describes the definition and development of neural networks, perceptrons and their training rules, common types of neural networks (CNN, RNN, and GAN), and the Common Problems of neural networks in AI engineering and solutions.

# Quiz

1. (True or false) Compared with the recurrent neural network, the convolutional neural network is more suitable for image recognition. ( )
  - A. True
  - B. False
2. (True or false) GAN is a deep learning model, which is one of the most promising methods for unsupervised learning of complex distribution in recent years. ( )
  - A. True
  - B. False

# Quiz

3. (Single-choice) There are many types of deep learning neural networks. Which of the following is not a deep learning neural network? ( )
- A. CNN
  - B. RNN
  - C. LSTM
  - D. Logistic
4. (Multi-choice) There are many important "components" in the convolutional neural network architecture. Which of the following are the convolutional neural network "components"? ( )
- A. Activation function
  - B. Convolutional kernel
  - C. Pooling
  - D. Fully connected layer

# Recommendations

---

- Online learning website
  - <https://e.huawei.com/cn/talent/#/home>
- Huawei Knowledge Base
  - <https://support.huawei.com/enterprise/servicecenter?lang=zh>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Mainstream Development Frameworks in the Industry



# Foreword

---

- This chapter describes:
  - Definition of deep learning framework and its advantages, and two mainstream deep learning frameworks PyTorch and TensorFlow
  - Basic operations and common modules of TensorFlow 2.x (by focusing on code)
  - MNIST handwritten digit recognition experiment performed based on TensorFlow for deeply understanding and getting familiar with a deep learning modeling process

# Objectives

On completion of this course, you will be able to:

- Describe a deep learning framework.
- Know mainstream deep learning frameworks.
- Know the features of PyTorch.
- Know the features of TensorFlow.
- Differentiate between TensorFlow 1.x and 2.x.
- Master the basic syntax and common modules of TensorFlow 2.x.
- Master the process of an MNIST handwritten digit recognition experiment.

# Contents

## **1. Mainstream Development Frameworks**

- Deep Learning Framework
  - PyTorch
  - TensorFlow

## 2. TensorFlow 2.x Basics

## 3. Common Modules of TensorFlow 2.x

## 4. Basic Steps of Deep Learning Development

# Deep Learning Framework

- A deep learning framework is an interface, library or a tool which allows us to build deep learning models more easily and quickly, without getting into the details of underlying algorithms. A deep learning framework can be regarded as a set of building blocks. Each component in the building blocks is a model or algorithm. Therefore, developers can use components to assemble models that meet requirements, and do not need to start from scratch.
- The emergence of deep learning frameworks lowers the requirements for developers. Developers no longer need to compile code starting from complex neural networks and back-propagation algorithms. Instead, they can use existing models to configure parameters as required, where the model parameters are automatically trained. Moreover, they can add self-defined network layers to the existing models, or select required classifiers and optimization algorithms directly by invoking existing code.

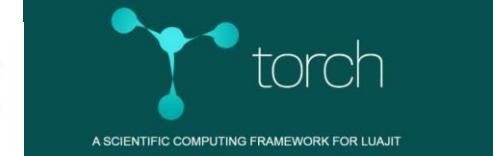
theano

Caffe

dmlc  
mxnet

DL4J Deep Learning for Java

P Y T  ORCH



[M]<sup>s</sup>  
MindSpore



# Contents

## **1. Mainstream Development Frameworks**

- Deep Learning Framework
  - PyTorch
  - TensorFlow

## 2. TensorFlow 2.x Basics

## 3. Common Modules of TensorFlow 2.x

## 4. Basic Steps of Deep Learning Development

# PyTorch

- PyTorch is a Python-based machine learning computing framework developed by Facebook. It is developed based on Torch, a scientific computing framework supported by a large number of machine learning algorithms. Torch is a tensor operation library similar to NumPy, featured by high flexibility, but is less popular because it uses the programming language Lua. This is why PyTorch is developed.
- In addition to Facebook, institutes such as Twitter, GMU, and Salesforce also use PyTorch.



Image source: <http://PyTorch123.com/FirstSection/PyTorchIntro/>

# Features of PyTorch

- **Python first:** PyTorch does not simply bind Python to a C++ framework. PyTorch directly supports Python access at a fine grain. Developers can use PyTorch as easily as using NumPy or SciPy. This not only lowers the threshold for understanding Python, but also ensures that the code is basically consistent with the native Python implementation.
- **Dynamic neural network:** Many mainstream frameworks such as TensorFlow 1.x do not support this feature. To run TensorFlow 1.x, developers must create static computational graphs in advance, and run the **feed** and **run** commands to repeatedly execute the created graphs. In contrast, PyTorch with this feature is free from such complexity, and PyTorch programs can dynamically build/adjust computational graphs during execution.
- **Easy to debug:** PyTorch can generate dynamic graphs during execution. Developers can stop an interpreter in a debugger and view output of a specific node.
- PyTorch provides tensors that support CPUs and GPUs, greatly accelerating computing.

# Contents

## **1. Mainstream Development Frameworks**

- Deep Learning Framework
- PyTorch
- TensorFlow

## 2. TensorFlow 2.x Basics

## 3. Common Modules of TensorFlow 2.x

## 4. Basic Steps of Deep Learning Development

# TensorFlow

- TensorFlow is Google's second-generation open-source software library for digital computing. The TensorFlow computing framework supports various deep learning algorithms and multiple computing platforms, ensuring high system stability.



Image source: <https://www.TensorFlow.org/>

# Features of TensorFlow

Scalability

Multi-lingual

GPU

Multi-platform

Powerful  
computing

Distributed

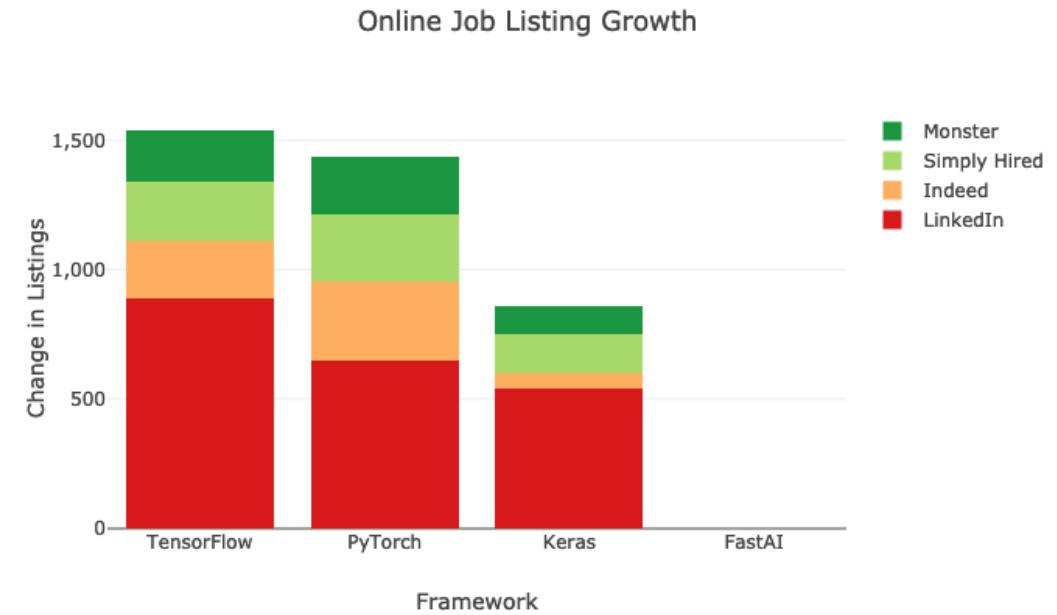


# TensorFlow - Distributed

- TensorFlow can run on different computers:
  - From smartphones to computer clusters, to generate desired training models.
- Currently, supported native distributed deep learning frameworks include only TensorFlow, CNTK, Deeplearning4J, and MXNet.
- When a single GPU is used, most deep learning frameworks rely on cuDNN, and therefore support almost the same training speed, provided that the hardware computing capabilities or allocated memories slightly differ. However, for large-scale deep learning, massive data makes it difficult for the single GPU to complete training in a limited time. To handle such cases, TensorFlow enables distributed training.

# Why TensorFlow?

- TensorFlow is considered as one of the best libraries for neural networks, and can reduce difficulty in deep learning development. In addition, as it is open-source, it can be conveniently maintained and updated, thus the efficiency of development can be improved.
- Keras, ranking third in the number of stars on GitHub, is packaged into an advanced API of TensorFlow 2.0, which makes TensorFlow 2.x more flexible, and easier to debug.



Demand on the  
recruitment market

# TensorFlow 2.x vs. TensorFlow 1.x

- Disadvantages of TensorFlow 1.0:
  - After a tensor is created in TensorFlow 1.0, the result cannot be returned directly. To obtain the result, the session mechanism needs to be created, which includes the concept of graph, and code cannot run without `session.run`. This style is more like the hardware programming language VHDL.
  - Compared with some simple frameworks such as PyTorch, TensorFlow 1.0 adds the session and graph concepts, which are inconvenient for users.
  - It is complex to debug TensorFlow 1.0, and its APIs are disordered, making it difficult for beginners. Learners will come across many difficulties in using TensorFlow 1.0 even after gaining the basic knowledge. As a result, many researchers have turned to PyTorch.

# TensorFlow 2.x vs. TensorFlow 1.x

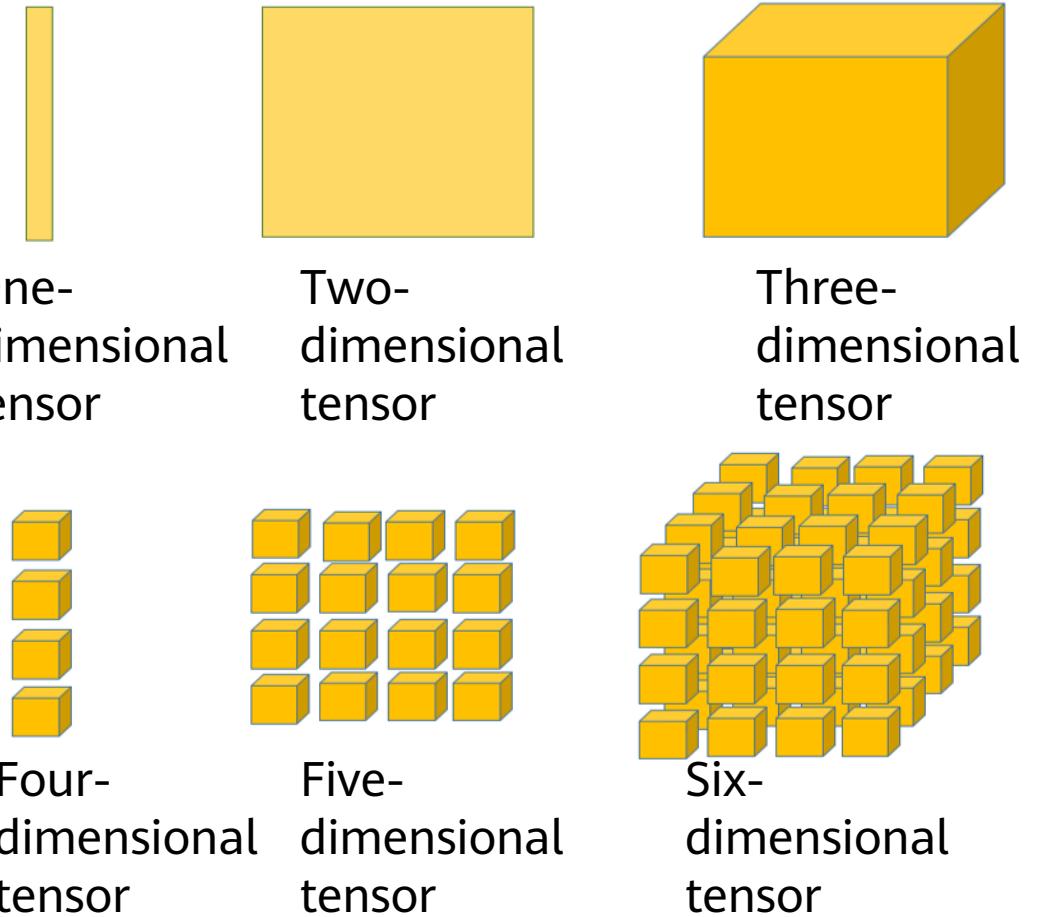
- Features of TensorFlow 2.x:
  - Advanced API Keras:
    - Easy to use: The graph and session mechanisms are removed. What you see is what you get, just like Python and PyTorch.
  - Major improvements:
    - The core function of TensorFlow 2.x is the dynamic graph mechanism called eager execution. It allows users to compile and debug models like normal programs, making TensorFlow easier to learn and use.
    - Multiple platforms and languages are supported, and compatibility between components can be improved via standardization on exchange formats and alignment of APIs.
    - Deprecated APIs are deleted and duplicate APIs are reduced to avoid confusion.
    - Compatibility and continuity: TensorFlow 2.x provides a module enabling compatibility with TensorFlow 1.x.
    - The tf.contrib module is removed. Maintained modules are moved to separate repositories. Unused and unmaintained modules are removed.

# Contents

1. Mainstream Development Frameworks
- 2. TensorFlow 2.x Basics**
3. Common Modules of TensorFlow 2.x
4. Basic Steps of Deep Learning Development

# Tensors

- Tensors are the most basic data structures in TensorFlow. All data is encapsulated in tensors.
- Tensor: a multidimensional array
  - A scalar is a rank-0 tensor. A vector is a rank-1 tensor. A matrix is a rank-2 tensor.
- In TensorFlow, tensors are classified into:
  - Constant tensors
  - Variable tensors



# Basic Operations of TensorFlow 2.x

- The following describes common APIs in TensorFlow by focusing on code. The main content is as follows:
  - Methods for creating constants and variables
  - Tensor slicing and indexing
  - Dimension changes of tensors
  - Arithmetic operations on tensors
  - Tensor concatenation and splitting
  - Tensor sorting

# Eager Execution Mode of TensorFlow 2.x

- Static graph: TensorFlow 1.x using static graphs (graph mode) separates computation definition and execution by using computational graphs. This is a declarative programming model. In graph mode, developers need to build a computational graph, start a session, and then input data to obtain an execution result.
- Static graphs are advantageous in distributed training, performance optimization, and deployment, but inconvenient for debugging. Executing a static graph is similar to invoking a compiled C language program, and internal debugging cannot be performed in this case. Therefore, eager execution based on dynamic computational graphs emerges.
- Eager execution is a command-based programming method, which is the same as native Python. A result is returned immediately after an operation is performed.

# AutoGraph

- Eager execution is enabled in TensorFlow 2.x by default. Eager execution is intuitive and flexible for users (easier and faster to run a one-time operation), but may compromise performance and deployability.
- To achieve optimal performance and make a model deployable anywhere, you can run `@tf.function` to add a decorator to build a graph from a program, making Python code more efficient.
- `tf.function` can build a TensorFlow operation in the function into a graph. In this way, this function can be executed in graph mode. Such practice can be considered as encapsulating the function as a TensorFlow operation of a graph.

# Contents

1. Mainstream Development Frameworks
2. TensorFlow 2.x Basics
- 3. Common Modules of TensorFlow 2.x**
4. Basic Steps of Deep Learning Development

# Common Modules of TensorFlow 2.x (1)

- `tf`: Functions in the `tf` module are used to perform common arithmetic operations, such as `tf.abs` (calculating an absolute value), `tf.add` (adding elements one by one), and `tf.concat` (concatenating tensors). Most operations in this module can be performed by NumPy.
- `tf.errors`: error type module of TensorFlow
- `tf.data`: implements operations on datasets.
  - Input pipes created by `tf.data` are used to read training data. In addition, data can be easily input from memories (such as NumPy).
- `tf.distributions`: implements various statistical distributions.
  - The functions in this module are used to implement various statistical distributions, such as Bernoulli distribution, uniform distribution, and Gaussian distribution.

# Common Modules of TensorFlow 2.x (2)

- `tf.io.gfile`: implements operations on files.
  - Functions in this module can be used to perform file I/O operations, copy files, and rename files.
- `tf.image`: implements operations on images.
  - Functions in this module include image processing functions. This module is similar to OpenCV, and provides functions related to image luminance, saturation, phase inversion, cropping, resizing, image format conversion (RGB to HSV, YUV, YIQ, or gray), rotation, and sobel edge detection. This module is equivalent to a small image processing package of OpenCV.
- `tf.keras`: a Python API for invoking Keras tools.
  - This is a large module that enables various network operations.

# Keras Interface

- TensorFlow 2.x recommends Keras for network building. Common neural networks are included in **Keras.layers**.
- Keras is a high-level API used to build and train deep learning models. It can be used for rapid prototype design, advanced research, and production. It has the following three advantages:
  - Easy to use  
Keras provides simple and consistent GUIs optimized for common cases. It provides practical and clear feedback on user errors.
  - Modular and composable  
You can build Keras models by connecting configurable building blocks together, with little restriction.
  - Easy to extend  
You can customize building blocks to express new research ideas, create layers and loss functions, and develop advanced models.

# Common Keras Methods and Interfaces

- The following describes common methods and interfaces of tf.keras by focusing on code. The main content is as follows:
  - Dataset processing: datasets and preprocessing
  - Neural network model creation: Sequential, Model, Layers...
  - Network compilation: compile, Losses, Metrics, and Optimizers
  - Network training and evaluation: fit, fit\_generator, and evaluate

# Contents

1. Mainstream Development Frameworks
2. TensorFlow 2.x Basics
3. Common Modules of TensorFlow 2.x
- 4. Basic Steps of Deep Learning Development**

# TensorFlow Environment Setup in Windows 10

- Environment setup in Windows 10:
  - Operating system: Windows 10
  - pip software built in Anaconda 3 (adapting to Python 3)
  - TensorFlow installation:
    - Open Anaconda Prompt and run the pip command to install TensorFlow.
    - Run pip install TensorFlow in the command line interface.

```
(base) C:\Users\ThinkPad>pip install tensorflow
Requirement already satisfied: tensorflow in d:\vs\anaconda3_64\lib\site-packages (1.14.0)
Requirement already satisfied: astor>=0.6.0 in c:\users\thinkpad\appdata\roaming\python\python36\site-packages (from tensorflow) (0.8.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\users\thinkpad\appdata\roaming\python\python36\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: six>=1.10.0 in d:\vs\anaconda3_64\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: keras-applications>=1.0.6 in c:\users\thinkpad\appdata\roaming\python\python36\site-packages (from tensorflow) (1.0.8)
Requirement already satisfied: grpcio>=1.8.6 in d:\vs\anaconda3_64\lib\site-packages (from tensorflow) (1.23.0)
Requirement already satisfied: gast>=0.2.0 in d:\vs\anaconda3_64\lib\site-packages (from tensorflow) (0.3.2)
Requirement already satisfied: tensorflow-estimator<1.15.0rc0,>=1.14.0rc0 in c:\users\thinkpad\appdata\roaming\python\python36\site-packages (from tensorflow) (1.14.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\thinkpad\appdata\roaming\python\python36\site-packages (from tensorflow) (1.1.0)
```

# TensorFlow Environment Setup in Ubuntu/Linux

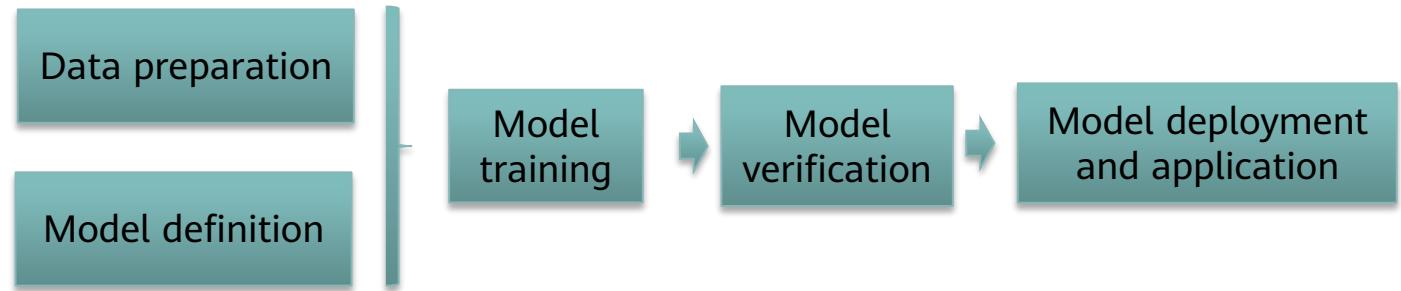
- The simplest way for installing TensorFlow in Linux is to run the pip command.

```
czy@czy-System-Product-Name:~$ pip install tensorflow==2.0.0-alpha0
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting tensorflow==2.0.0-alpha0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/29/39/f99185d3913
afcfe1dcdb0629c2ffc4ecfb0e4c14ca210d620e56c/tensorflow-2.0.0a0-cp36-cp36m-
nux1_x86_64.whl (79.9MB)
    37% |██████████| 29.8MB 98.5MB/s eta 0:00:01
```

- pip command: pip install TensorFlow==2.1.0

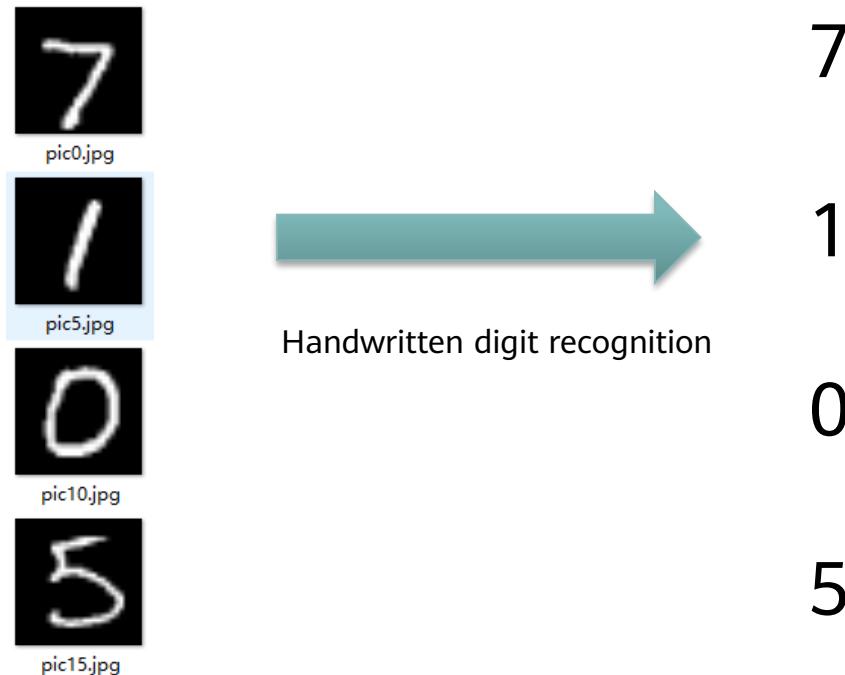
# TensorFlow Development Process

- Data preparation
  - Data exploration
  - Data processing
- Network construction
  - Defining a network structure.
  - Defining loss functions, selecting optimizers, and defining model evaluation indicators.
- Model training and verification
- Model saving
- Model restoration and invoking



# Project Description

- Handwritten digit recognition is a common image recognition task where computers recognize text in handwriting images. Different from printed fonts, handwriting of different people has different sizes and styles, making it difficult for computers to recognize handwriting. This project applies deep learning and TensorFlow tools to train and build models based on the MNIST handwriting dataset.



# Data Preparation

- MNIST datasets
  - Download the MNIST datasets from <http://yann.lecun.com/exdb/mnist/>.
  - The MNIST datasets consist of a training set and a test set.
    - Training set: 60,000 handwriting images and corresponding labels
    - Test set: 10,000 handwriting images and corresponding labels

**Examples**



**Corresponding  
labels**

[0,0,0,0,0,  
1,0,0,0,0]

[0,0,0,0,0,  
0,0,0,0,1]

[0,0,0,0,0,  
0,0,1,0,0]

[0,0,0,1,0,  
0,0,0,0,0]

[0,0,0,0,1,  
0,0,0,0,0]

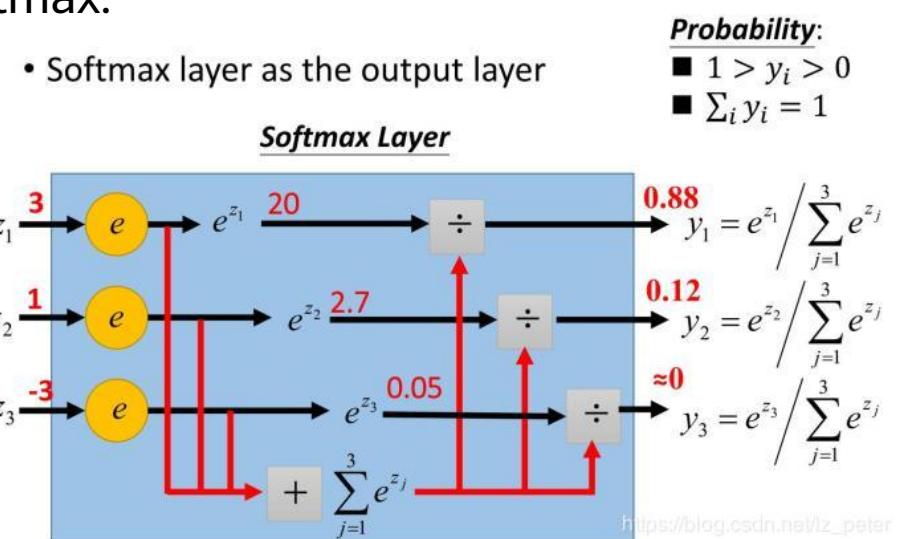
# Network Structure Definition (1)

- Softmax regression model

$$evidence_i = \sum_j W_{i,j} x_j + b_i$$

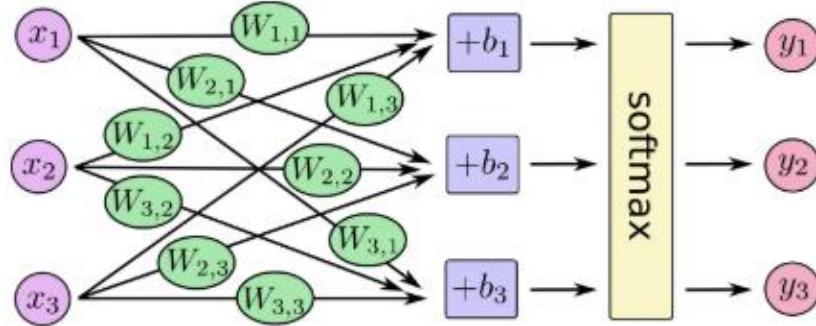
$$y = \text{softmax}(evidence)$$

- The softmax function is also called normalized exponential function. It is a derivative of the binary classification function sigmoid in terms of multi-class classification. The following figure shows the calculation method of softmax.



# Network Structure Definition (2)

- The process of model establishment is the core process of network structure definition.
- The network operation process defines how model output is calculated based on input.



- Matrix multiplication and vector addition are used to express the calculation process of softmax.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left( \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

# Network Structure Definition (3)

- TensorFlow-based softmax regression model

```
## import tensorflow
import tensorflow as tf
##define input variables with operator symbol variables.
''' we use a variable to feed data into the graph through the placeholders X. Each input
image is flattened into a 784-dimensional vector. In this case, the shape of the tensor is
[None, 784], None indicates can be of any length.'''
X = tf.placeholder(tf.float32,[None,784])
''' The variable that can be modified is used to indicate the weight w and bias b. The initial
values are set to 0.'''
w = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
''' If tf.matmul(x, w) is used to indicate that x is multiplied by w, the Soft regression
equation is y = softmax(wx+b)'''
y = tf.nn.softmax(tf.matmul(x,w)+b)
```

# Network Compilation

- Model compilation involves the following two parts:
  - Loss function selection
- In machine learning/deep learning, an indicator needs to be defined to indicate whether a model is proper. This indicator is called cost or loss, and is minimized as far as possible. In this project, the cross entropy loss function is used.
  - Gradient descent method
- A loss function is constructed for an original model needs to be optimized by using an optimization algorithm, to find optimal parameters and further minimize a value of the loss function. Among optimization algorithms for solving machine learning parameters, the gradient descent-based optimization algorithm (Gradient Descent) is usually used.

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss=tf.keras.losses.categorical_crossentropy,
              metrics=[tf.keras.metrics.categorical_accuracy])
```

# Model Training

- Training process:
  - All training data is trained through batch iteration or full iteration. In the experiment, all data is trained five times.
  - In TensorFlow, `model.fit` is used for training, where epoch indicates the number of training iterations.

```
model.fit(mnist.train.images, mnist.train.labels, epochs=5)

Epoch 1/5
55000/55000 [=====] - 4s 74us/sample - loss: 0.3043 - categorical_accuracy: 0.9110
Epoch 2/5
55000/55000 [=====] - 4s 73us/sample - loss: 0.1460 - categorical_accuracy: 0.9569
Epoch 3/5
55000/55000 [=====] - 4s 79us/sample - loss: 0.1104 - categorical_accuracy: 0.9669
Epoch 4/5
55000/55000 [=====] - 4s 74us/sample - loss: 0.0881 - categorical_accuracy: 0.9722
Epoch 5/5
55000/55000 [=====] - 4s 73us/sample - loss: 0.0767 - categorical_accuracy: 0.9760
```

# Model Evaluation

- You can test the model using the test set, compare predicted results with actual ones, and find correctly predicted labels, to calculate the accuracy of the test set.

```
model.evaluate(mnist.test.images, mnist.test.labels)  
10000/10000 [=====] - 0s 42us/sample - loss: 0.0779 - categorical_accuracy: 0.9764
```

[0.07786676207473502, 0.9764]

Loss value

Accuracy

# Quiz

1. In TensorFlow 2.x, eager execution is enabled by default. ( )
  - A. True
  - B. False
2. Which of the following statements about tf.keras.Model and tf.keras.Sequential is incorrect when the tf.keras interface is used to build a network model? ( )
  - A. tf.keras.Model supports network models with multiple inputs, while tf.keras.Sequential does not.
  - B. tf.keras.Model supports network models with multiple outputs, while tf.keras.Sequential does not.
  - C. tf.keras.Model is recommended for model building when a sharing layer exists on the network.
  - D. tf.keras.Sequential is recommended for model building when a sharing layer exists on the network.

# Summary

- This chapter describes the following content by focusing on code: Features of common deep learning frameworks, including PyTorch and TensorFlow Basic syntax and common modules of TensorFlow 2.x Development procedure of TensorFlow.

# More Information

---

Official TensorFlow website: <https://tensorflow.google.cn>

Official PyTorch website: <https://PyTorch.org/>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Huawei MindSpore AI Development Framework



# Foreword

---

- This chapter introduces the structure, design concept, and features of MindSpore based on the issues and difficulties facing by the AI computing framework, and describes the development and application process in MindSpore.

# Objectives

Upon completion of this course, you will be able to:

- Learn what MindSpore is
- Understand the framework of MindSpore
- Understand the design concept of MindSpore
- Learn features of MindSpore
- Grasp the environment setup process and development cases

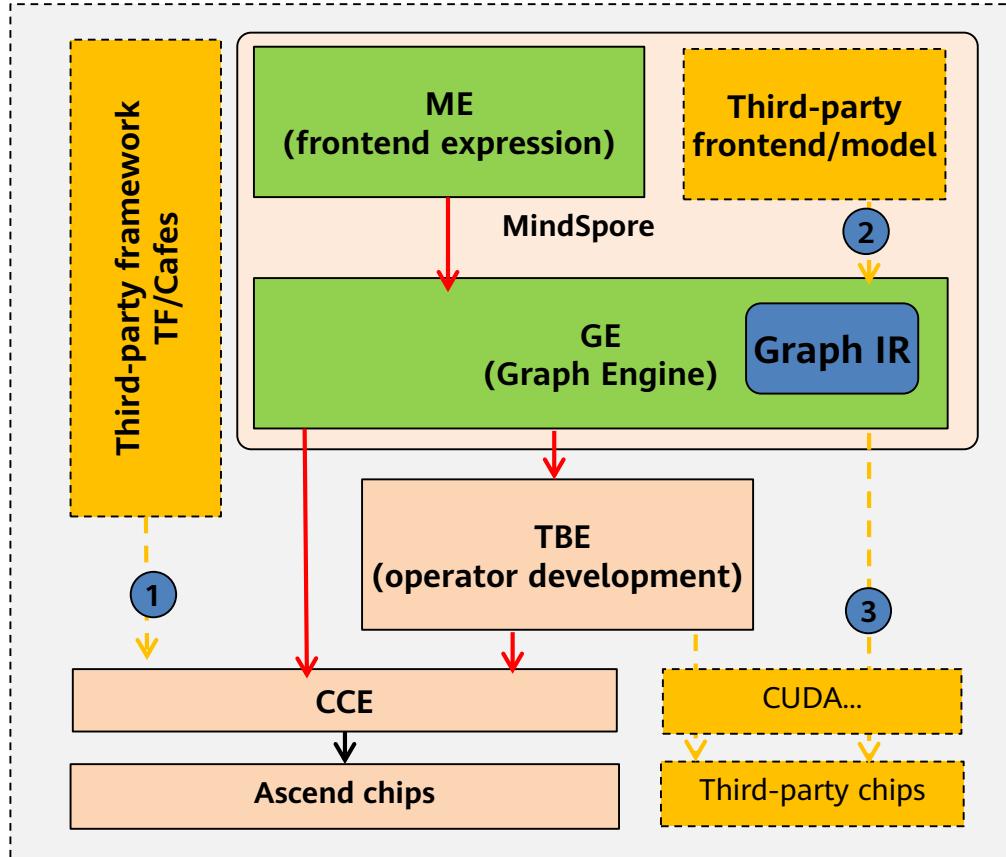
# Contents

## **1. Development Framework**

- Architecture
  - Key Features

## 2. Development and Application

# Architecture: Easy Development and Efficient Execution



## ME (Mind Expression): interface layer (Python)

**Usability:** automatic differential programming and original mathematical expression

- Auto diff: operator-level automatic differential
- Auto parallel: automatic parallelism
- Auto tensor: automatic generation of operators
- Semi-auto labeling: semi-automatic data labeling

## GE (Graph Engine): graph compilation and execution layer

**High performance:** software/hardware co-optimization, and full-scenario application

- Cross-layer memory overcommitment
- Deep graph optimization
- On-device execution
- Device-edge-cloud synergy (including online compilation)

- ① Equivalent to open-source frameworks in the industry, MindSpore preferentially serves self-developed chips and cloud services.
- ② It supports upward interconnection with third-party frameworks and can interconnect with third-party ecosystems through Graph IR, including training frontends and inference models. Developers can expand the capability of MindSpore.
- ③ It also supports interconnection with third-party chips and helps developers increase MindSpore application scenarios and expand the AI ecosystem.

# Overall Solution: Core Architecture

## MindSpore

Unified APIs for all scenarios

Auto differ

Auto parallelism

Auto tuning

MindSpore intermediate representation (IR) for computational graph

On-device execution

Pipeline parallelism

Deep graph optimization

Device-edge-cloud co-distributed architecture (deployment, scheduling, communications, etc.)

Easy development:

AI Algorithm As Code

Efficient execution:

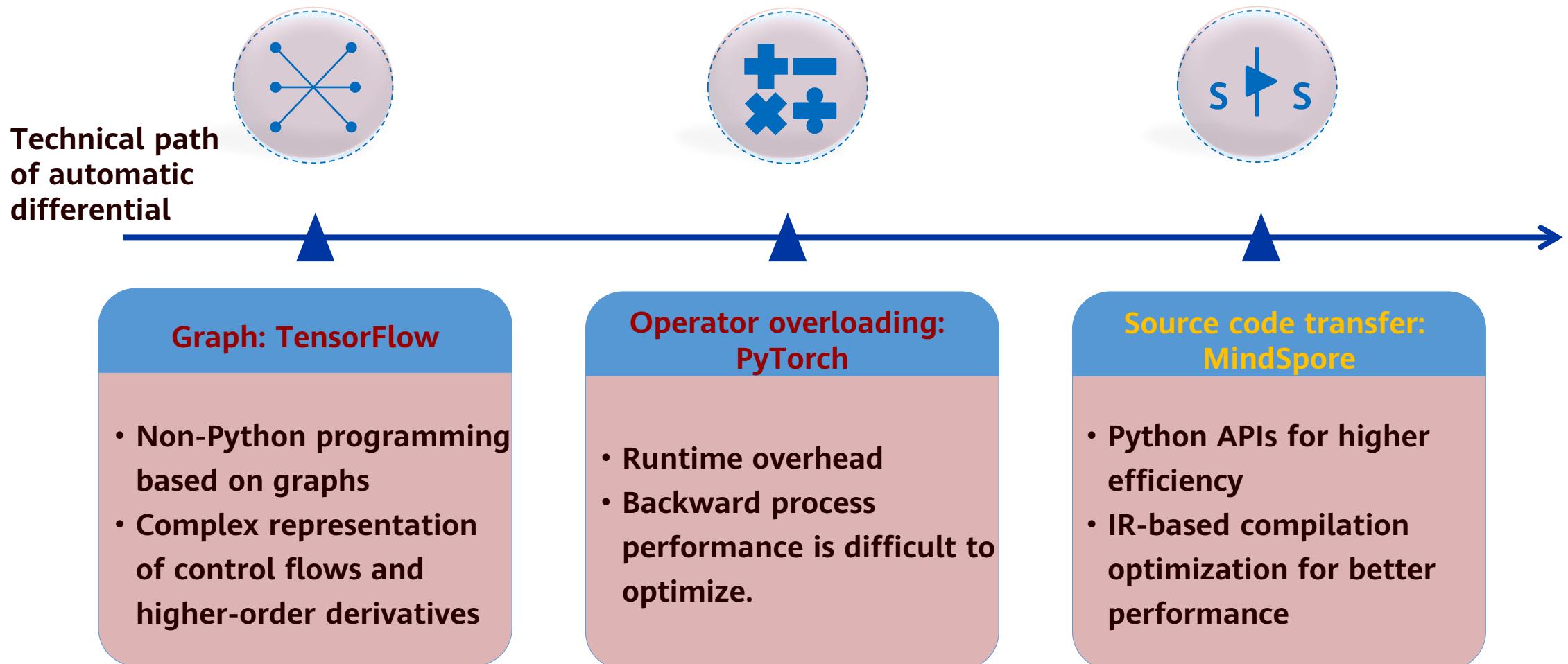
Optimized for Ascend

GPU support

Flexible deployment: on-demand cooperation across all scenarios

Processors: Ascend, GPU, and CPU

# MindSpore Design: Auto Differ



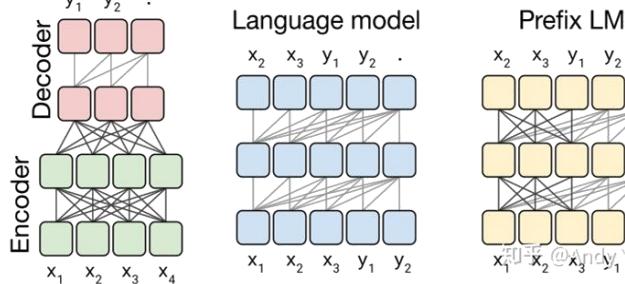
# Auto Parallelism

## Challenges

### Ultra-large models realize efficient distributed training:

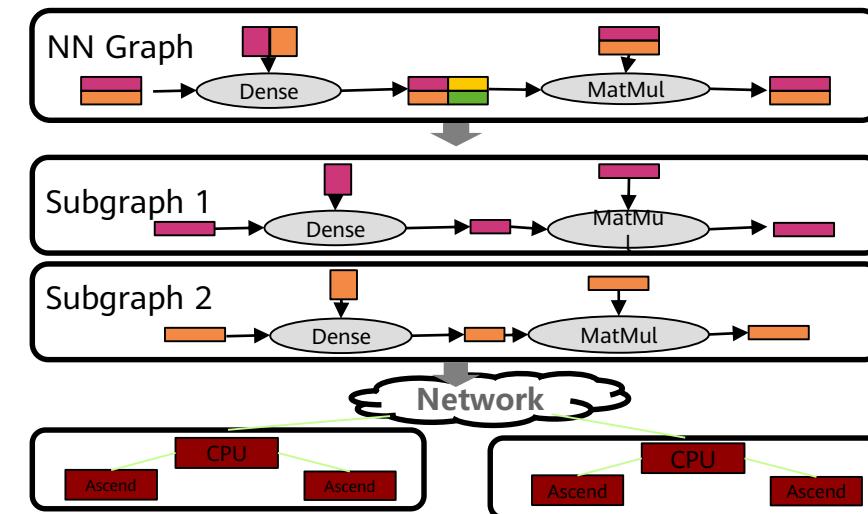
As NLP-domain models swell, the memory overhead for training ultra-large models such as Bert (340M)/GPT-2(1542M) has exceeded the capacity of a single card. Therefore, the models need to be split into multiple cards before execution.

Manual model parallelism is **used currently**. Model segmentation needs to be designed and the cluster topology needs to be understood. The development is extremely challenging. The performance is lackluster and can be hardly optimized.



## Key Technologies

Automatic graph segmentation: It can segment the entire graph based on the input and output data dimensions of the operator, and integrate the data and model parallelism. Cluster topology awareness scheduling: It can perceive the cluster topology, schedule subgraphs automatically, and minimize the communication overhead.



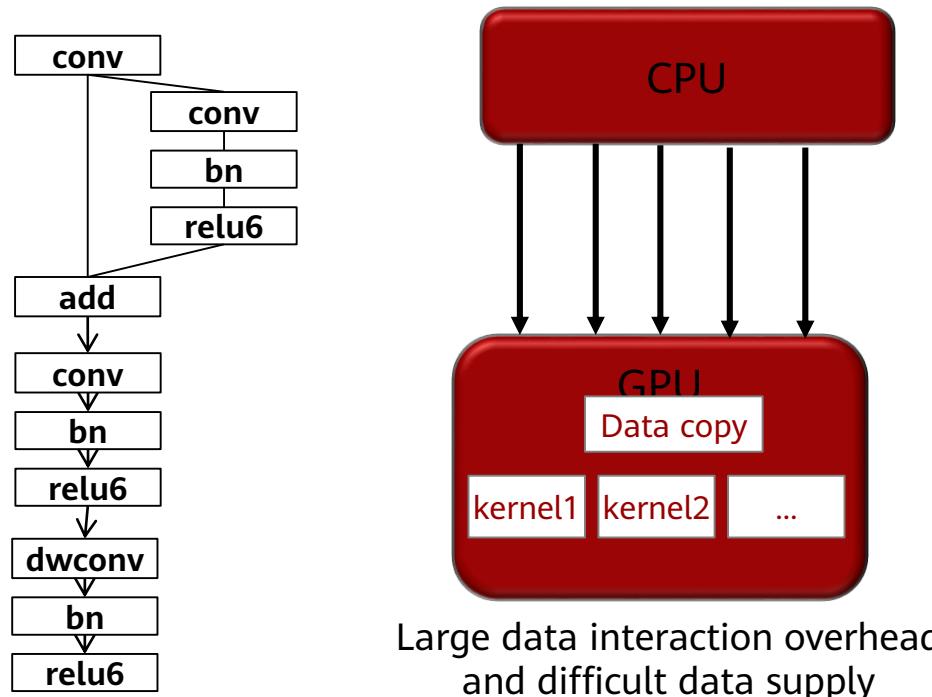
**Effect:** Realize model parallelism based on the existing single-node code logic, improving the development efficiency tenfold compared with manual parallelism.

## On-Device Execution (1)

## Challenges

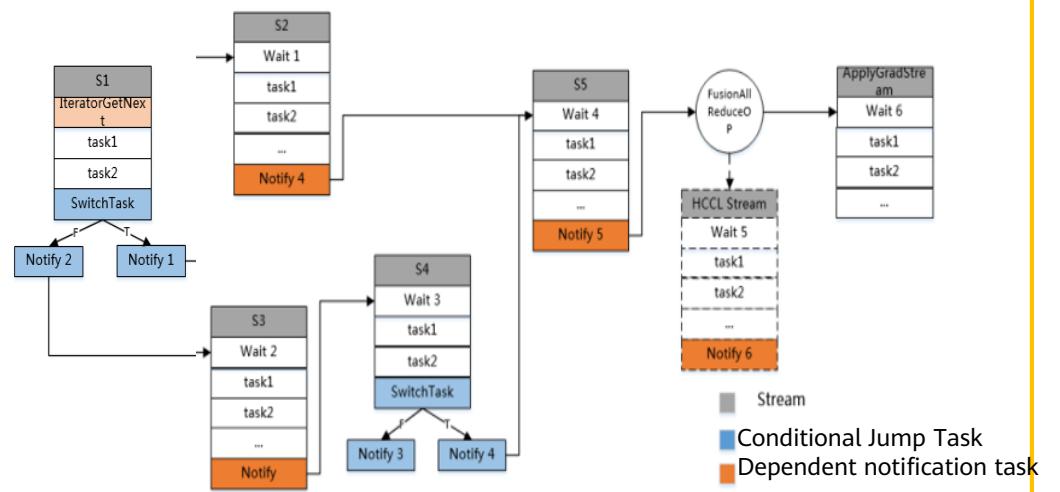
Challenges for model execution with supreme chip computing power:

Memory wall, high interaction overhead, and data supply difficulty. Partial operations are performed on the host, while the others are performed on the device. The interaction overhead is much greater than the execution overhead, resulting in the low accelerator usage.



## Key Technologies

**Chip-oriented deep graph optimization** reduces the synchronization waiting time and maximizes the parallelism of data, computing, and communication. Data pre-processing and computation are integrated into the Ascend chip:

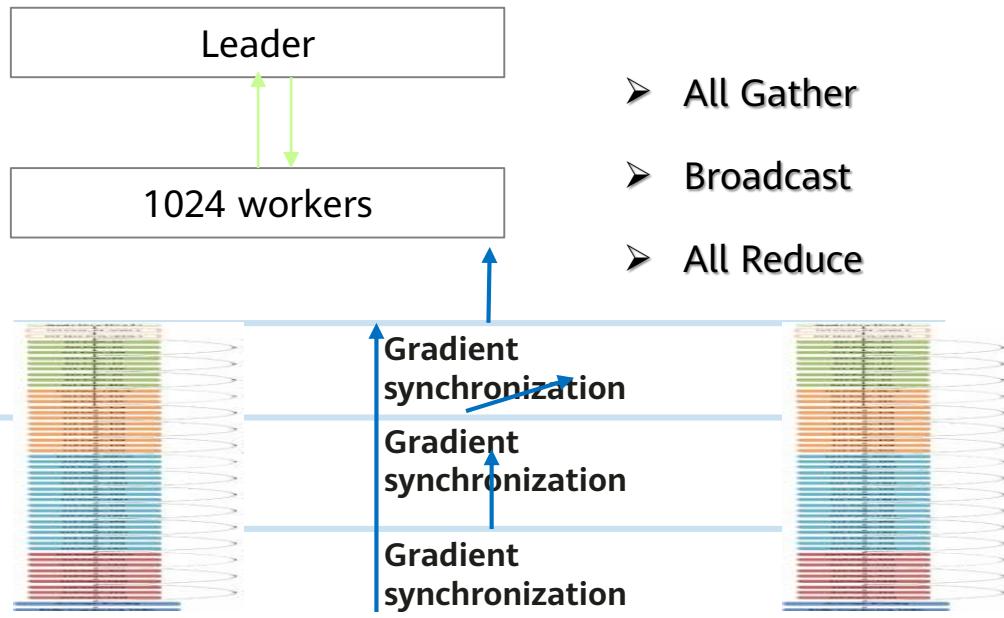


**Effect:** Elevate the training performance tenfold compared with the on-host graph scheduling.

# On-Device Execution (2)

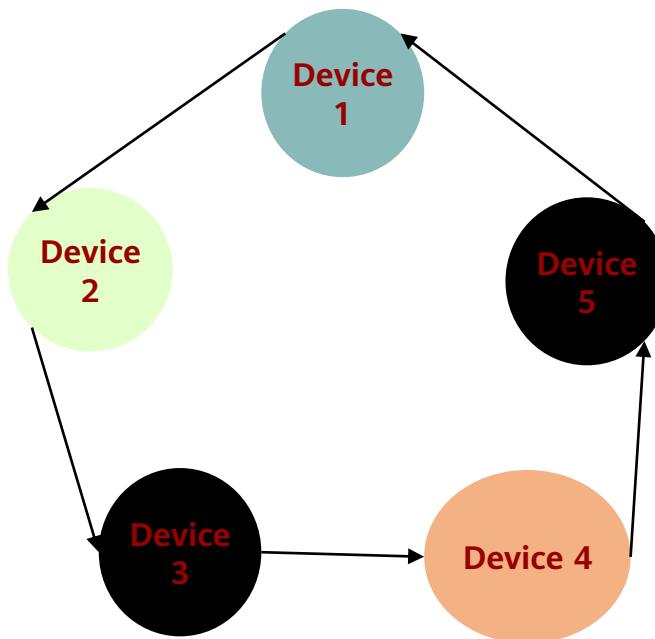
## Challenges

Challenges for distributed gradient aggregation with supreme chip computing power:  
the synchronization overhead of central control and the communication overhead of frequent synchronization of ResNet50 under the single iteration of 20 ms; the traditional method can only complete All Reduce after three times of synchronization, while the data-driven method can autonomously perform All Reduce without causing control overhead.



## Key Technologies

The optimization of the **adaptive graph segmentation driven by gradient data** can realize decentralized All Reduce and synchronize gradient aggregation, boosting computing and communication efficiency.



**Effect: a smearing overhead of less than 2 ms**

# Distributed Device-Edge-Cloud Synergy Architecture

## Challenges

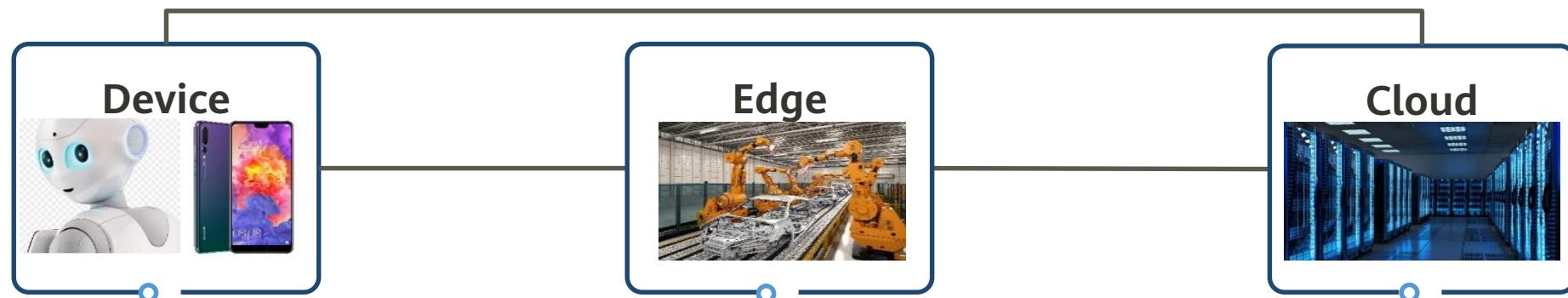
The diversity of hardware architectures leads to full-scenario deployment differences and performance uncertainties. The separation of training and inference leads to isolation of models.

## Key Technologies

- **Unified model IR** delivers a consistent deployment experience.
- **The graph optimization technology featuring software and hardware collaboration** bridges different scenarios.
- Device-cloud Synergy Federal Meta Learning breaks the device-cloud boundary and updates the multi-device collaboration model in real time.

**Effect: consistent model deployment performance across all scenarios thanks to the unified architecture, and improved precision of personalized models**

**On-demand collaboration in all scenarios and consistent development experience**



# Contents

## **1. Development Framework**

- Architecture
- Features

## **2. Development and Application**

# AI Computing Framework: Challenges

## Industry Challenges

A huge gap between industry research and all-scenario AI application

- High entry barriers
- High execution cost
- Long deployment duration

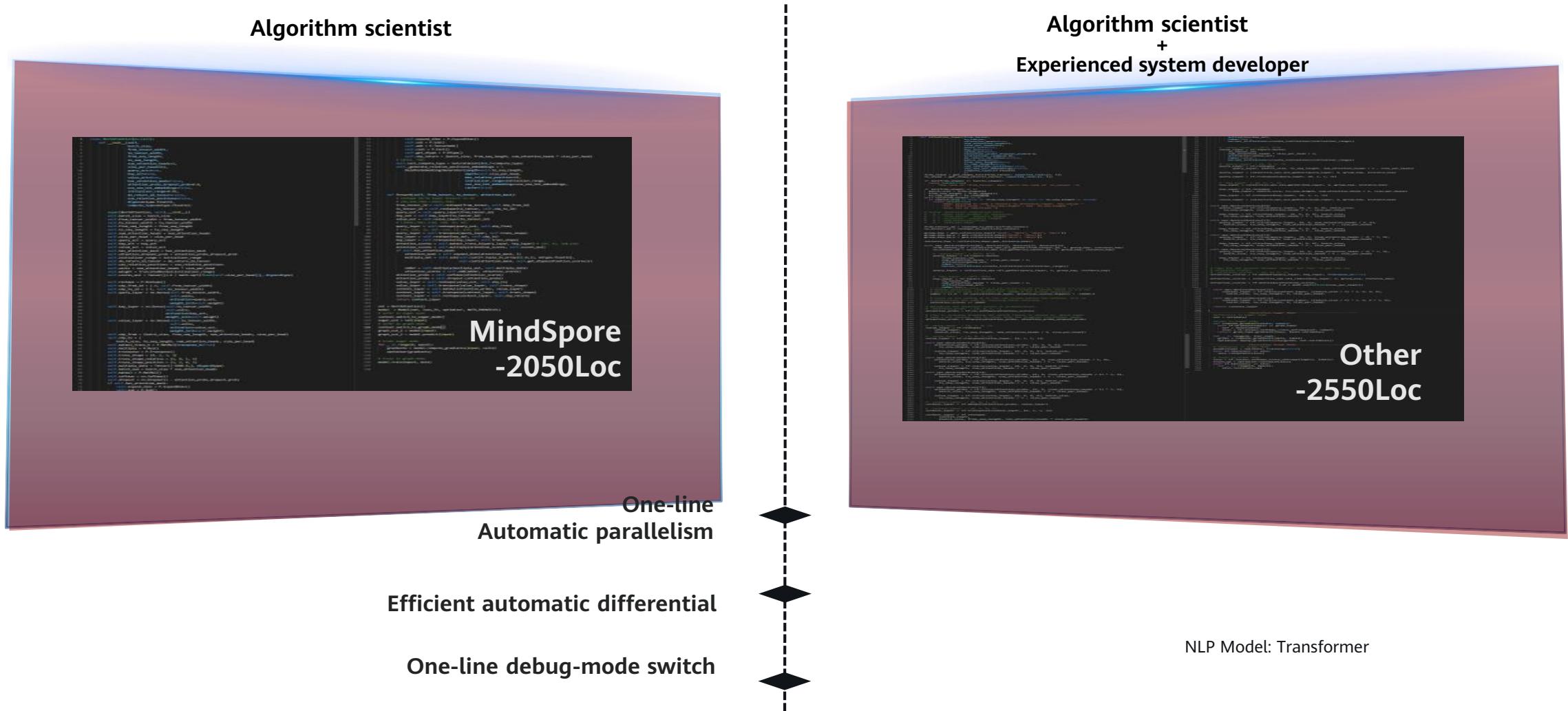


## Technological Innovation

MindSpore facilitates inclusive AI across applications

- New programming mode
- New execution mode
- New collaboration mode

# New Programming Paradigm



# Code Example

## TensorFlow code snippet: XX lines, manual parallelism

```
1 import tensorflow as tf
2
3 model() {
4     with tf.device("/device:0")
5         token_type_table = tf.get_variable(
6             name=token_type_embedding_name,
7             shape=[token_type_vocab_size, width],
8             initializer=create_initializer(initializer_range))
9         flat_token_type_ids = tf.reshape(token_type_ids, [-1])
10        one_hot_ids = tf.one_hot(flat_token_type_ids, depth=token_type_vocab_size)
11        token_type_embeddings = tf.matmul(one_hot_ids, token_type_table)
12
13    with tf.device("/device:1")
14        query_layer = tf.layers.dense(
15            from_tensor_2d,
16            num_attention_heads * size_per_head,
17            activation=query_act,
18            name="query",
19            kernel_initializer=create_initializer(initializer_range))
20
21    with tf.device("/device:2")
22        key_layer = tf.layers.dense(
23            to_tensor_2d,
24            num_attention_heads * size_per_head,
25            activation=key_act,
26            name="key",
27            kernel_initializer=create_initializer(initializer_range))
```

## MindSpore code snippet: two lines, automatic parallelism

```
class DenseMatMulNet(nn.Cell):
    def __init__(self):
        super(DenseMatMulNet, self).__init__()
        self.matmul1 = ops.MatMul.set_strategy({[4, 1], [1, 1]})
        self.matmul2 = ops.MatMul.set_strategy({[1, 1], [1, 4]})

    def construct(self, x, w, v):
        y = self.matmul1(x, w)
        z = self.matmul2(y, v)
        return z
```

## Typical scenarios: ReID

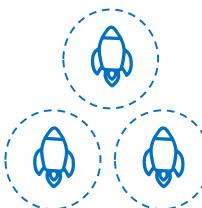
# New Execution Mode (1)

## Execution Challenges



### Complex AI computing and diverse computing units

1. CPU cores, cubes, and vectors
2. Scalar, vector, and tensor computing
3. Mixed precision computing
4. Dense matrix and sparse matrix computing



### Multi-device execution: High cost of parallel control

Performance cannot linearly increase as the node quantity increases.

## On-device execution

Offloads graphs to devices, maximizing the computing power of Ascend

### Framework optimization

Pipeline parallelism  
Cross-layer memory overcommitment

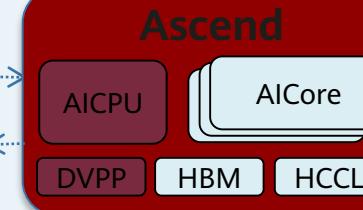
### Soft/hardware co-optimization

On-device execution  
Deep graph optimization

### MindSpore computing framework

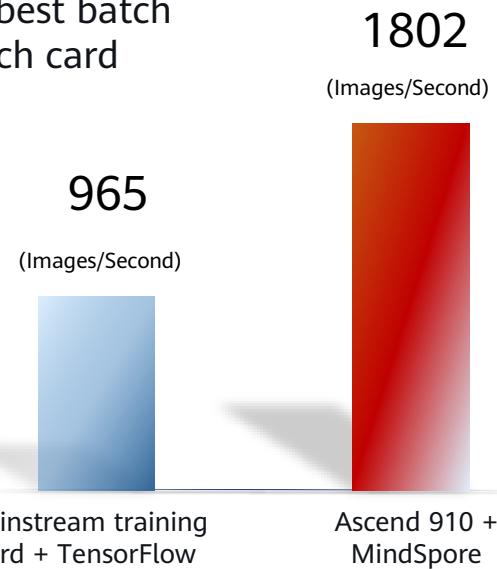
Graph + Operator

Host CPU



# New Execution Mode (2)

- ResNet 50 V1.5
- ImageNet 2012
- With the best batch size of each card



**Performance of ResNet-50 is doubled.**

Single iteration:

**58 ms** (other frameworks+V100) v.s. about **22 ms** (MindSpore) (ResNet50+ImageNet, single-server, eight-device, batch size=32)



**Detecting objects in 60 ms**

**Tracking objects in 5 ms**

**Multi-object real-time recognition**

MindSpore-based mobile deployment, a smooth experience of multi-object detection

# New Collaboration Mode

## Deployment Challenge



V.S.



- Varied requirements, objectives, and constraints for device, edge, and cloud application scenarios

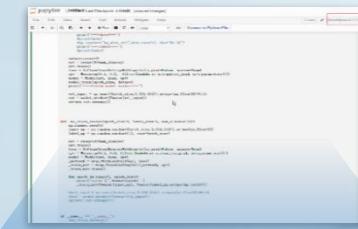


V.S.



- Different hardware precision and speed

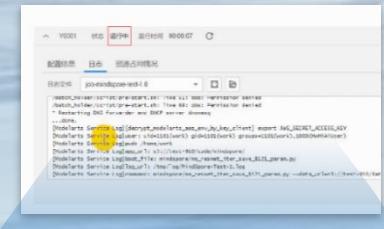
Unified development; flexible deployment; on-demand collaboration, and high security and reliability



Development



Deployment



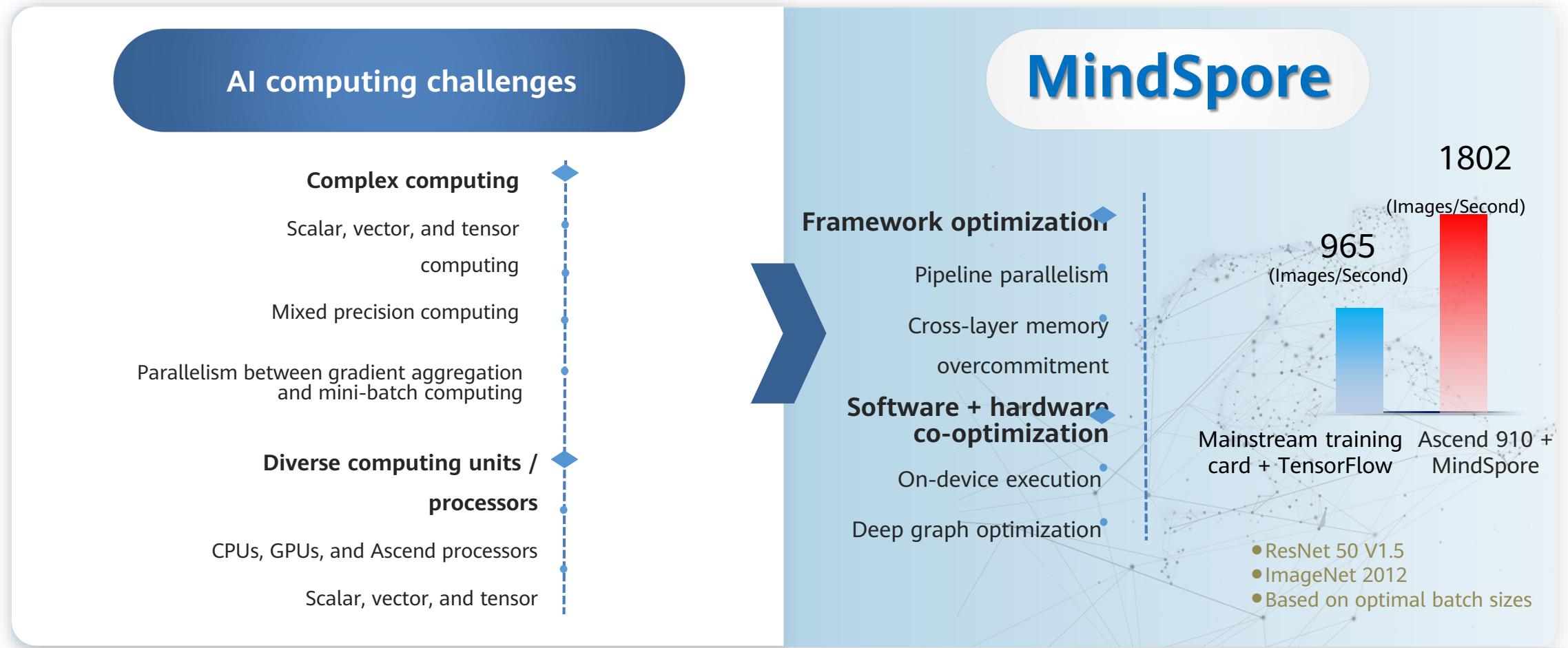
Execution



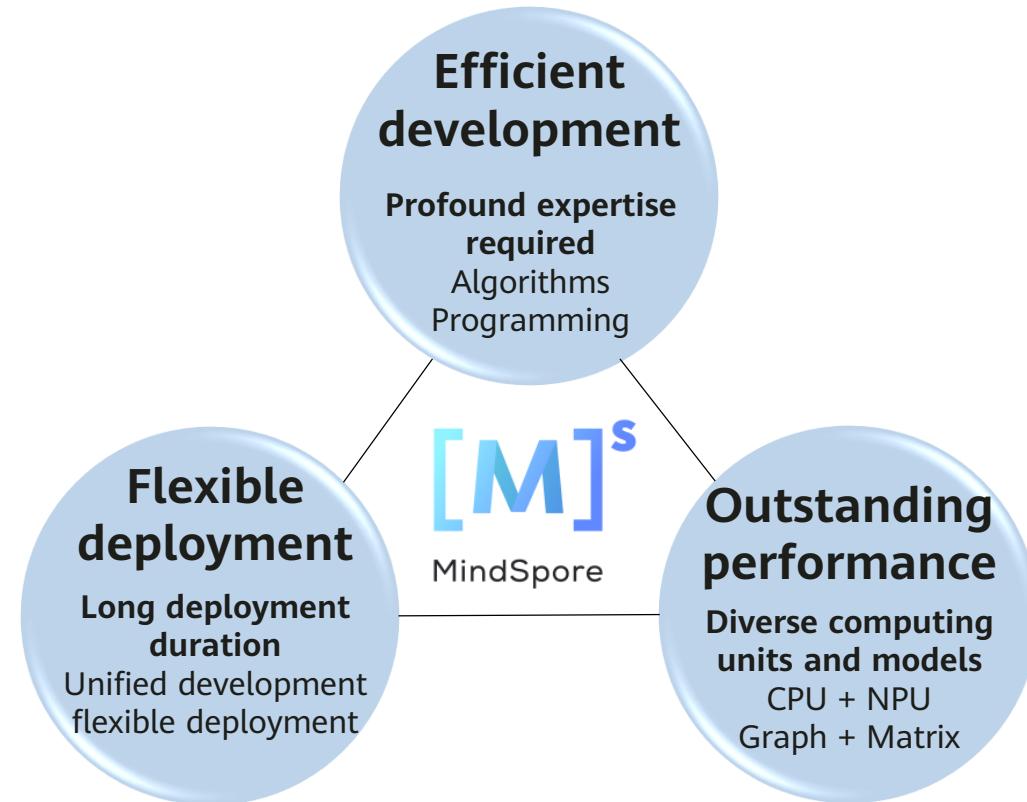
Saving model

Unified development and flexible deployment

# High Performance



# Vision and Value



# Contents

1. Development Framework
2. **Development and Application**
  - Environment Setup
  - Application Development Cases

# Installing MindSpore

## Environment Requirements

### System Requirements and Software Dependencies

Version	Operating System	Executable File Installation Dependencies	Source Code Compilation and Installation Dependencies
MindInsight 0.2.0-alpha	- Ubuntu 16.04 or later x86_64 - EulerOS 2.8 arrch64 - EulerOS 2.5 x86_64	- Python 3.7.5 - MindSpore 0.2.0-alpha - For details about other dependency items, see <a href="#">requirements.txt</a> .	<b>Compilation dependencies:</b> - Python 3.7.5 - CMake >= 3.14.1 - GCC 7.3.0 - node.js >= 10.19.0 - wheel >= 0.32.0 - pybind11 >= 2.4.3 <b>Installation dependencies:</b> same as the executable file installation dependencies.

- When the network is connected, dependency items in the requirements.txt file are automatically downloaded during .whl package installation. In other cases, you need to manually install dependency items.

## Installation Guide

### Installing Using Executable Files

- Download the .whl package from the [MindSpore website](#). It is recommended to perform SHA-256 integrity verification first and run the following command to install MindInsight:

```
pip install mindinsight-{version}-cp37-cp37m-linux_{arch}.whl
```

- Run the following command. If web address: <http://127.0.0.1:8080> is displayed, the installation is successful.

```
mindinsight start
```

### Method 1: source code compilation and installation

Two installation environments: Ascend and CPU

```
adding 'mindspore/transforms/validators.py'  
adding 'mindspore-0.1.0.dist-info/METADATA'  
adding 'mindspore-0.1.0.dist-info/WHEEL'  
adding 'mindspore-0.1.0.dist-info/top_level.txt'  
adding 'mindspore-0.1.0.dist-info/RECORD'  
removing build/bdist.linux-x86_64/wheel  
----- Successfully created mindspore package -----  
----- mindspore: build test end -----
```

### Method 2: direct installation using the installation package

Two installation environments: Ascend and CPU

Installation commands:

- `pip install -y mindspore-cpu`
- `pip install -y mindspore-d`

# Getting Started

- In MindSpore, data is stored in tensors. Common tensor operations:
  - asnumpy()
  - size()
  - dim()
  - dtype()
  - set\_dtype()
  - tensor\_add(other: Tensor)
  - tensor\_mul(other: Tensor)
  - shape()
  - \_\_Str\_\_# (conversion into strings)

## Components of ME

Module	Description
model_zoo	Defines common network models
communication	Data loading module, which defines the dataloader and dataset and processes data such as images and texts.
dataset	Dataset processing module, which reads and processes data.
common	Defines tensor, parameter, dtype, and initializer.
context	Defines the context class and sets model running parameters, such as graph and PyNative switching modes.
akg	Automatic differential and custom operator library.
nn	Defines MindSpore cells (neural network units), loss functions, and optimizers.
ops	Defines basic operators and registers reverse operators.
train	Training model and summary function modules.
utils	Utilities, which verify parameters. This parameter is used in the framework.

# Programming Concept: Operation

## Softmax operator

```
53 class Softmax(PrimitiveWithInfer):  
54     """  
55         Returns A tensor of the same shape of input.  
56  
57         This op will do softmax operation on the specified axis. Suppose a slice along the given  
58         axis is :math:`^x` then for each element :math:`^x_i` softmax function is as follows:  
59  
60         .. math::  
61             \text{output}(x_i) = \frac{\exp(x_i)}{\sum_{j=0}^{N-1}\exp(x_j)},  
62  
63         where :math:`^N` is the length of the Tensor.  
64  
65         Args:  
66             axis (Union[int, tuple]): The axis to do softmax operation. Default: -1.  
67             ....  
68  
69         @prim_attr_register  
70         def __init__(self, axis=-1):  
71             """init softmax Layer"""\n72             self.init_prim_io_names(inputs=['x'], outputs=['output'])\n73             validator.check_type("axis", axis, [int, tuple])\n74             if isinstance(axis, int):\n75                 self.add_prim_attr('axis', (axis,))\n76             for item in self.axis:\n77                 validator.check_type("item of axis", item, [int])\n78  
79         @tf.function  
80         def infer_shape(self, x_shape):  
81             return x_shape  
82         @tf.function  
83         def infer_dtype(self, x_dtype):  
84             return x_dtype
```

1. Name      2. Base class      3. Comment

4. Attributes of the operator are initialized here

5. The shape of the output tensor can be derived based on the input parameter shape of the operator.

6. The data type of the output tensor can be derived based on the data type of the input parameters.

## Common operations in MindSpore:

### - array: Array-related operators

- ExpandDims
- Squeeze
- Concat
- OnesLike
- Select
- StridedSlice
- ScatterNd...

### - math: Math-related operators

- AddN
- Cos
- Sub
- Sin
- Mul
- LogicalAnd
- MatMul
- LogicalNot
- RealDiv
- Less
- ReduceMean
- Greater...

### - nn: Network operators

- Conv2D
- MaxPool
- Flatten
- AvgPool
- Softmax
- TopK
- ReLU
- Sigmoid
- Pooling
- SGD
- BatchNorm
- SmoothL1Loss
- SoftmaxCrossEntropy
- SigmoidCrossEntropy...

### - control: Control operators

- ControlDepend

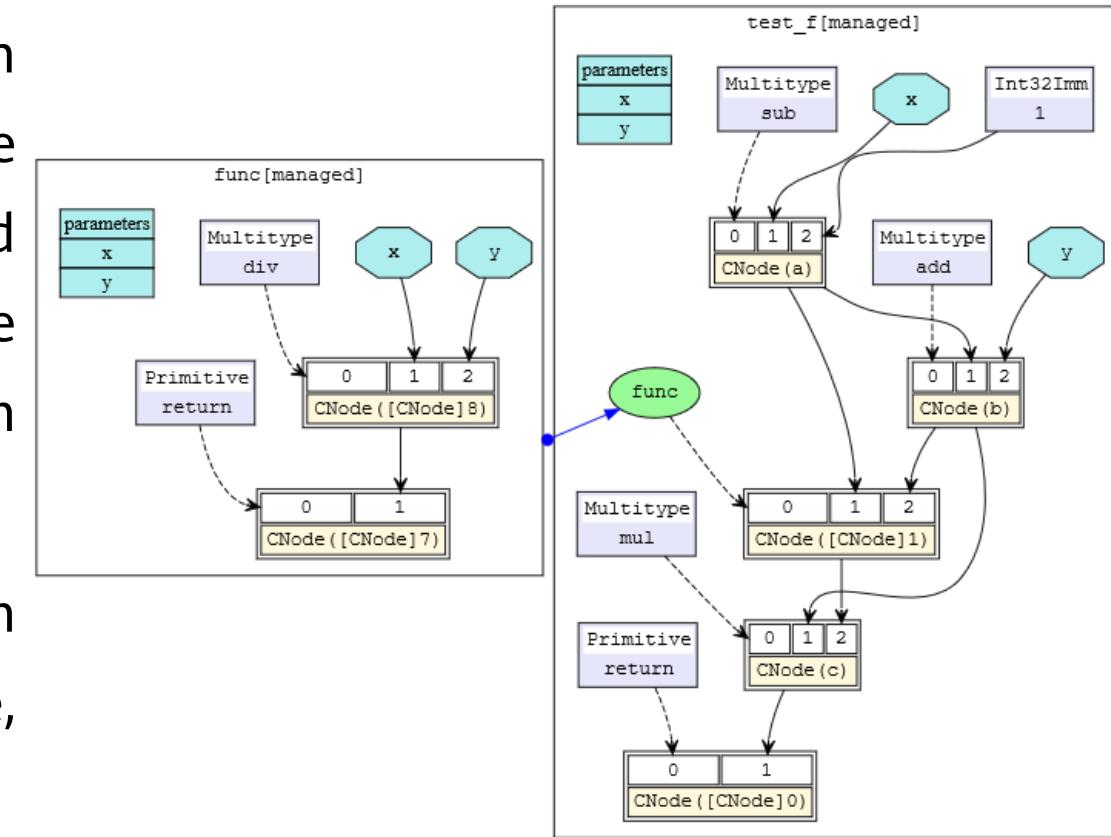
### - random: Random operators

# Programming Concept: Cell

- A cell defines the basic module for calculation. The objects of the cell can be directly executed.
  - `__init__`: It initializes and verifies modules such as parameters, cells, and primitives.
  - Construct: It defines the execution process. In graph mode, a graph is compiled for execution and is subject to specific syntax restrictions.
  - `bprop` (optional): It is the reverse direction of customized modules. If this function is undefined, automatic differential is used to calculate the reverse of the construct part.
- Cells predefined in MindSpore mainly include: common loss (Softmax Cross Entropy With Logits and MSELoss), common optimizers (Momentum, SGD, and Adam), and common network packaging functions, such as `TrainOneStepCell` network gradient calculation and update, and `WithGradCell` gradient calculation.

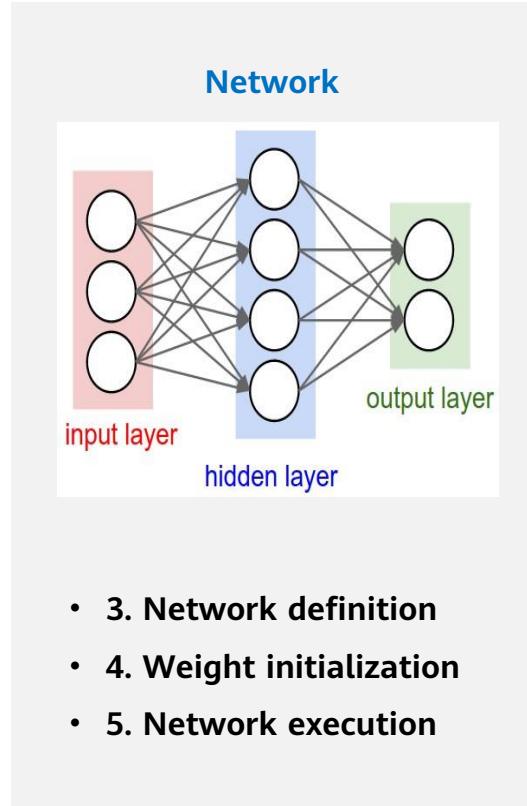
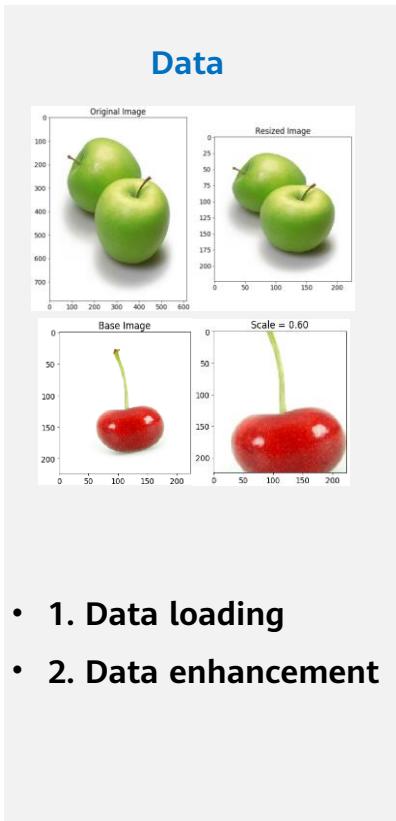
# Programming Concept: MindSporeIR

- MindSporeIR is a compact, efficient, and flexible graph-based functional IR that can represent functional semantics such as free variables, high-order functions, and recursion. It is a program carrier in the process of AD and compilation optimization.
- Each graph represents a function definition graph and consists of ParameterNode, ValueNode, and ComplexNode (CNode).
- The figure shows the def-use relationship.



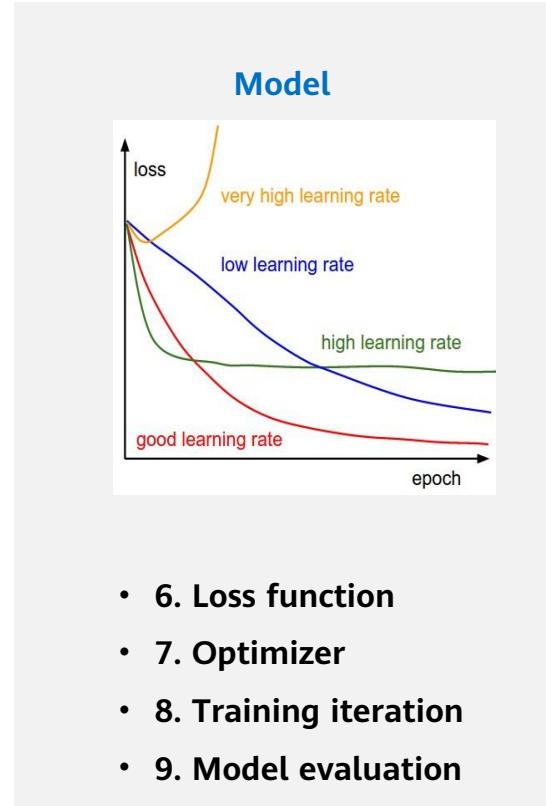
# Development Case

- Let's take the recognition of MNIST handwritten digits as an example to demonstrate the modeling process in MindSpore.

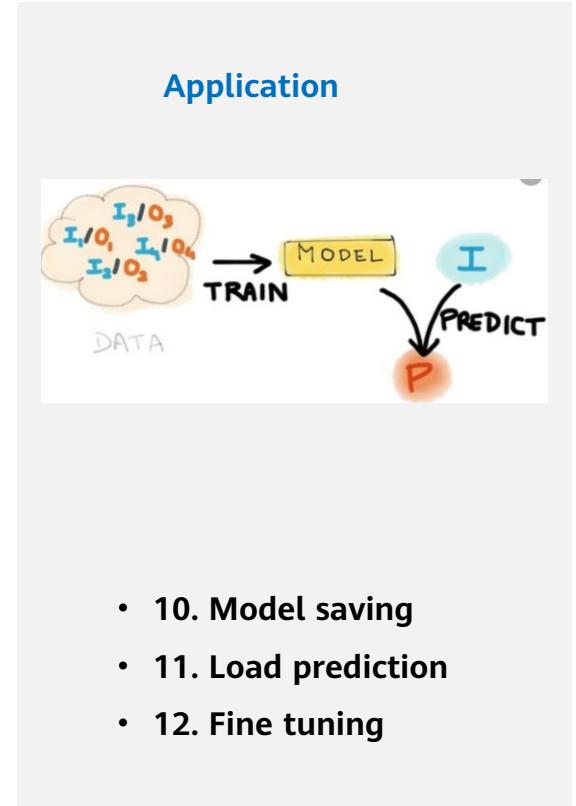


- 1. Data loading
- 2. Data enhancement

- 3. Network definition
- 4. Weight initialization
- 5. Network execution



- 6. Loss function
- 7. Optimizer
- 8. Training iteration
- 9. Model evaluation



- 10. Model saving
- 11. Load prediction
- 12. Fine tuning

# Quiz

1. In MindSpore, which of the following is the operation type of nn? ( )
  - A. Mathematical
  - B. Network
  - C. Control
  - D. Others

# Summary

- This chapter describes the framework, design, features, and the environment setup process and development procedure of MindSpore.

# More Information

---

TensoFlow: <https://www.tensorflow.org/>

PyTorch: <https://pytorch.org/>

Mindspore: <https://www.mindspore.cn/en>

Ascend developer community: <http://122.112.148.247/home>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Atlas AI Computing Platform



# Foreword

---

- This chapter describes Huawei's Ascend AI chips, and hardware and software architectures of Ascend chips, and full-stack all-scenario solutions of Ascend AI chips.

# Objectives

On completion of the course, you will be able to:

- Get an overview of AI chips.
- Understand hardware and software architectures of Huawei Ascend chips.
- Learn about Huawei Atlas AI computing platform.
- Understand industry applications of Atlas.

# Contents

## 1. Overview of AI Chips

### ■ Summary

- Classification of AI Chips
- Current Status of AI Chips
- GPU, TPU, and Ascend 310 Design Comparison
- Ascend AI Processors

## 2. Hardware Architecture of Ascend Chips

## 3. ...

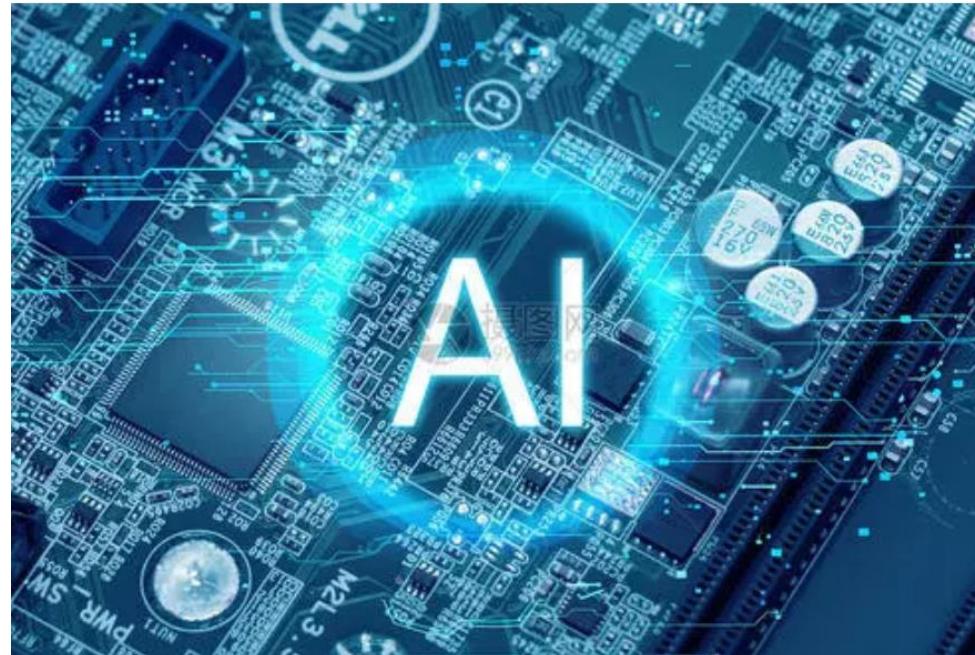
# Overview and Objectives

---

- This section is an overview of AI chips, including the introduction, classification, and status of AI chips, comparison between GPUs and CPUs, and introduction of Ascend AI processors.

# Definition

- Four elements of AI: data, algorithm, scenario, and computing power
- AI chips, also known as AI accelerators, are function modules that process massive computing tasks in AI applications.



# Contents

## **1. Overview of AI Chips**

- Summary

- Classification of AI Chips**

- Current Status of AI Chips

- Design Comparison of GPUs and CPUs

- Ascend AI Processors

## **2. Hardware Architecture of Ascend Chips**

## **3. ...**

# Classification of AI Chips (1)

- AI Chips can be divided into four types by technical architecture:
  - A central processing unit (CPU): a super-large-scale integrated circuit, which is the computing core and control unit of a computer. It can interpret computer instructions and process computer software data.
  - A graphics processing unit (GPU): a display core, visual processor, and display chip. It is a microprocessor that processes images on personal computers, workstations, game consoles, and mobile devices, such as tablet computers and smart phones.
  - An application specific integrated circuit (ASIC): an integrated circuit designed for a specific purpose.
  - A field programmable gate array (FPGA): designed to implement functions of a semi-customized chip. The hardware structure can be flexibly configured and changed in real time based on requirements.

# Classification of AI Chips (2)

- AI chips can be divided into training and inference by business application.
  - In the training phase, a complex deep neural network model needs to be trained through a large number of data inputs or an unsupervised learning method such as enhanced learning. The training process requires massive training data and a complex deep neural network structure. The huge computing amount requires ultra-high performance including computing power, precision, and scalability of processors. Nvidia GPU cluster and Google TPUs are commonly used in AI training.
  - Inferences are made using trained models and new data. For example, a video surveillance device uses the background deep neural network model to recognize a captured face. Although the calculation amount of the inference is much less than that of training, a large number of matrix operations are involved. GPU, FPGA and ASIC are also used in the inference process.

# Contents

---

## 1. Overview of AI Chips

- Summary
- Classification of AI Chips

### ■ Current Status of AI Chips

- Design Comparison of GPUs and CPUs
- Ascend AI Processors

## 2. Hardware Architecture of Ascend Chips

## 3. ...

# Current Status of AI Chips - CPU

- Central processing unit (CPU)
  - The computer performance has been steadily improved based on the Moore's Law.
  - The CPU cores added for performance enhancement also increase power consumption and cost.
  - Extra instructions have been introduced and the architecture has been modified to improve AI performance.
    - Instructions, such as AVX512, have been introduced into Intel processors (CISC architecture) and vector computing modules, such as FMA, into the ALU computing module.
    - Instruction sets including Cortex A have been introduced into ARM (RISC architecture), which will be upgraded continuously.
  - Despite that boosting the processor frequency can elevate the performance, the high frequency will cause huge power consumption and overheating of the chip as the frequency reaches the ceiling.

# Current Status of AI Chips - GPU

- Graph processing unit (GPU)
  - GPU performs remarkably in matrix computing and parallel computing and plays a key role in heterogeneous computing. It was first introduced to the AI field as an acceleration chip for deep learning. Currently, the GPU ecosystem has matured.
  - Using the GPU architecture, NVIDIA focuses on the following two aspects of deep learning:
    - Diversifying the ecosystem: It has launched the cuDNN optimization library for neural networks to improve usability and optimize the GPU underlying architecture.
    - Improving customization: It supports various data types, including int8 in addition to float32; introduces modules dedicated for deep learning. For example, the optimized architecture of Tensor cores has been introduced, such as the TensorCore of V100.
  - The existing problems include high costs and latency and low energy efficiency.

# Current Status of AI Chips - TPU

- Tensor processing unit (TPU)
  - Since 2006, Google has sought to apply the design concept of ASICs to the neural network field and released TPU, a customized AI chip that supports TensorFlow, which is an open-source deep learning framework.
  - Massive systolic arrays and large-capacity on-chip storage are adopted to accelerate the most common convolution operations in deep neural networks.
    - Systolic arrays optimize matrix multiplication and convolution operations to elevate computing power and lower energy consumption.



# Current Status of AI Chips - FPGA

- Field programmable gate array (FPGA)
  - Using the HDL programmable mode, FPGAs are highly flexible, reconfigurable and re-programmable, and customizable.
  - Multiple FPGAs can be used to load the DNN model on the chips to lower computing latency. FPGAs outperform GPUs in terms of computing performance. However, the optimal performance cannot be achieved due to continuous erasing and programming. Besides, redundant transistors and cables, logic circuits with the same functions occupy a larger chip area.
  - The reconfigurable structure lowers supply and R&D risks. The cost is relatively flexible depending on the purchase quantity.
  - The design and tapeout processes are decoupled. The development period is long, generally half a year. The entry barrier is high.

# Contents

## **1. Overview of AI Chips**

- Summary
- Classification of AI Chips
- Current Status of AI Chips
- Design Comparison of GPUs and CPUs
  - Ascend AI Processors

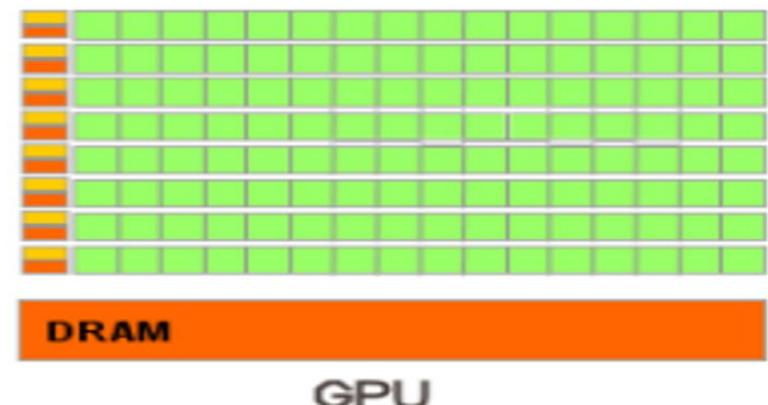
## **2. Hardware Architecture of Ascend Chips**

## **3. Software Architecture of Ascend Chips**

## **4. ...**

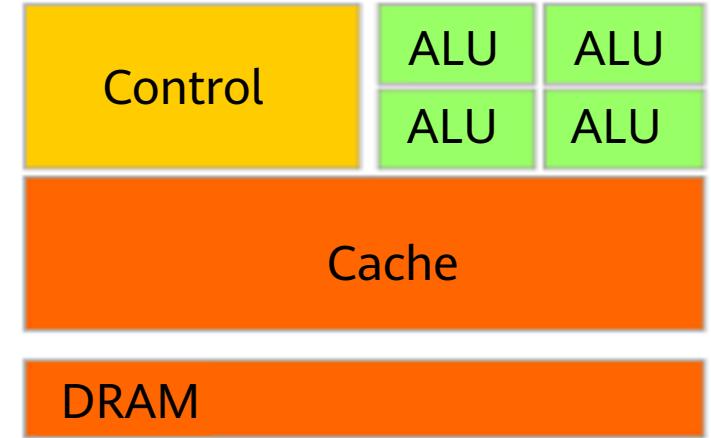
# Design Comparison of GPUs and CPUs

- GPUs are designed for massive data of the same type independent from each other and pure computing environments that do not need to be interrupted.
  - Each GPU comprises several large-sized parallel computing architectures with thousands of smaller cores designed to handle multiple tasks simultaneously.
  - Throughput-oriented design
    - With many ALUs and few caches, which improve services for threads, unlike those in CPU. The cache merges access to DRAM, causing latency.
    - The control unit performs combined access.
    - A large number of ALUs process numerous threads concurrently to cover up the latency.
  - Specialized in computing-intensive and easy-to-parallel programs



# Design Comparison of GPUs and CPUs

- CPUs need to process different data types in a universal manner, perform logic judgment, and introduce massive branch jumps and interrupted processing.
  - Composed of several cores optimized for sequential serial processing
  - Low-latency design
    - The powerful ALU unit can complete the calculation in a short clock cycle.
    - The large cache lowers latency.
    - High clock frequency
    - Complex logic control unit, multi-branch programs can reduce latency through branch prediction.
    - For instructions that depend on the previous instruction result, the logic unit determines the location of the instructions in the pipeline to speed up data forwarding.
  - Specialized in logic control and serial operation



# Contents

---

## 1. Overview of AI Chips

- Summary
- Classification of AI Chips
- Current Status of AI Chips
- Design Comparison of GPUs and CPUs
- Ascend AI Processors

## 2. Hardware Architecture of Ascend Chips

## 3. Software Architecture of Ascend Chips

## 4. ...

# Ascend AI Processors

- Neural-network processing unit (NPU): uses a deep learning instruction set to process a large number of human neurons and synapses simulated at the circuit layer. One instruction is used to process a group of neurons.
- Typical NPUs: Huawei Ascend AI chips, Cambricon chips, and IBM TrueNorth



- Ascend-Mini
- Architecture: Da Vinci
- Half precision (FP16): 8 Tera-FLOPS
- Integer precision (INT8): 16 Tera-OPS
- 16-channel full-HD video decoder: H.264/H.265
- 1-channel full-HD video decoder: H.264/H.265
- **Max. power: 8W**
- 12nm FFC



- Ascend-Max
- Architecture: Da Vinci
- Half precision (FP16): 256 Tera-FLOPS
- Integer precision (INT8): 512 Tera-OPS
- 128-channel full-HD video decoder: H.264/H.265
- Max. power: 350W
- 7nm

# Section Summary

---

- This section describes AI chips, including classification of AI chips by technologies and functions, AI chip ecosystem, and comparison between GPUs and CPUs.

# Contents

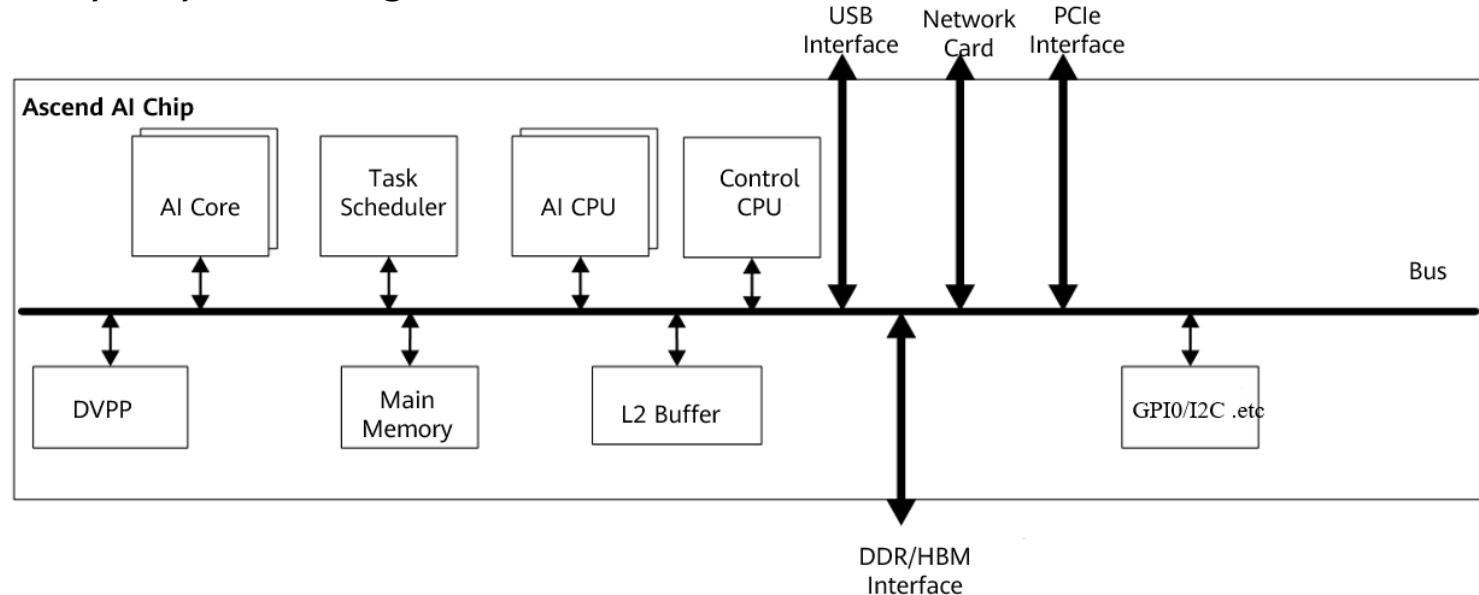
1. Overview of AI Chips
2. **Hardware Architecture of Ascend Chips**
  - Logic Architecture of Ascend AI Processors
    - Da Vinci Architecture
3. Software Architecture of Ascend Chips
4. Huawei Atlas AI Computing Platform
5. Industry Applications of Atlas

# Overview and Objectives

- This section describes the hardware architecture of Ascend chips, including the logic architecture of the Ascend AI processors and Da Vinci architecture.

# Logic Architecture of Ascend AI Processors

- Ascend AI processor consist of:
  - Control CPU
  - AI computing engine, including AI core and AI CPU
  - Multi-layer system-on-chip (SoC) caches or buffers
  - Digital vision pre-processing (DVPP) module



# Contents

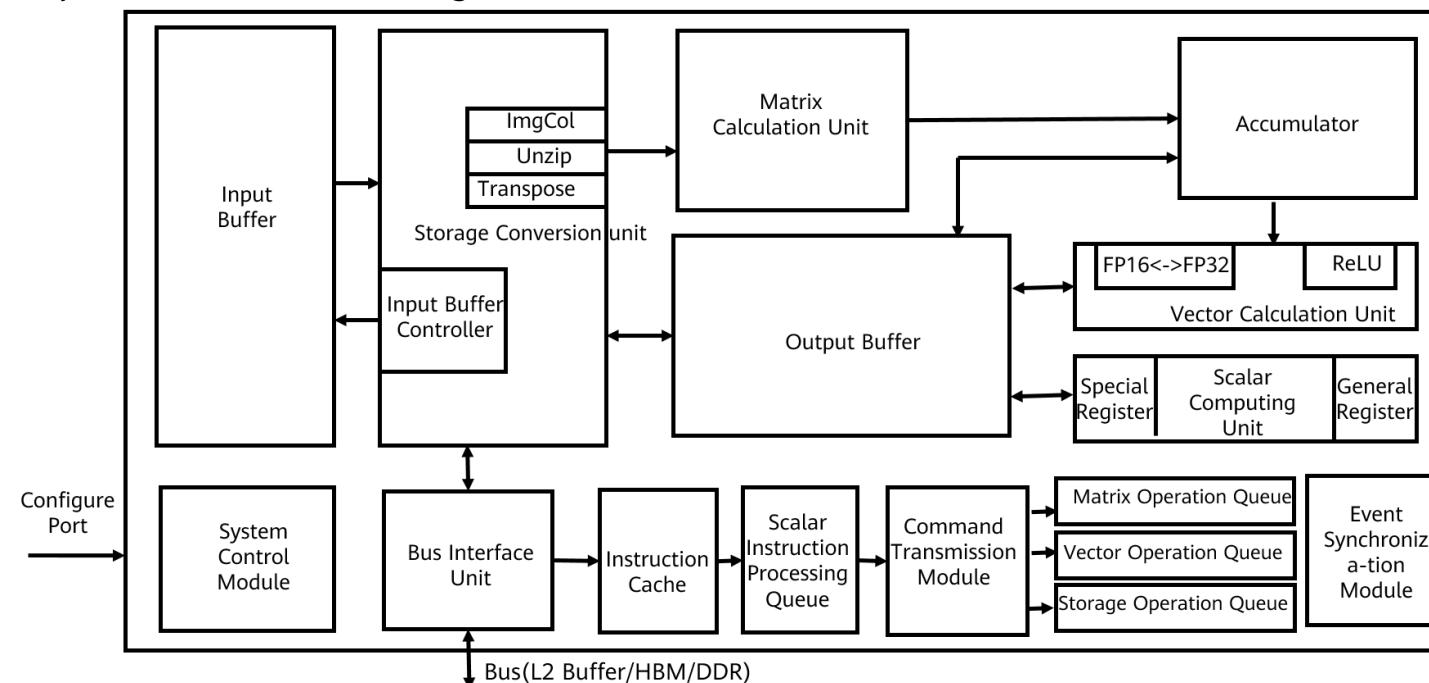
1. Overview of AI Chips
2. **Hardware Architecture of Ascend Chips**
  - Logic Architecture of Ascend AI Processors
  - Da Vinci Architecture
3. Software Architecture of Ascend Chips
4. Huawei Atlas AI Computing Platform
5. Industry Applications of Atlas

# Ascend AI Computing Engine - Da Vinci Architecture

- One of the four major architectures of Ascend AI processors is the AI computing engine, which consists of the AI core (Da Vinci architecture) and AI CPU. The Da Vinci architecture developed to improve the AI computing power serves as the core of the Ascend AI computing engine and AI processor.

# Da Vinci Architecture (AI Core)

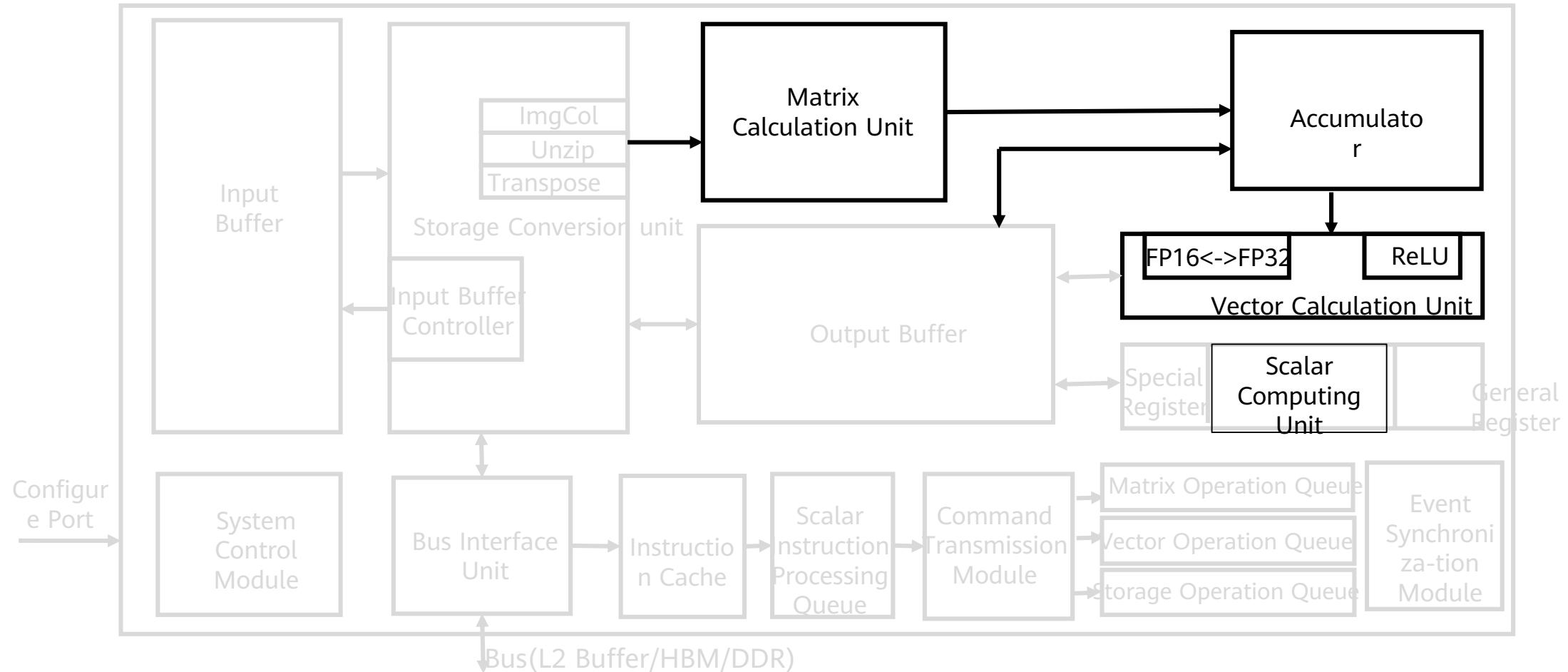
- Main components of the Da Vinci architecture:
  - Computing unit: It consists of the cube unit, vector unit, and scalar unit.
  - Storage system: It consists of the on-chip storage unit of the AI core and data channels.
  - Control unit provides instruction control for the entire computing process. It is equivalent to the command center of the AI core and is responsible for the running of the entire AI core.



# Da Vinci Architecture (AI Core) - Computing Unit

- Three types of basic computing units: cube, vector, and scalar units, which correspond to matrix, vector and scalar computing modes respectively.
- Cube computing unit: The matrix computing unit and accumulator are used to perform matrix-related operations. Completes a matrix (4096) of 16x16 multiplied by 16x16 for FP16, or a matrix (8192) of 16x32 multiplied by 32x16 for the INT8 input in a shot.
- Vector computing unit: Implements computing between vectors and scalars or between vectors. This function covers various basic computing types and many customized computing types, including computing of data types such as FP16, FP32, INT32, and INT8.
- Scalar computing unit: Equivalent to a micro CPU, the scalar unit controls the running of the entire AI core. It implements loop control and branch judgment for the entire program, and provides the computing of data addresses and related parameters for cubes or vectors as well as basic arithmetic operations.

# Da Vinci Architecture (AI Core) - Computing Unit

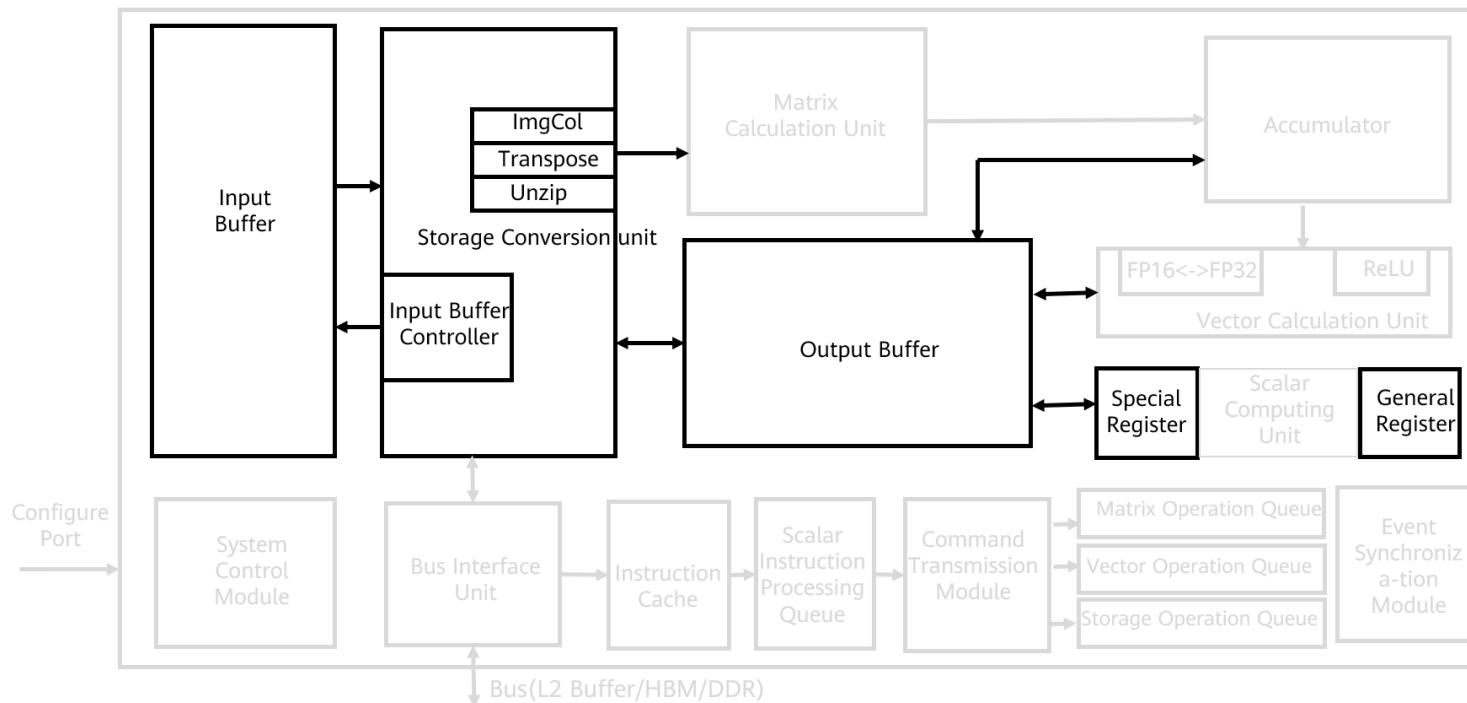


# Da Vinci Architecture (AI Core) - Storage System (1)

- The storage system of the AI core is composed of the storage unit and corresponding data channel.
- The storage unit consists of the storage control unit, buffer, and registers:
- Storage control unit: The cache at a lower level than the AI core can be directly accessed through the bus interface. The memory can also be directly accessed through the DDR or HBM. A storage conversion unit is set as a transmission controller of the internal data channel of the AI core to implement read/write management of internal data of the AI core between different buffers. It also completes a series of format conversion operations, such as zero padding, Img2Col, transposing, and decompression.
- Input buffer: The buffer temporarily stores the data that needs to be frequently used so the data does not need to be read from the AI core through the bus interface each time. This mode reduces the frequency of data access on the bus and the risk of bus congestion, thereby reducing power consumption and improving performance.
- Output buffer: The buffer stores the intermediate results of computing at each layer in the neural network, so that the data can be easily obtained for next-layer computing. Reading data through the bus involves low bandwidth and long latency, whereas using the output buffer greatly improves the computing efficiency.
- Register: Various registers in the AI core are mainly used by the scalar unit.

# Da Vinci Architecture (AI Core) - Storage System (2)

- Data channel: path for data flowing in the AI core during execution of computing tasks
  - A data channel of the Da Vinci architecture is characterized by multiple-input single-output. Considering various types and a large quantity of input data in the computing process on the neural network, parallel inputs can improve data inflow efficiency. On the contrary, only an output feature matrix is generated after multiple types of input data are processed. The data channel with a single output of data reduces the use of chip hardware resources.

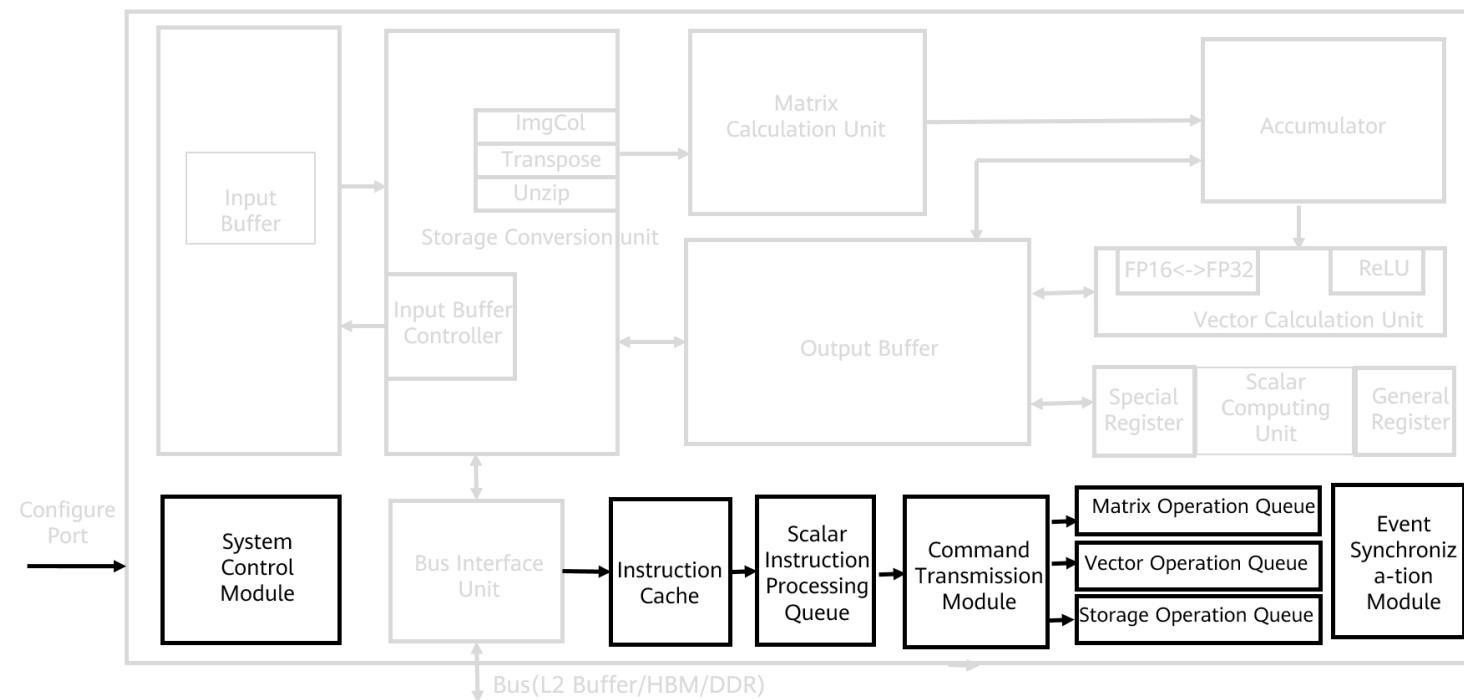


# Da Vinci Architecture (AI Core) - Control Unit (1)

- The control unit consists of the system control module, instruction cache, scalar instruction processing queue, instruction transmitting module, matrix operation queue, vector operation queue, storage conversion queue, and event synchronization module.
  - System control module: Controls the execution process of a task block (minimum task computing granularity for the AI core). After the task block is executed, the system control module processes the interruption and reports the status. If an error occurs during the execution, the error status is reported to the task scheduler.
  - Instruction cache: Prefetches subsequent instructions in advance during instruction execution and reads multiple instructions into the cache at a time, improving instruction execution efficiency.
  - Scalar instruction procession queue: After being decoded, the instructions are imported into a scalar queue to implement address decoding and operation control. The instructions include matrix computing instructions, vector calculation instructions, and storage conversion instructions.
  - Instruction transmitting module: Reads the configured instruction addresses and decoded parameters in the scalar instruction queue, and sends them to the corresponding instruction execution queue according to the instruction type. The scalar instructions reside in the scalar instruction processing queue for subsequent execution.

# Da Vinci Architecture (AI Core) - Control Unit (2)

- Instruction execution queue: Includes a matrix operation queue, vector operation queue, and storage conversion queue. Different instructions enter corresponding operation queues, and instructions in the queues are executed according to the entry sequence.
- Event synchronization module: Controls the execution status of each instruction pipeline in real time, and analyzes dependence relationships between different pipelines to resolve problems of data dependence and synchronization between instruction pipelines.



# Section Summary

- This section describes the hardware architecture of Ascend chips, including the computing unit, storage unit, and control unit of the core Da Vinci architecture.

# Contents

---

1. Overview of AI Chips
2. Hardware Architecture of Ascend Chips

## **3. Software Architecture of Ascend Chips**

### ■ Logic Architecture of Ascend 310

- Neural Network Software Flow of Ascend 310
- Data Flowchart of Ascend 310

## 4. Huawei Atlas AI Computing Platform

## 5. Industry Applications of Atlas

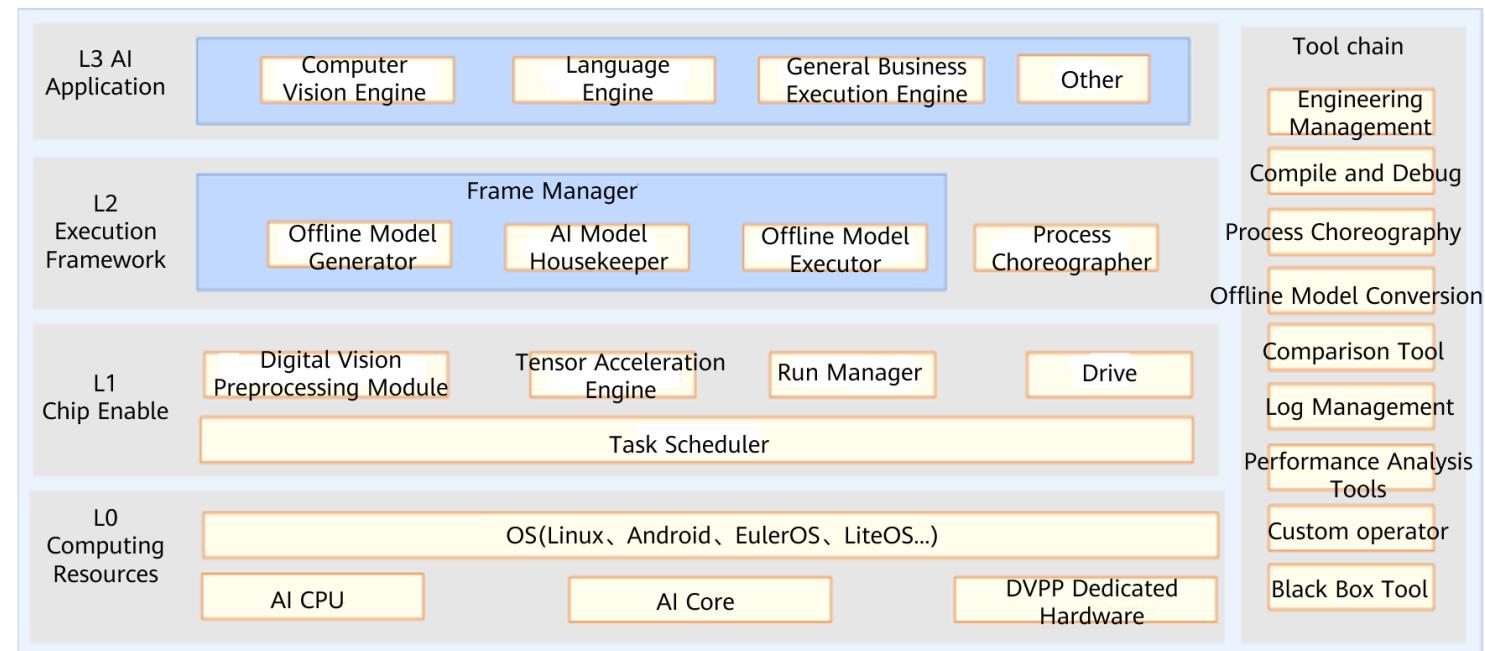
# Overview and Objectives

---

- This section describes the software architecture of Ascend chips, including the logic architecture and neural network software flow of Ascend AI processors.

# Logic Architecture of Ascend AI Processor Software Stack (1)

- L3 application enabling layer: It is an application-level encapsulation layer that provides different processing algorithms for specific application fields. L3 provides various fields with computing and processing engines. It can directly use the framework scheduling capability provided by L2 to generate corresponding NNs and implement specific engine functions.
  - Generic engine: provides the generic neural network inference capability.
  - Computer vision engine: encapsulates video or image processing algorithms.
  - Language and text engine: encapsulates basic processing algorithms for voice and text data.



# Logic Architecture of Ascend AI Processor Software Stack (2)

- L2 execution framework layer: encapsulates the framework calling capability and offline model generation capability. After the application algorithm is developed and encapsulated into an engine at L3, L2 calls the appropriate deep learning framework, such as Caffe or TensorFlow, based on the features of the algorithm to obtain the neural network of the corresponding function, and generates an offline model through the framework manager. After L2 converts the original neural network model into an offline model that can be executed on Ascend AI chips, the offline model executor (OME) transfers the offline model to Layer 1 for task allocation.
- L1 chip enabling layer: bridges the offline model to Ascend AI chips. L1 accelerates the offline model for different computing tasks via libraries. Nearest to the bottom-layer computing resources, L1 outputs operator-layer tasks to the hardware.
- L0 computing resource layer: provides computing resources and executes specific computing tasks. It is the hardware computing basis of the Ascend AI chip.

# Contents

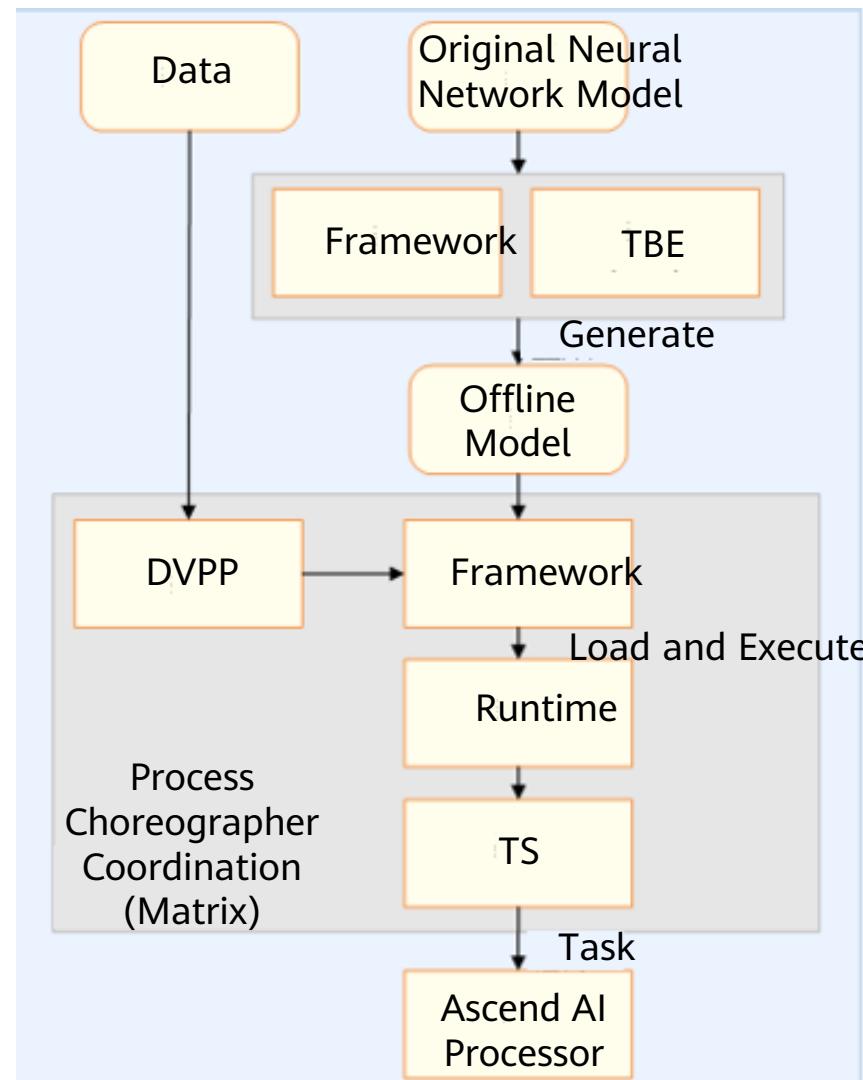
---

1. Overview of AI Chips
2. Hardware Architecture of Ascend Chips
- 3. Software Architecture of Ascend Chips**
  - Logic Architecture of Ascend 310
  - Neural Network Software Flow of Ascend 310
    - Data Flowchart of Ascend 310
4. Huawei Atlas AI Computing Platform
5. Industry Applications of Atlas

# Neural Network Software Flow of Ascend AI Processors

- The neural network software flow of Ascend AI processors is a bridge between the deep learning framework and Ascend AI chips. It realizes and executes a neural network application and integrates the following functional modules.
- Process orchestrator: implements the neural network on Ascend AI chips, coordinates the whole process of effecting the neural network, and controls the loading and execution of offline models.
- Digital vision pre-processing (DVPP) module: performs data processing and cleaning before input to meet format requirements for computing.
- Tensor boosting engine (TBE): functions as a neural network operator factory that provides powerful computing operators for neural network models.
- Framework manager: builds an original neural network model into a form supported by Ascend AI chips, and integrates the new model into Ascend AI chips to ensure efficient running of the neural network.
- Runtime manager: provides various resource management paths for task delivery and allocation of the neural network.
- Task scheduler: As a task driver for hardware execution, it provides specific target tasks for Ascend AI chips. The operation manager and task scheduler work together to form a dam system for neural network task flow to hardware resources, and monitor and distribute different types of execution tasks in real time.

# Neural Network Software Flow of Ascend AI Processors



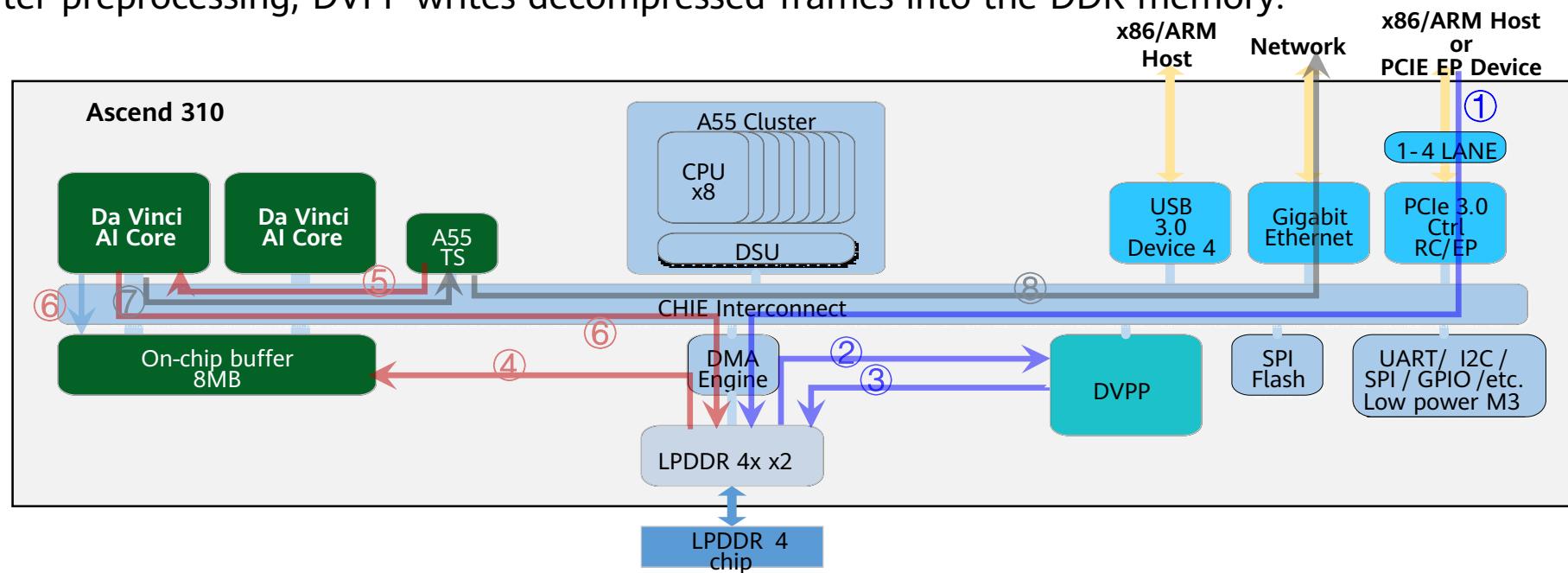
# Contents

---

1. Overview of AI Chips
2. Hardware Architecture of Ascend Chips
- 3. Software Architecture of Ascend Chips**
  - Logic Architecture of Ascend 310
  - Neural Network Software Flow of Ascend 310
  - Data Flowchart of Ascend 310
4. Huawei Atlas AI Computing Platform
5. Industry Applications of Atlas

# Data Flowchart of the Ascend AI Processor - Facial Recognition Inference Application (1)

- Camera data collection and processing
  - Compressed video streams are transmitted from the camera to the DDR memory through PCIe.
  - DVPP reads the compressed video streams into the cache.
  - After preprocessing, DVPP writes decompressed frames into the DDR memory.



# Data Flowchart of the Ascend AI Processor - Facial Recognition Inference Application (2)

- Data inference
  - The task scheduler (TS) sends an instruction to the DMA engine to pre-load the AI resources from the DDR to the on-chip buffer.
  - The TS configures the AI core to execute tasks.
  - The AI core reads the feature map and weight, and writes the result to the DDR or on-chip buffer.
- Facial recognition result output
  - After processing, the AI core sends the signals to the TS, which checks the result. If another task needs to be allocated, the operation in step ④ is performed.
  - When the last AI task is complete, the TS reports the result to the host.

# Section Summary

---

- This section describes the software architecture of Ascend chips, including the L0-L3 software architectures and the neural network software flow of the Ascend AI processor.

# Contents

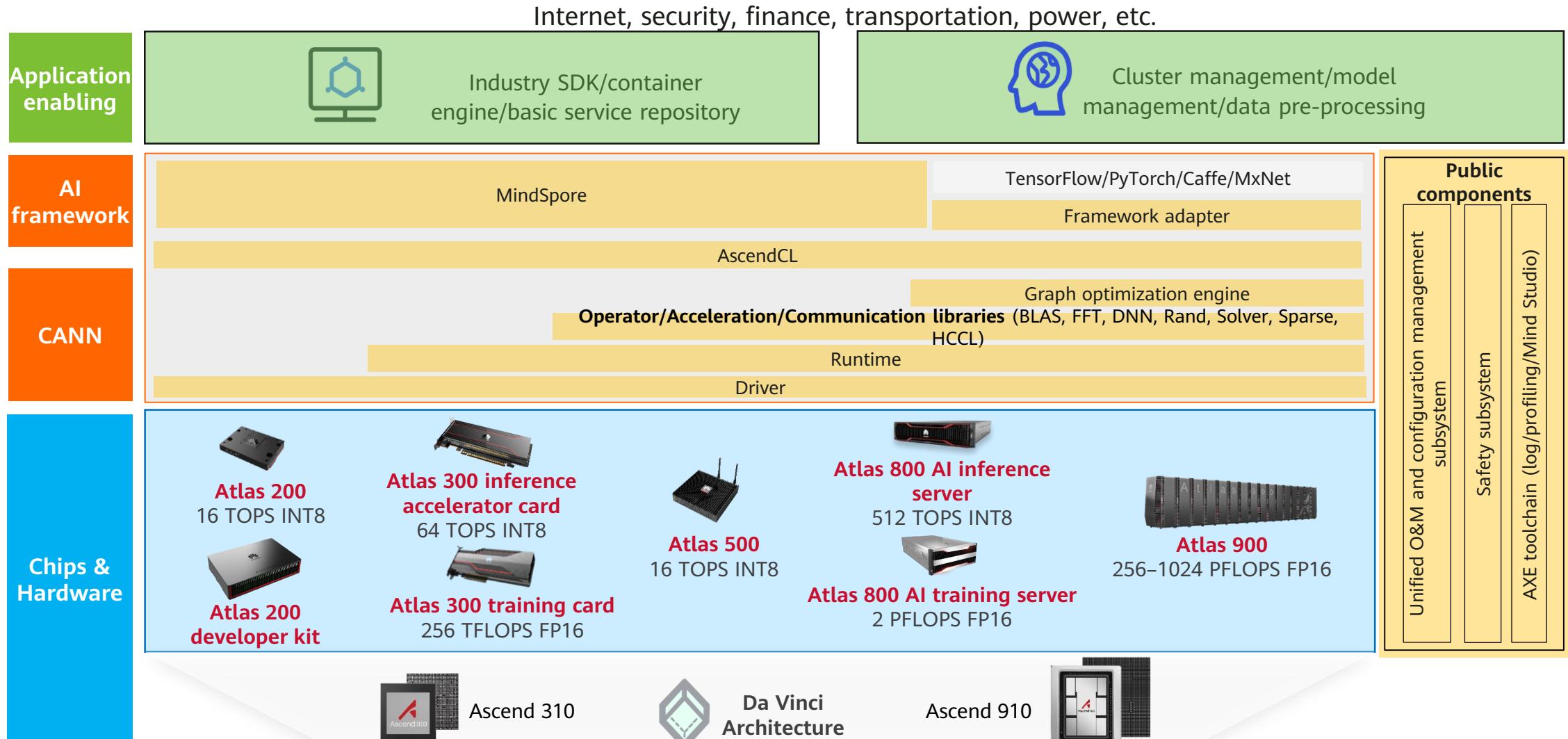
1. Overview of AI Chips
2. Hardware Architecture of Ascend Chips
3. Software Architecture of Ascend Chips
- 4. Huawei Atlas AI Computing Platform**
5. Industry Applications of Atlas

# Overview and Objectives

---

- This section describes the main products of Huawei Atlas AI computing platform, including inference and training.

# Atlas AI Computing Platform Portfolio



# Atlas Accelerates AI Inference



Ascend 310  
AI processor

**Performance improved 7x for terminal devices**



Atlas 200 Developer Kit (DK) AI developer kit  
Model: 3000



Atlas 200 AI accelerator module  
Model: 3000

**Highest density in the industry (64-channel) for video inference**



Atlas 300 AI accelerator card  
Model: 3000

**Edge intelligence and cloud-edge collaboration**



Atlas 500 AI edge station  
Model: 3000

**Powerful computing platform for AI inference**



Atlas 800 AI server  
Model: 3000/3010

# Atlas 200DK: Strong Computing Power and Ease-of-Use



**Full-Stack AI development on  
and off the cloud**

## Developers



Set up a dev environment  
with one laptop  
Ultra low cost for local  
independent environment,  
with multiple functions and  
interfaces to meet basic  
requirements

## Researchers



**Local dev + cloud training  
collaboration**  
Same protocol stack for  
Huawei Cloud and the  
developer kit; training on the  
cloud and deployment at  
local; no modification required

## Startups



**Code-level demo**  
Implementing the algorithm  
function by modifying 10% code  
based on the reference  
architecture; interaction with the  
Developer Community; seamless  
migration of commercial products

# Atlas 200DK: Strong Computing Power and Ease-of-Use

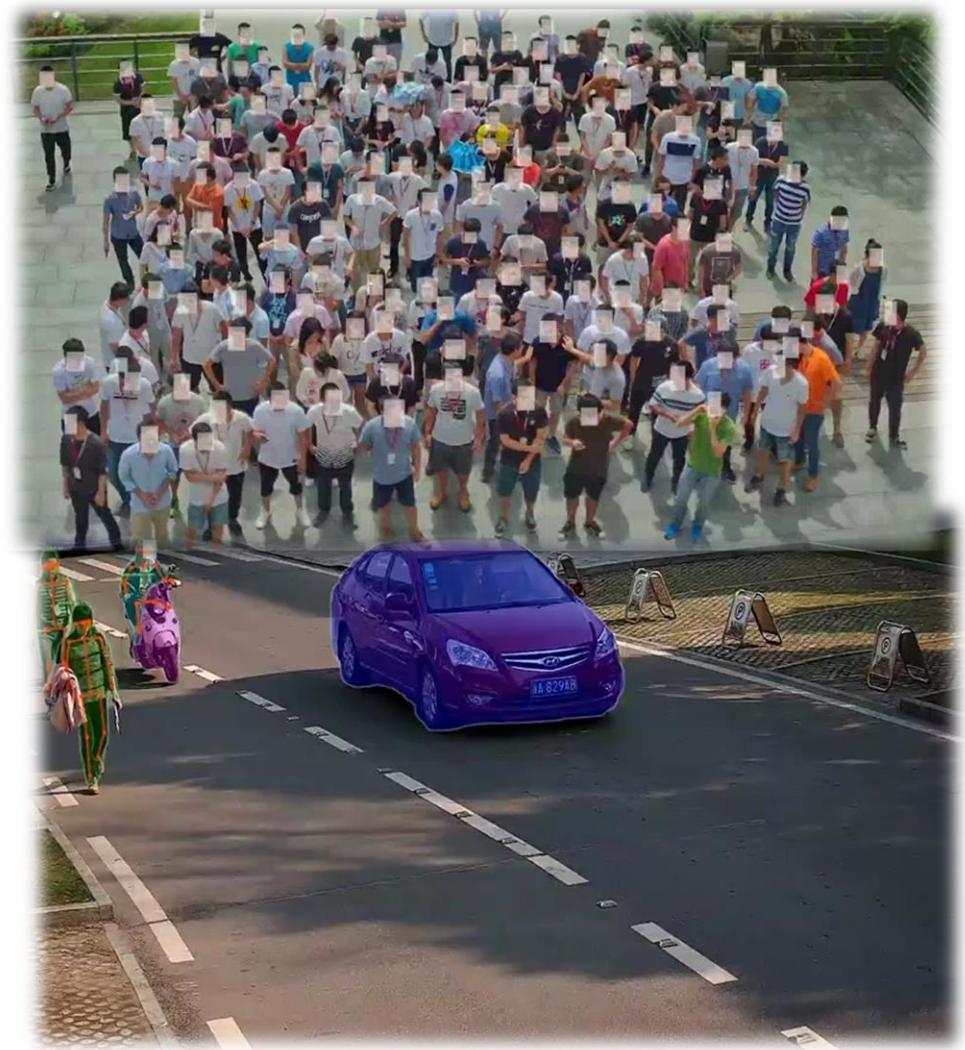


- **16 TOPS INT8, 9.5 w**
- **16-channel HD video real-time analytics, JPEG decoding**
- **4 GB /8 GB memory, four PCIe 3.0 interfaces**
- **Operating temperature: -25° C to +80° C**

**30 → 200**  
Number of faces recognized per frame



- Concurrent running of multiple algorithms on AI camera



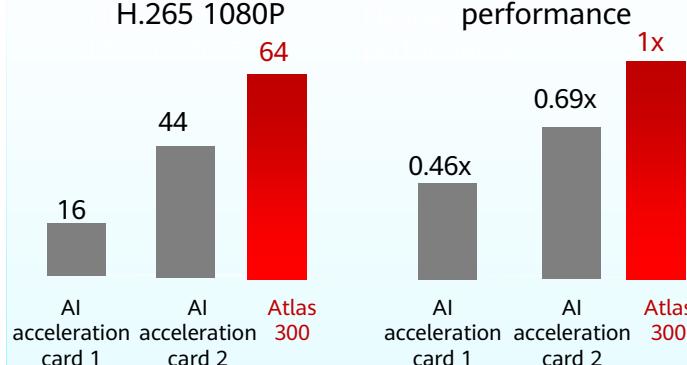
# Atlas 300: Highest-Density, 64-Channel Video Inference Accelerator Card



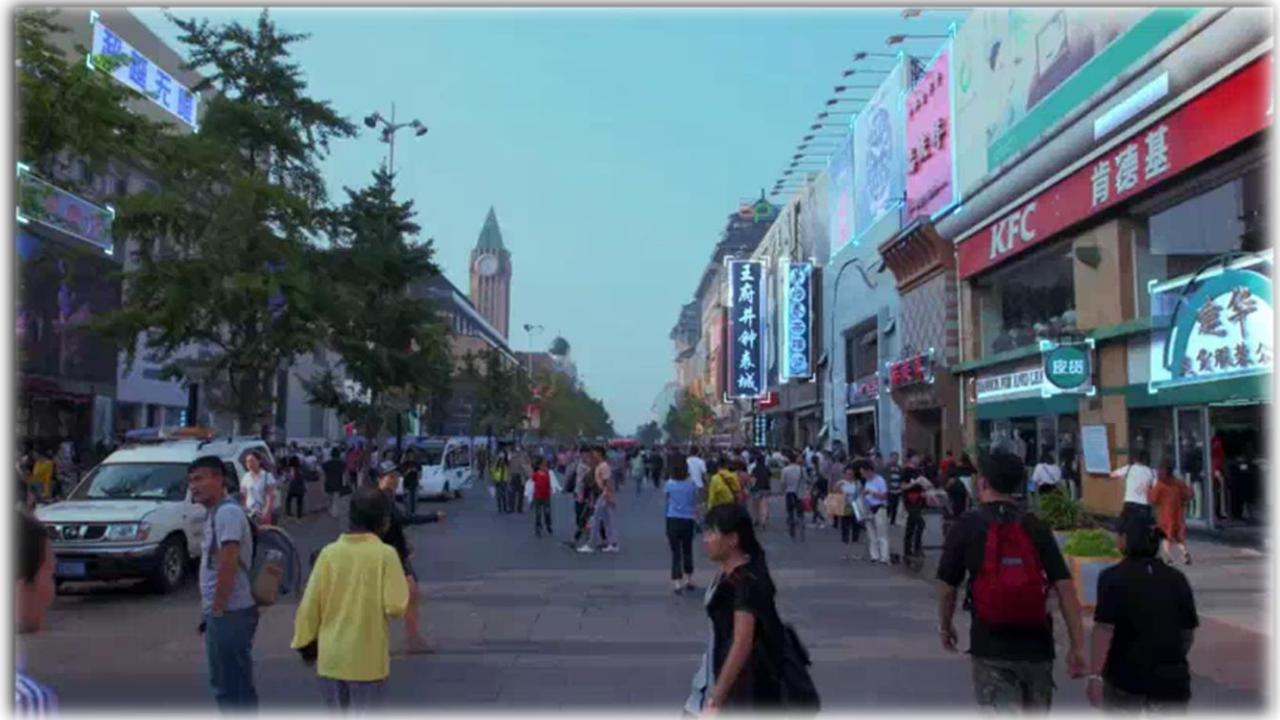
- 64 TOPS INT8, 67 w
- 32 GB memory
- 64-channel HD video analytics in real time

**Model: 3000**

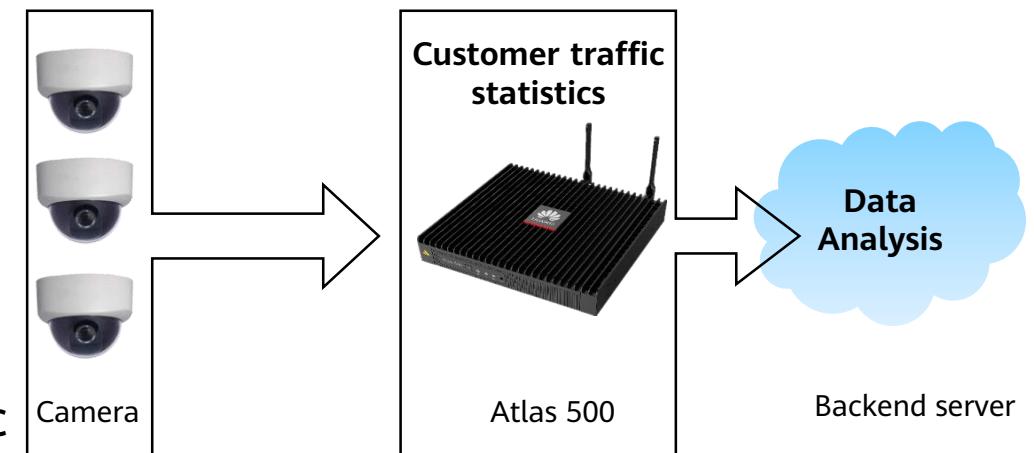
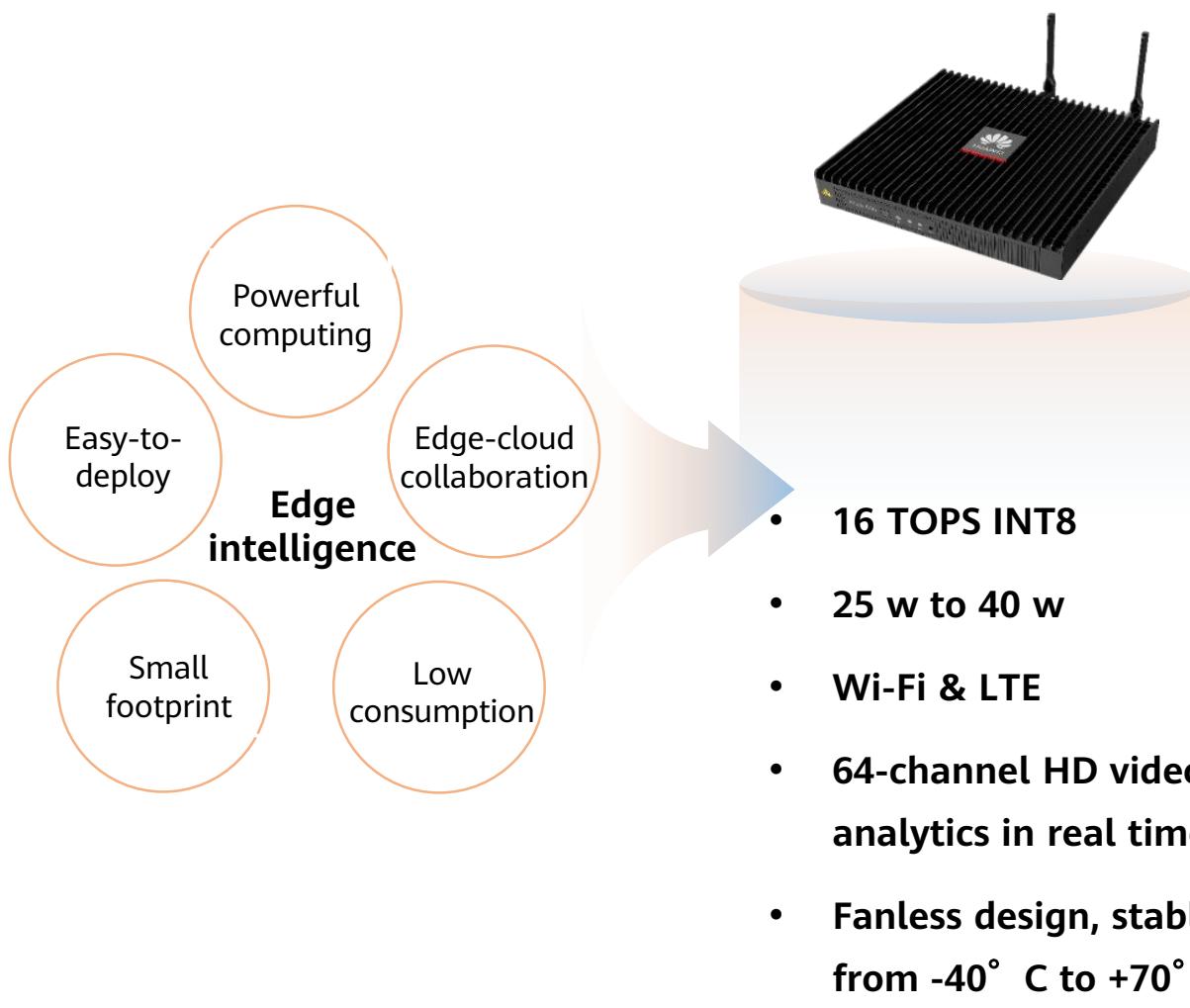
Channels of video decoding  
H.265 1080P



**Video Analytics | OCR | Speech Recognition |  
Precision Marketing | Medical Image Analysis**



# Atlas 500 AI Edge Station



# Atlas 800 AI Server



Model: 3000



Model: 3010

- **An efficient inference platform powered by Kunpeng**
- **Key functions:**
  - 2 Kunpeng 920 processors in a 2U space
  - 8 PCIe slots, supporting up to 8 Atlas 300 AI accelerator cards
  - Up to 512-channel HD video real-time analytics
  - Air-cooled, stable at 5° C to 40° C
- **A flexible inference platform powered by Intel**
- **Key functions:**
  - 2 Intel® Xeon® SP Skylake or Cascade Lake processors in a 2U space
  - 8 PCIe slots, supporting up to 7 Atlas 300/NVIDIA T4 AI accelerator cards
  - Up to 448-channel HD video real-time analytics
  - Air-cooled, stable at 5° C to 35°C

# Atlas Accelerates AI Training

**Training card with ultimate computing power**



**Atlas 300 AI accelerator card  
Model: 9000**

**World's most powerful training server**



**Atlas 800 AI server  
Model: 9000/9010**

**Ascend 910  
AI processor**



**World's fastest AI training cluster**



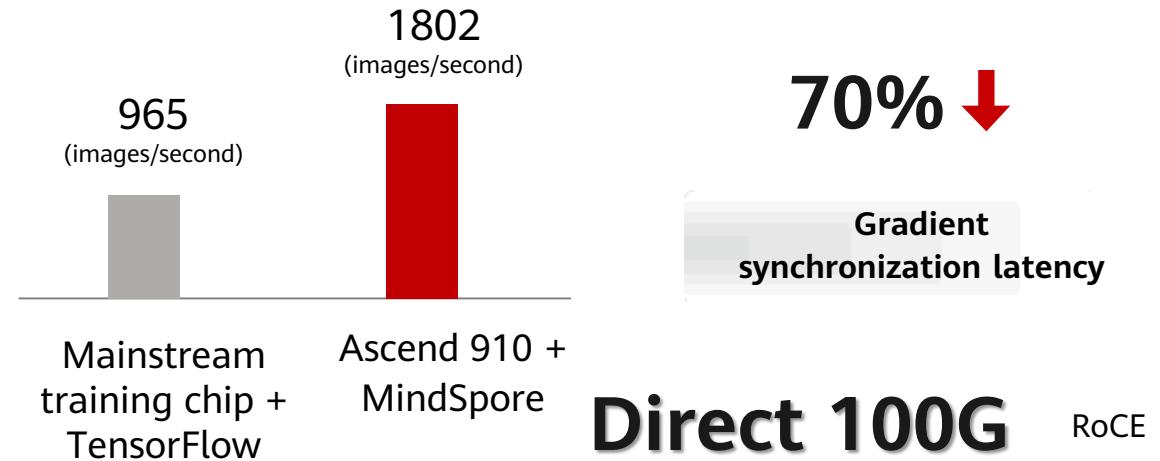
**Atlas 900 AI cluster**

# Atlas 300 AI Accelerator Card: Highest-Performing Accelerator Card for AI Training



Atlas 300  
Model: 9000

**256T** FLOPS FP16  
**2x ↑**  
Computing power per card



- **Test benchmark:**
  - ResNet 50 V1.5
  - ImageNet 2012
  - Optimal batch size respectively

# Atlas 800 Training Server: Industry's Most Powerful Server for AI Training



Atlas 800

Model: 9000

2.5x ↑

Density of computing power

**2P** FLOPS/4U

25x ↑

Hardware decoder

**16384**  
(1080p decoding) images/second

1.8x ↑

Perf./Watt

**2P** FLOPS/5.6kW

# Atlas 900 AI Cluster: Fastest Cluster for AI Training



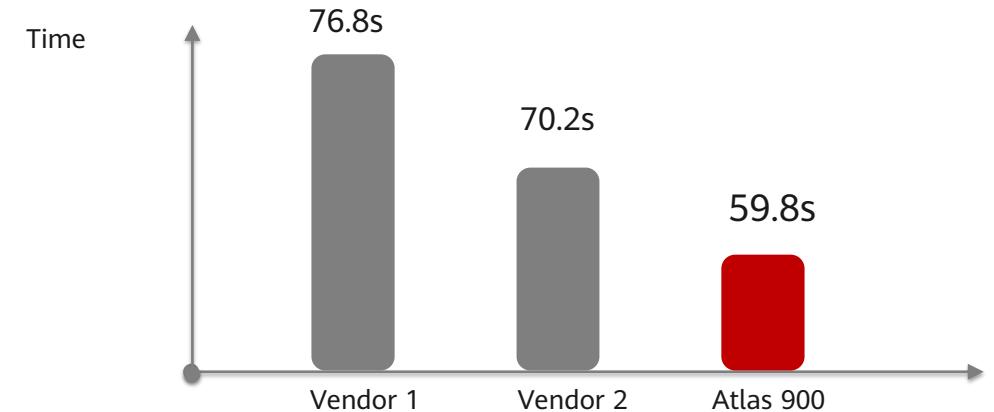
Atlas 900

256-1024 PFLOPS FP16

**Industry-leading computing power | Best cluster network | Ultimate heat dissipation**

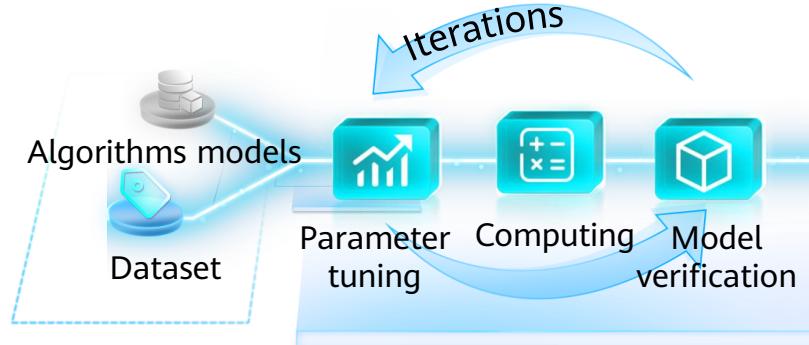


Shortest time consumption: 59.8s

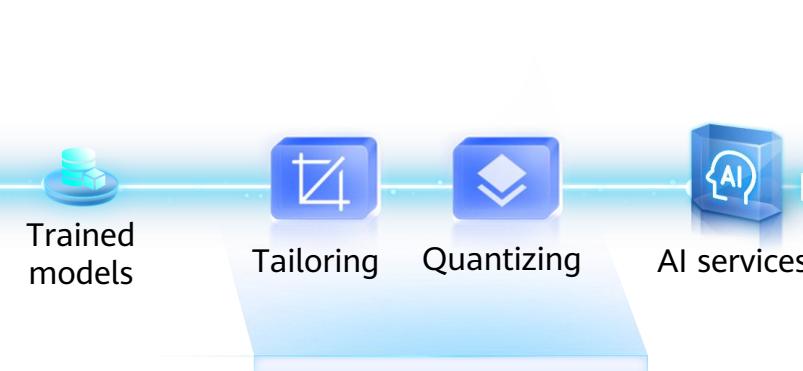


- **Test benchmark:**
  - Benchmark: ResNet-50 V1.5 model, ImageNet-1k dataset
  - Cluster: 1024 Ascend 910 AI processors
  - Accuracy: 75.9%

# Atlas Deep Learning System Accelerates AI Model Training and Builds Extensive Applications



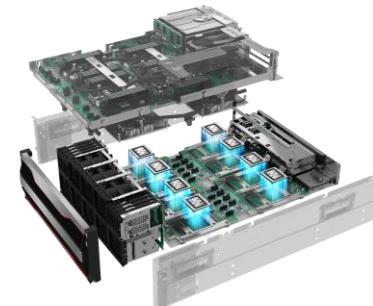
Model training



Model deployment



Atlas 300 training  
acceleration card  
Model: 9000



Atlas 800 AI  
training server  
Model: 9000



Atlas 900 AI cluster



Video analysis



Gene research



Automated  
driving

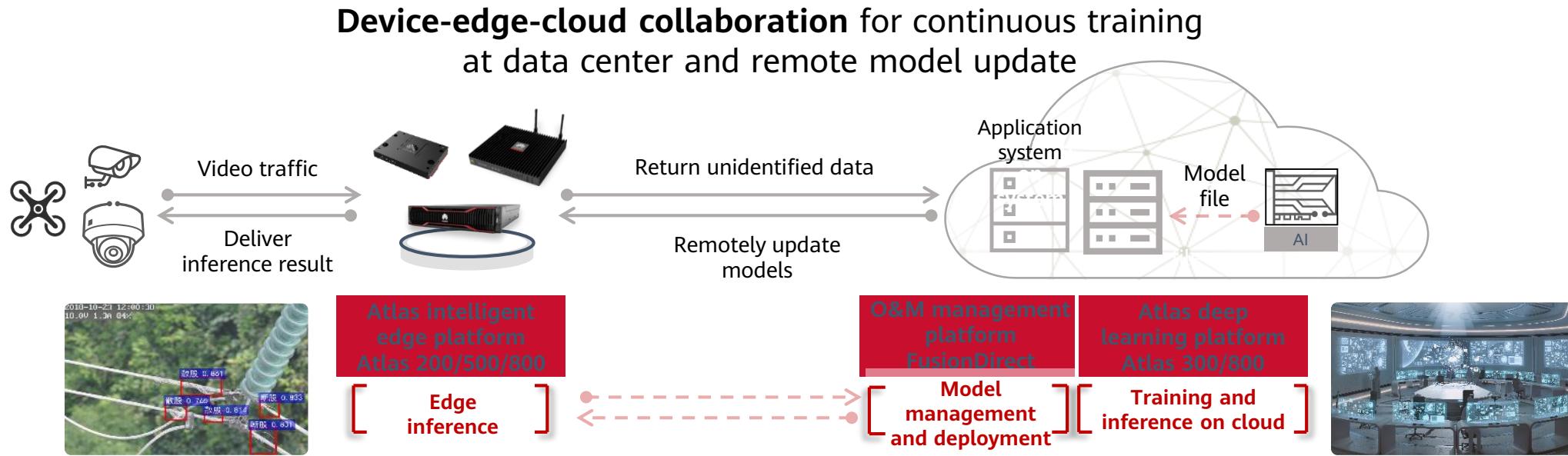


Weather  
forecast



Oil  
exploration

# Device-Edge-Cloud Collaboration Enables the Ultimate Development and User Experience



## Centralized development

Atlas

- Centralized development architecture based on Da Vinci and CANN, **develop once, deploy everywhere**

Industry solution

- Edge and data centers use different development architectures. **Models cannot transfer freely, requiring secondary development.**

## Centralized O&M

- FusionDirector manages up to **50,000 nodes**, **manages central and edge devices**, and remotely pushes models and upgrades devices

- No O&M management tools**; provides only APIs, so customers need to develop APIs by themselves.

## Enhanced security

- Transmission channel encryption
- Model encryption, double assurance**

- No encryption/decryption engine**; models are not encrypted.

# Section Summary

---

- This section introduces products of Huawei Atlas AI computing platform, mainly inference products including Atlas 200 DK, Atlas 200 AI accelerator module, Atlas 300 AI accelerator card, Atlas 500 AI edge station, and Atlas 800 AI server. The computing platforms used for training include Atlas 300 AI accelerator card, Atlas 800 AI server, and Atlas 900 AI cluster.

# Contents

1. Overview of AI Chips
2. Hardware Architecture of Ascend Chips
3. Software Architecture of Ascend Chips
4. Huawei Atlas AI Computing Platform
- 5. Industry Applications of Atlas**

# Overview and Objectives

---

- This section introduces industry application scenarios of the Atlas AI computing platform.

# Electric Power: Industry's First Intelligent Unattended Inspection Solution, with 5x Efficiency

The diagram illustrates the architecture of the Electric Power Intelligent Unattended Inspection Solution. It starts with a camera mounted on a power pole, connected via a fiber optic cable to a local unit. This unit includes a solar panel and an Atlas 200 module. The system performs front-end analysis and returns alarms. The data then flows through a surveillance and management platform, which includes AI model training (Atlas 300/800) and high computing power. The platform also handles model optimization and remote upgrade, along with real-time analytics. The overall average system consumption is 1W. The solution is highlighted as having "Intelligent analysis at the edge with ultra-low consumption".

Front-end analysis and return alarms

Model optimization and remote upgrade

Average system consumption 1W

Real-time analytics Once per minute

Surveillance and management platform

AI model training

Atlas 300/800

High computing power

Intelligent analysis at the edge with ultra-low consumption

Power consumption 80%↓ Ideal for edge computing

Power consumption 15W

Computing power 2.X

Power consumption 3W

Computing power 16TOPS

Huawei Other vendors

CHINA SOUTHERN POWER GRID

Manual inspection

AI inspection

Efficiency 5x+ up

System cost 30% down

Insulator

2019-08-19 17:53:58

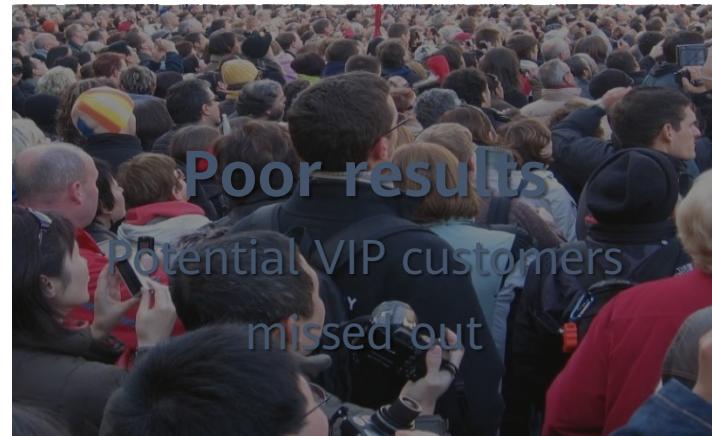
14.60 -1.68 0.92 290

220kV 鹅丽甲乙线 N31

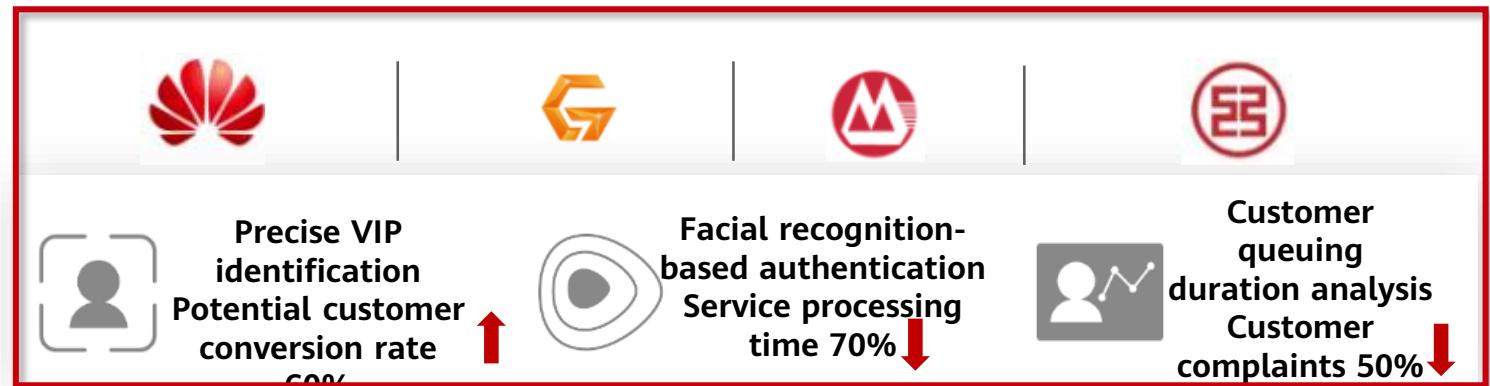
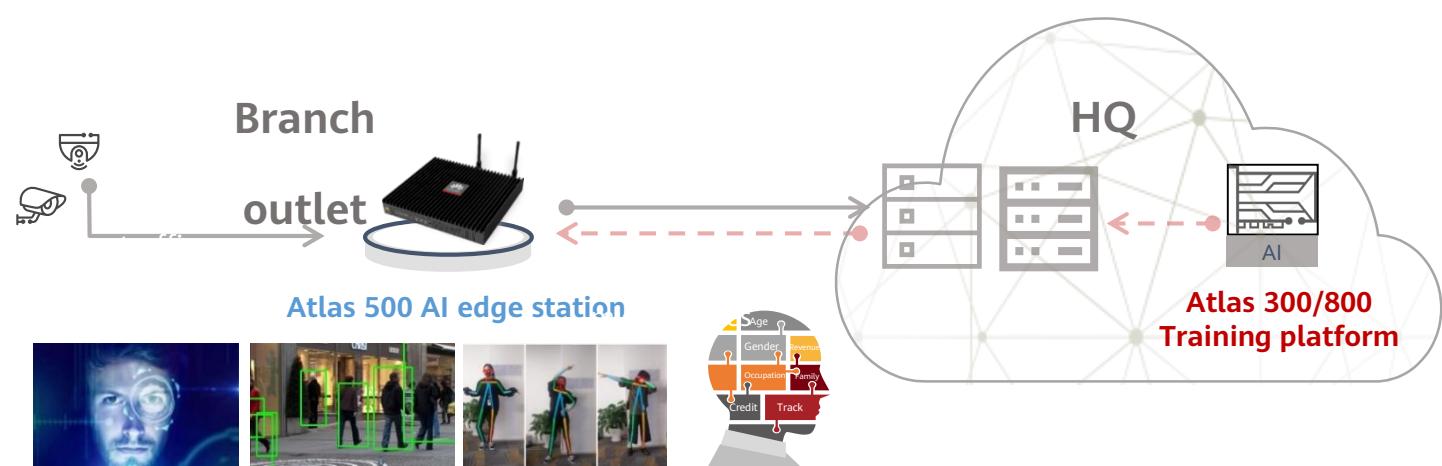
X1.0

# Finance: AI Enables Smart Branches for Banks

••• **Past:** Customer experience needs improvement



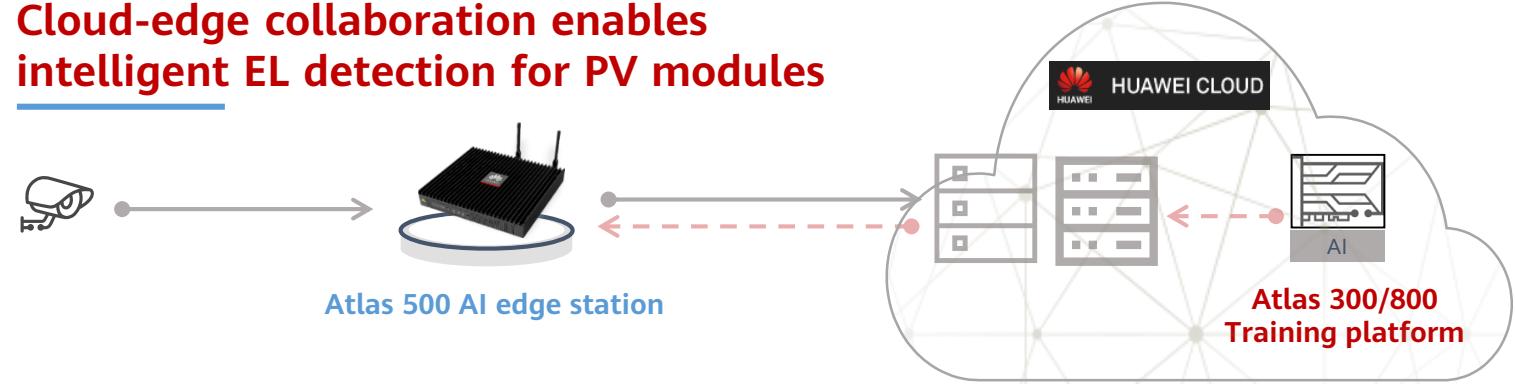
**Now:** cloud-edge collaboration, smart finance



# Manufacturing: AI Enables Intelligent Production Line with Machine Vision



**Cloud-edge collaboration enables intelligent EL detection for PV modules**



## Manual inspection

Unstable results	Low production efficiency	Discontinuous process	High labor costs
------------------	---------------------------	-----------------------	------------------

Number of defective battery chips  
**2**

Detection duration  
*about 5s*

Accuracy  
**33.33%**

## Smart inspection

Zero omission	High production efficiency	Cloud-edge synergy	Reduced labor costs
---------------	----------------------------	--------------------	---------------------

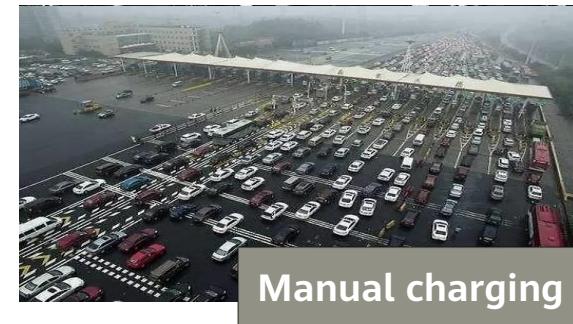
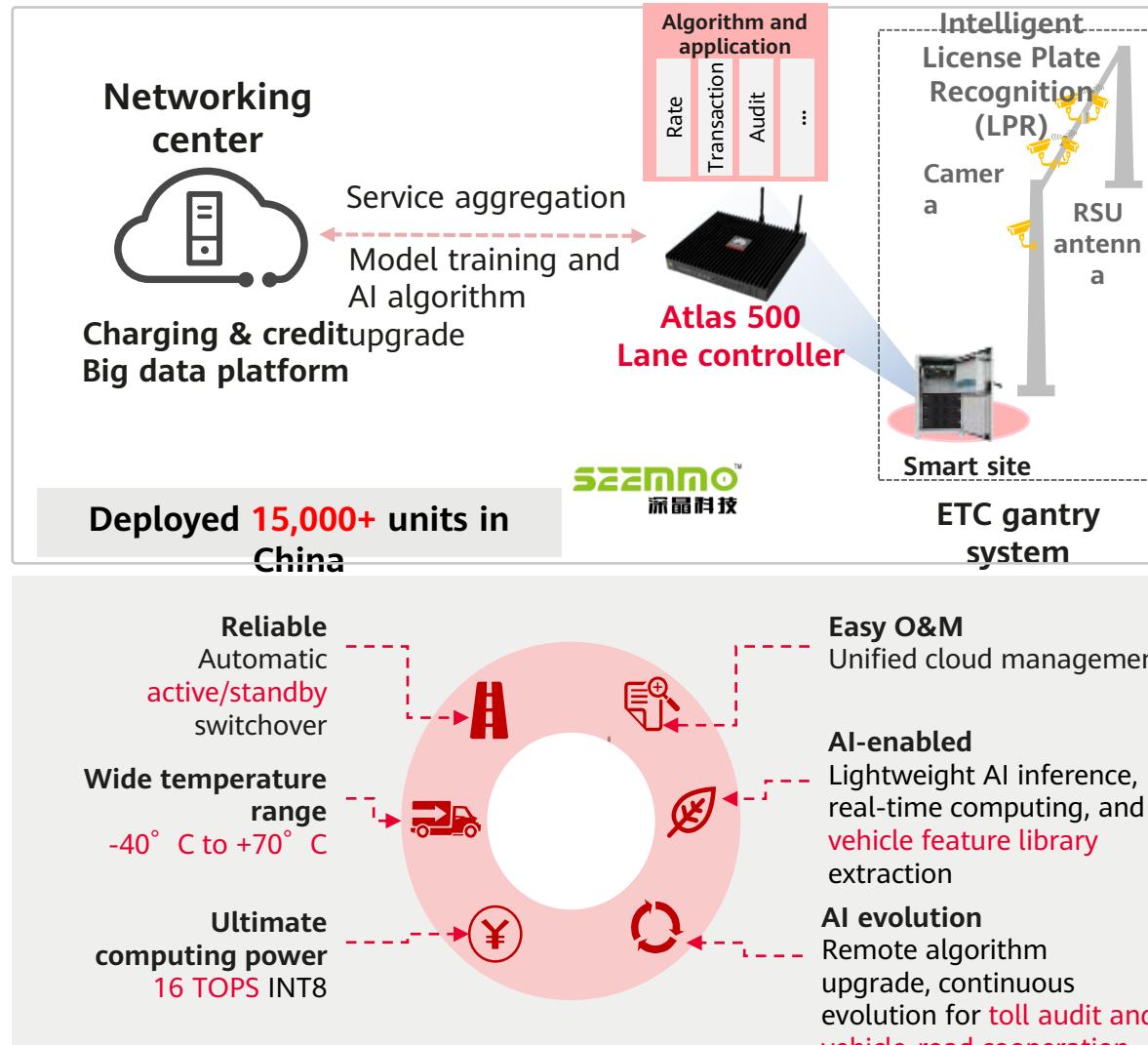


Defective battery chip  
**6**

Detection duration  
**1.36s**

Accuracy  
**100%**

# Transportation: AI Smooths Highways with 5x Efficiency Boost

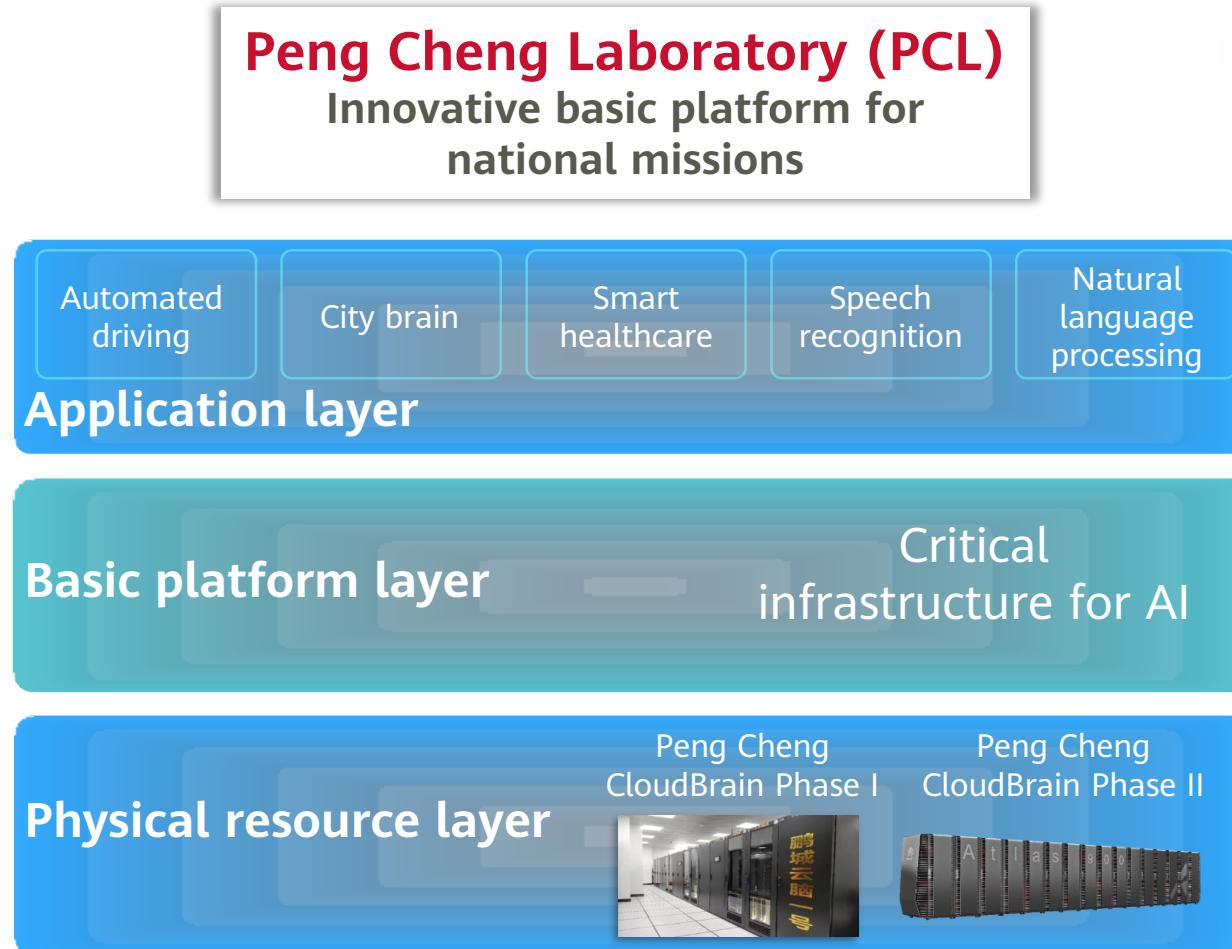


- Low efficiency
- Long queuing time

• Passing efficiency 5X↑  
Saving energy and reducing emission

- Proactive security control
- Road cooperation management
- Autonomous vehicle driving

# Supercomputing: Atlas Helps PCL Build CloudBrain Phase II



Peng Cheng CloudBrain Phase II mainly built **Atlas 900**,  
**the world's fastest training cluster**

**Ultimate computing power**

Level-E AI computing power

**Top cluster network**

HCCL communication supports 100 TB/s non-blocking parameter plane networking

**Ultimate energy efficiency**

AI cluster PUE < 1.1

# Attract More Developers Based on the Ascend Developer Community



- Developer-centric enabling platform
- <https://ascend.huawei.com/home>

## Annual technical salon

- Held in **10+** cities
- 20+ senior trainers
- Dozens of speeches

## Developer contest

- 1,500+ teams
- Annual prize of over RMB1 million
- Equal opportunities for enterprises and universities

## Developer support

- Public cloud vouchers
- Free certification course tickets
- Free Atlas developer kits

# Summary

- This chapter describes the products of Huawei Atlas computing platform and helps you to understand the working principles of Huawei Ascend chips. It focuses on the hardware and software architectures of Ascend chips and application scenarios of the Atlas AI computing platform.

# Quiz

1. What are the main applications of Ascend 310? (    )
  - A. Model inference
  - B. Model training
  - C. Model building

# Recommendations

---

- Ascend community:
  - <https://ascend.huawei.com>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# Huawei Open AI Platform for Smart Devices



# Foreword

---

- Huawei HiAI is an open artificial intelligence (AI) capability platform for smart devices, which adopts a "chip-device-cloud" architecture, opening up chip, app, and service capabilities for a fully intelligent ecosystem. This assists developers in delivering a better smart app experience for users by fully leveraging Huawei's powerful AI processing capabilities.

# Objectives

After this course, you will be able to:

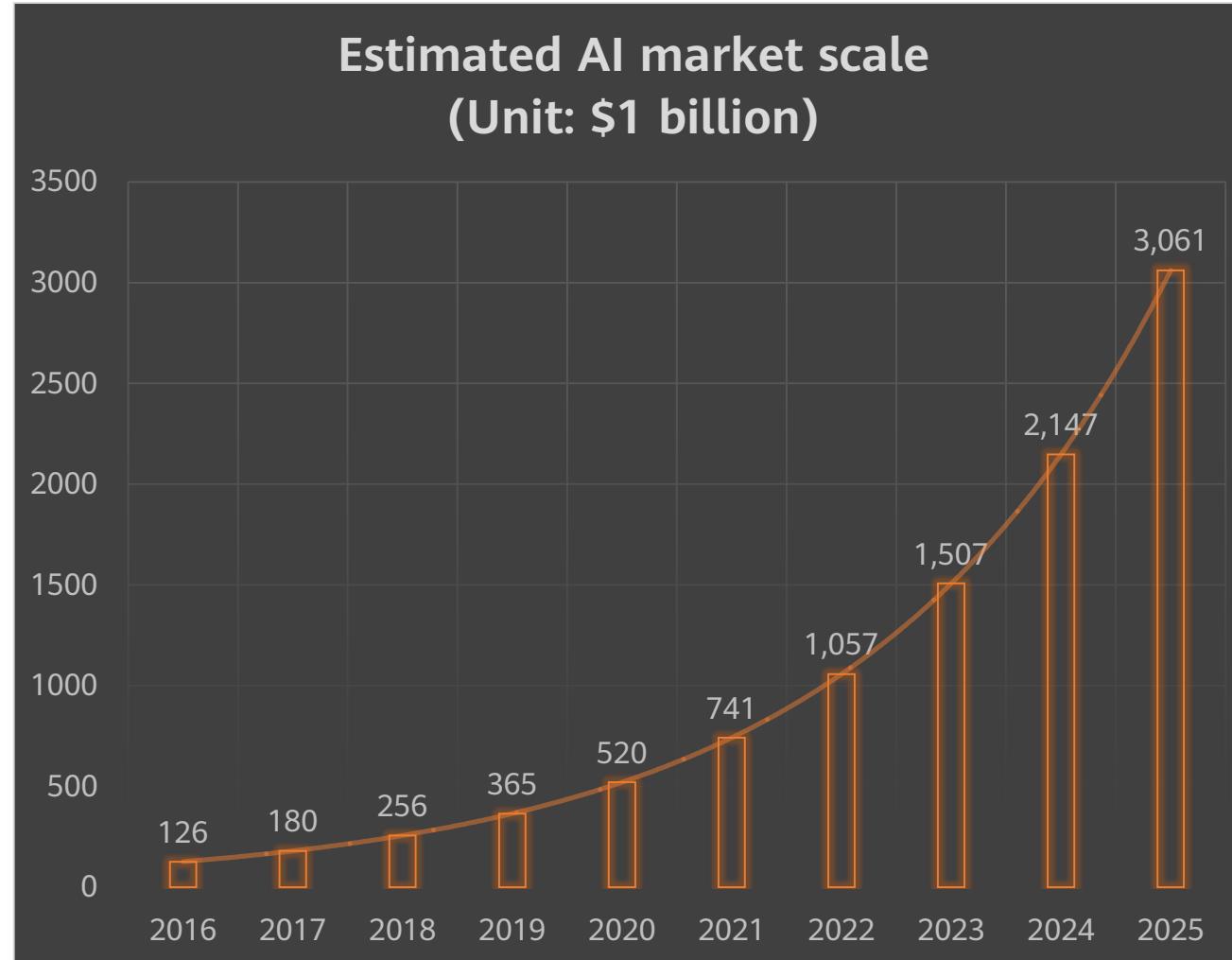
- Master the usage of Huawei HiAI platform.
- Understand the powerful functions of Huawei HiAI platform.

# Contents

---

- 1. AI Industry Ecosystem**
2. Huawei HiAI Platform
3. Developing Apps Based on Huawei HiAI Platform

# Huge Opportunities: Foreseeable AI Ubiquity in a \$3 Trillion Market



Involved industries: automobile, finance, consumer goods and retail, medical care, education, manufacturing, communications, energy, tourism, culture and entertainment, transportation, logistics, real estate, and environmental protection

Computing power breakthrough



Algorithm breakthrough

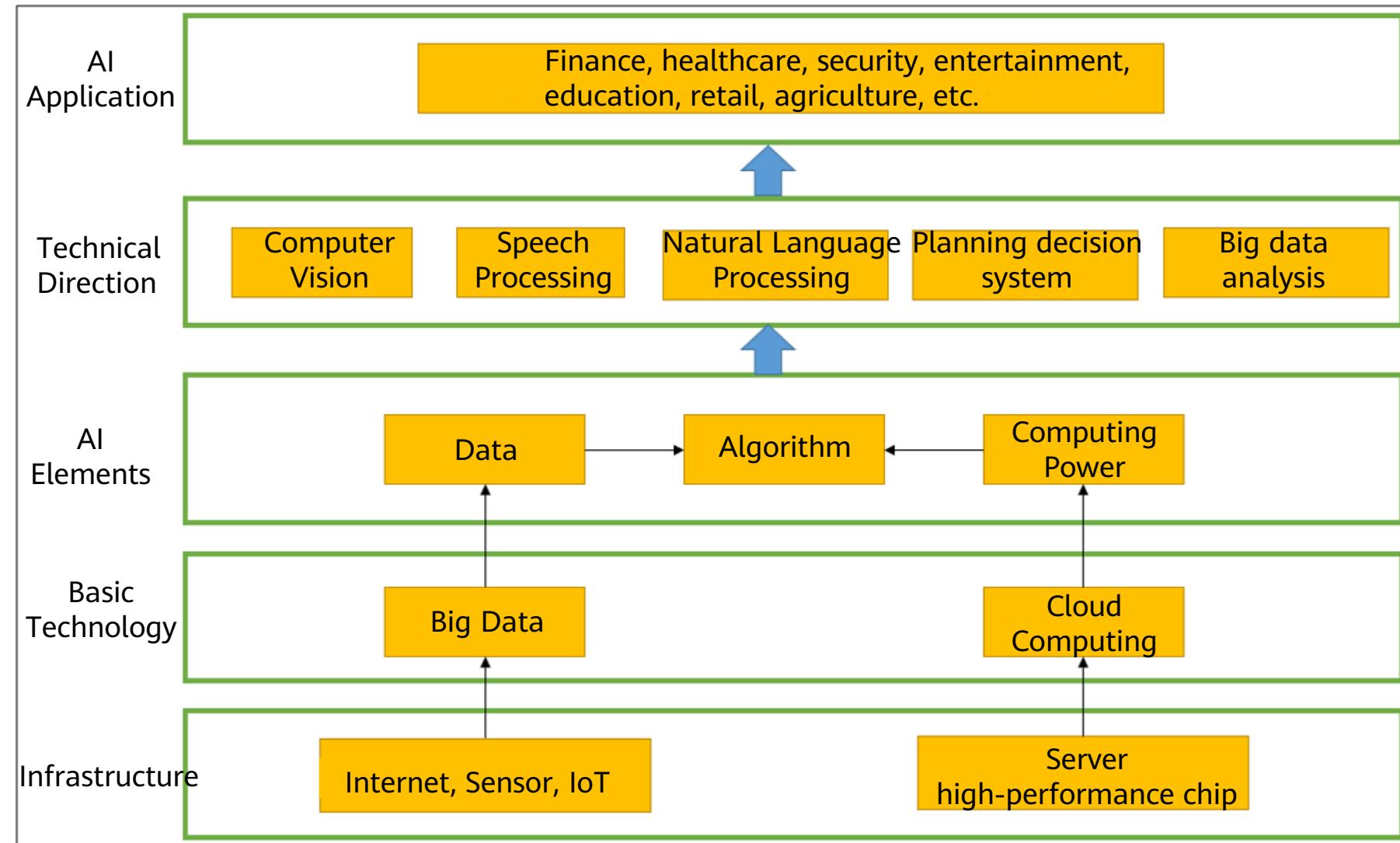


Data breakthrough



Data source: Forrester, Transparency Market Research, Chinese Association for Artificial Intelligence, and Roland Berger

# Architecture of the AI Application Platform



# Challenges in AI Capability Development and Application

## High thresholds

Demanding requirements



6 months: ML & DL  
2 months: Statistics  
4 months: Linear algebra  
3 months: Calculus

15 months

## Low efficiency

Long training cycles



Data collection and cleansing  
Model training and optimization  
Customized experience improvement

3–8 months

## Diverse requests

Fragmented experience



The entire training cycle must be repeated for each scenario.  
Lack of inheritability or transferability

N-fold workload

## Slow iteration

Limited capability enhancement

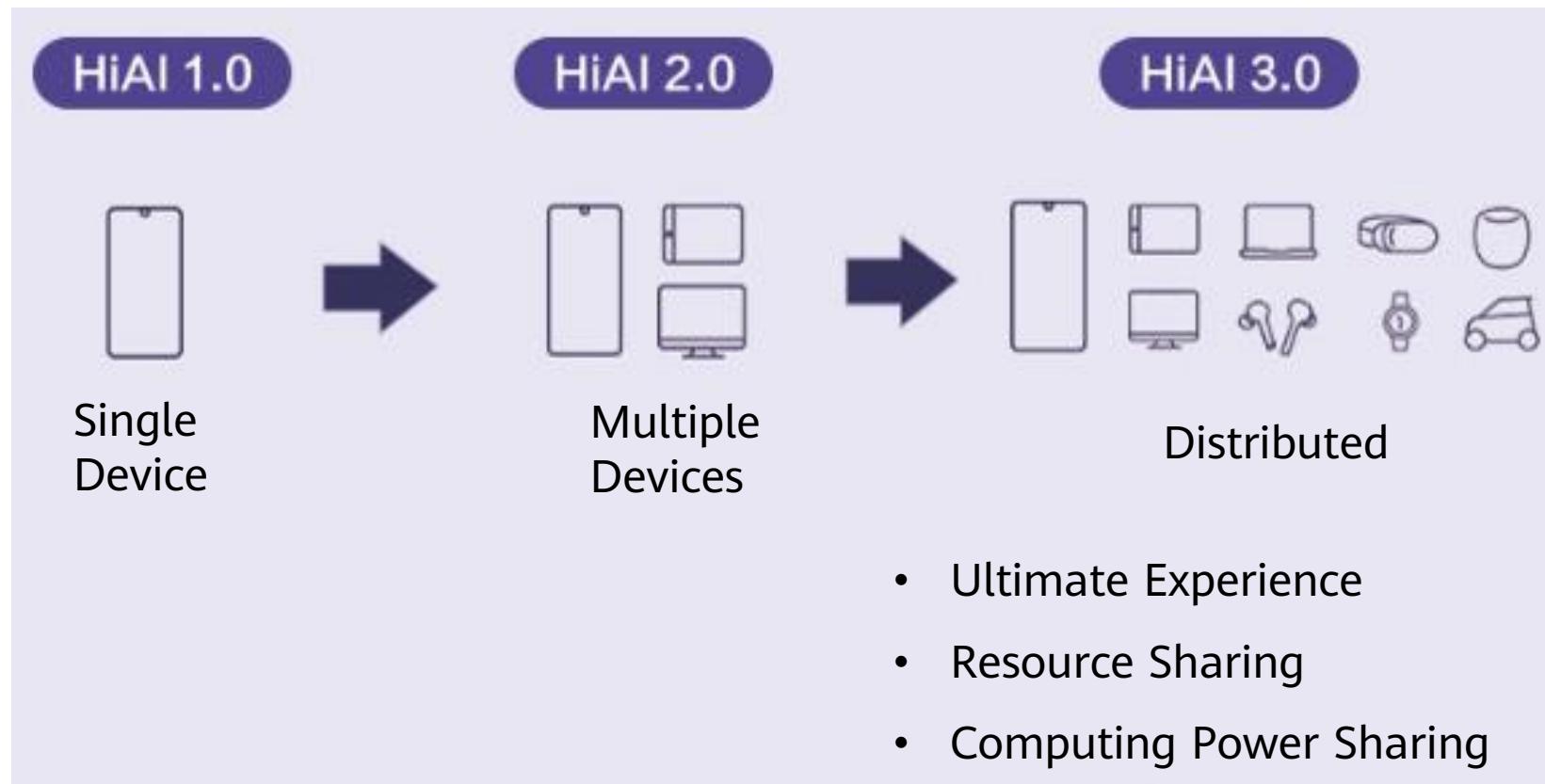


Difficulties in model upgrade  
Difficulties in valid data acquisition

# Contents

1. AI Industry Ecosystem
- 2. Huawei HiAI Platform**
3. Developing Apps Based on Huawei HiAI Platform

# HiAI 3.0 - Enabling Ultimate Experience in All-Scenario Smart Life



- 4000+ partners
- Over 96 million daily active users
- Over 600 billion monthly calls

# Huawei HiAI 3.0: Enabling Distributed All Scenarios by AI



Cloud 1000+ atomized services

## Huawei HiAI Service

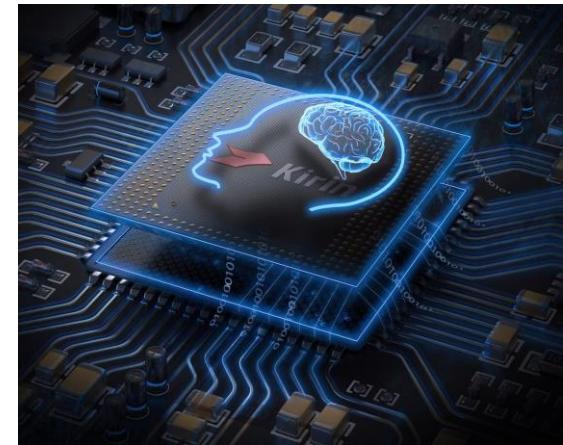
Service capability openness for mutual benefits  
Push services based on user requirements in an active manner.



Device 40+ application programming interfaces (APIs)

## Huawei HiAI Engine

AI capability openness for simplicity  
Integrate multiple AI capabilities into apps simply, making apps smarter and more powerful.



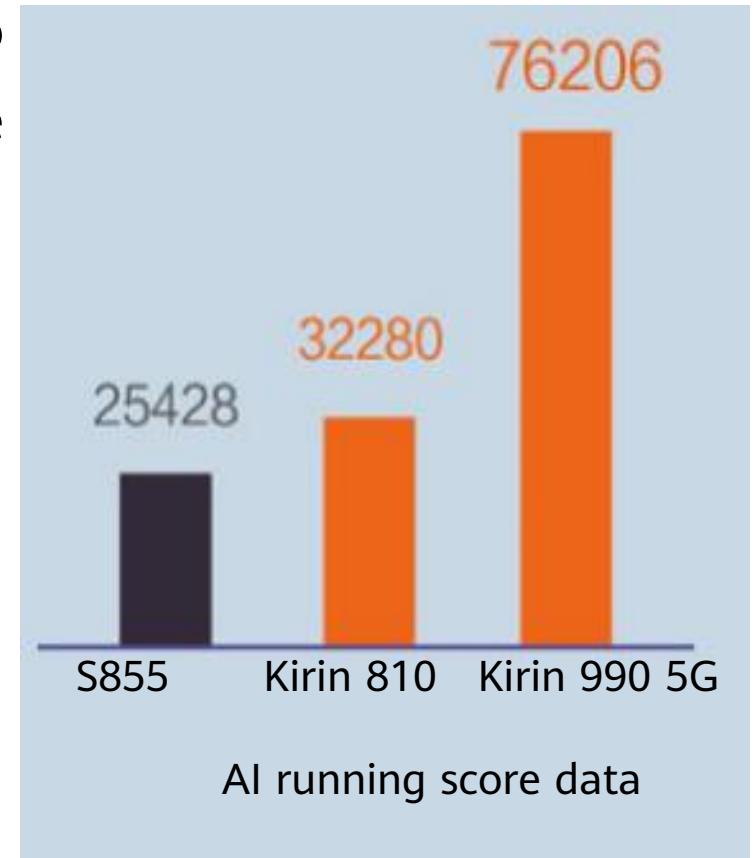
Chip 300+ operators

## Huawei HiAI Foundation

Chip capability openness for high efficiency  
Quickly convert and migrate existing models to obtain optimal performance based on heterogeneous scheduling and network process unit (NPU) acceleration.

# HiAI Foundation

- HiAI Foundation APIs constitute an AI computing library of a mobile computing platform, enabling developers to efficiently compile AI apps that can run on mobile devices.
  - Leveraging high performance and high precision of Kirin chips, better device-end AI performance will be delivered by more powerful computing power.
  - Support the largest number of operators (300+) in the industry and more frameworks, greatly improving flexibility and compatibility.
  - The Honghu, Kirin, and AI camera chips enable AI capabilities for more devices.



# HiAI Engine

- HiAI Engine opens app capabilities and integrates multiple AI capabilities into apps, making apps smarter and more powerful.
  - Provide handwriting recognition and dynamic gesture recognition capabilities, with 40+ underlying APIs.
  - Computer vision and speech recognition will develop toward a distributed mode, assisting developers in delivering more all-scenario smart life experience.

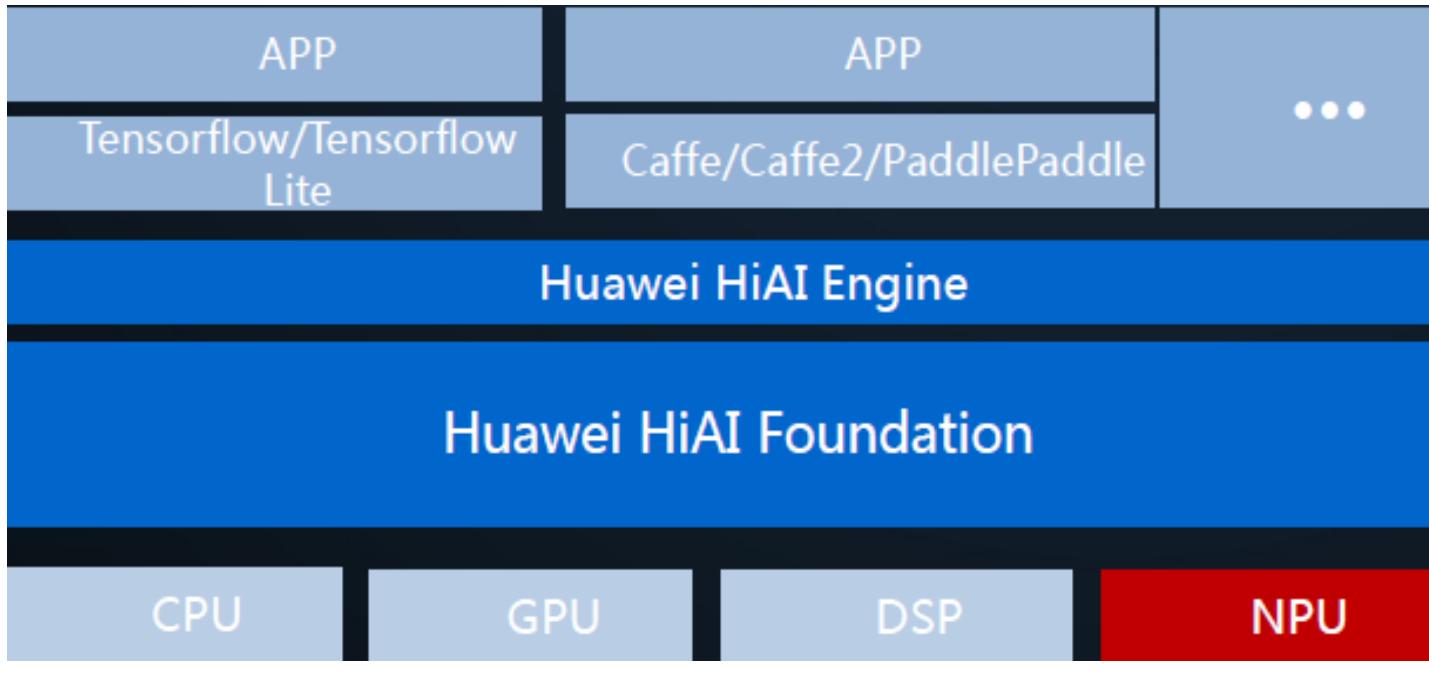


Distributed AI helps sports healthy  
and safe driving

# HiAI Service

- HiAI Service enables developers to reuse services on multiple devices, such as mobile phones, tablets, and large screens, with only one service access, efficiently implementing distribution.

# Architecture of the HiAI Mobile Computing Platform



Supports diverse mainstream front-end frameworks.

Provides various upper-layer function service APIs to ensure efficient running on mobile devices.

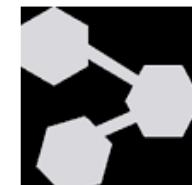
Allows flexible scheduling of heterogeneous resources, meeting developers' demand to accelerate neural network model computing and operator computing



Tool chain



Comprehensive documents



Different types of APIs



Source codes enabling quick start

# What can apps benefit from Huawei HiAI?



**Real time**



**Ready-to-use**



**Stability**



**Security**



**Lower cost**

# Huawei HiAI + Ctrip Help You Take Poetic Photos



# Real-time NPU AI Image Segmentation



# NPU+AI Human Image Matting for Flexible Switchover in Multiple Scenarios



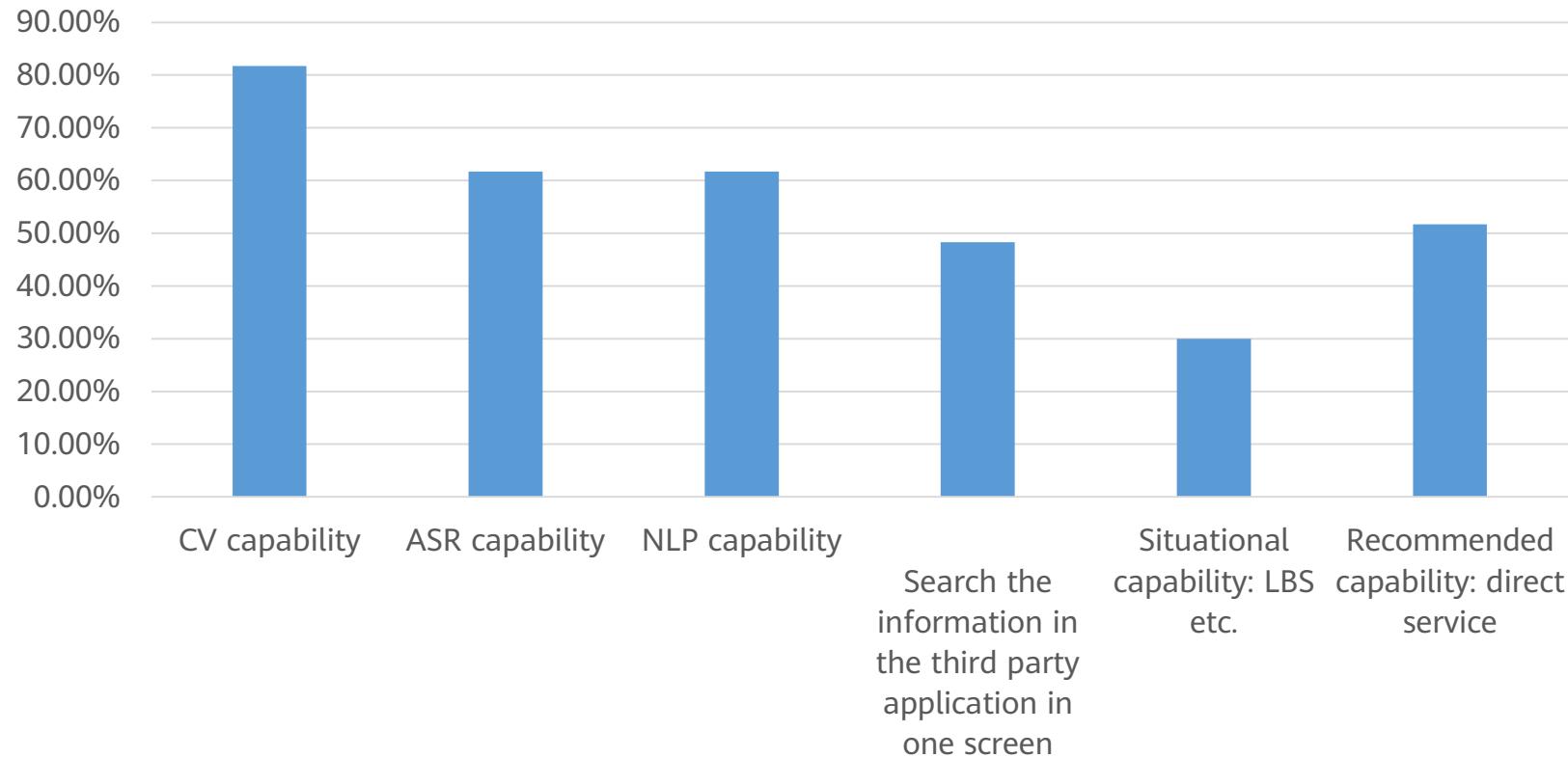
- <https://www.bilibili.com/video/av66898567/>

# AI Capability Provider, Accelerating Application Development

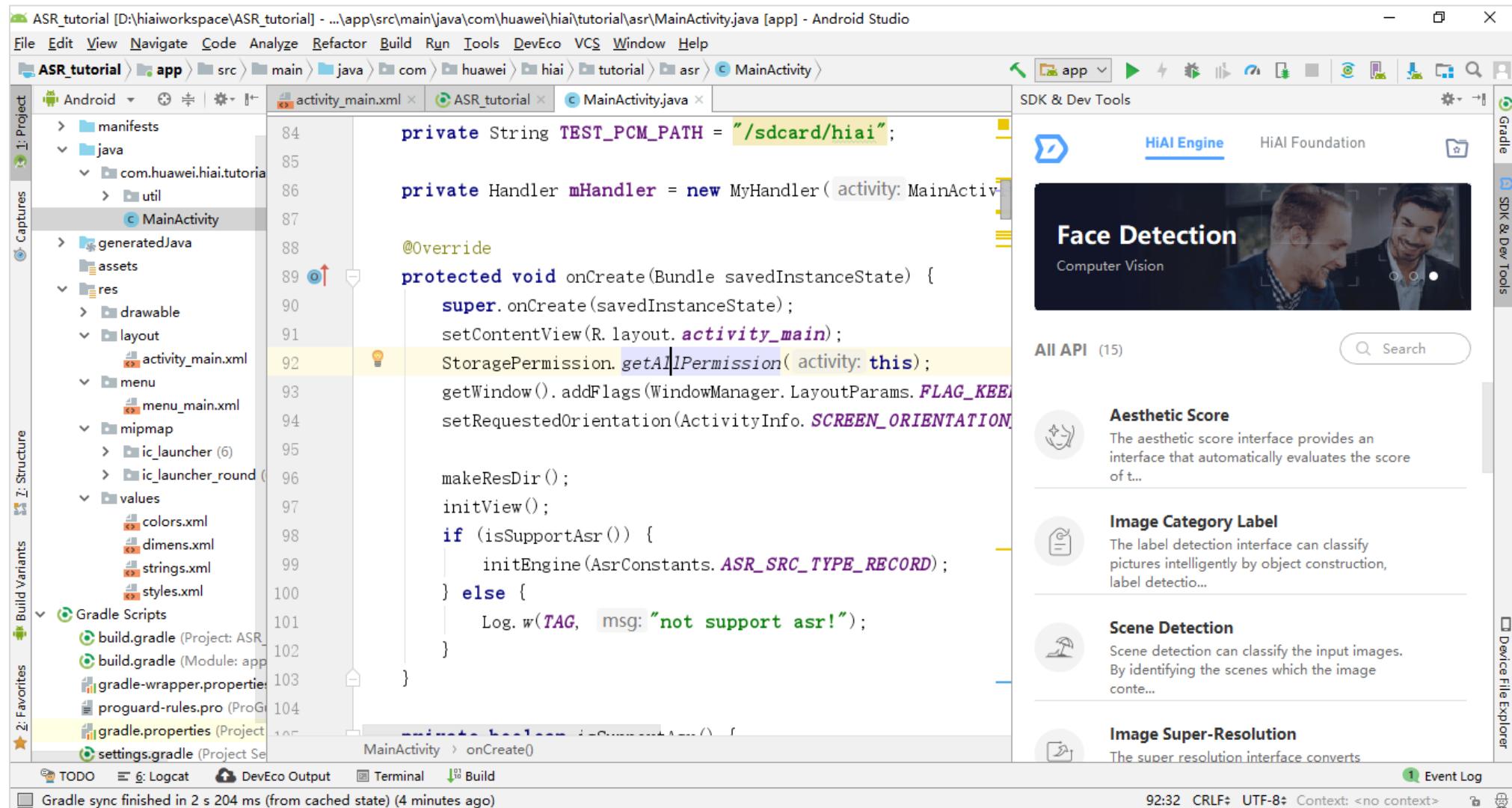
Short video and live streaming	Social media	Augmented reality (AR)	Photo taking and retouching	Shopping	Translation and text processing
Facial recognition Gesture recognition Portrait segmentation Posture recognition Video style Voice control Intelligent depth of field control Image scene recognition	Photo categorization Image recognition Image super-resolution (SR) Sensitive data recognition	Contextual awareness Voice control Depth estimation Light estimation	Beautification Image enhancement Aesthetics scoring Album generation Photographing by voice Photographing by gesture	QR code scan Direct service delivery and recommendation ID card recognition Bank card recognition Product identification	Translate by taking a photo Optical character recognition (OCR) Word splitting Named entity recognition Text emotion recognition Intelligent text reply Text and image SR
Computer vision (CV), automatic speech recognition (ASR)	CV, natural language understanding (NLU)	ASR, CV	CV	CV	NLU, CV, ASR

# AI Capability Provider, Accelerating Application Development

Research results of developers' demands for HiAI capabilities: more than 60% pay attention to CV, ASR, NLU



# Comprehensive Tools for Developers



# Contents

1. AI Industry Ecosystem
2. Huawei HiAI Platform
- 3. Developing Apps Based on Huawei HiAI Platform**



# Huawei HiAI Helps Deaf and Mute People



# Next Generation Mobile Experience with Huawei HiAI



Fast

Simple

Mutual Benefit



# Connecting Developers and Stimulating Innovation to Achieve Win-Win ecosystem

## Offline connection for in-depth communication

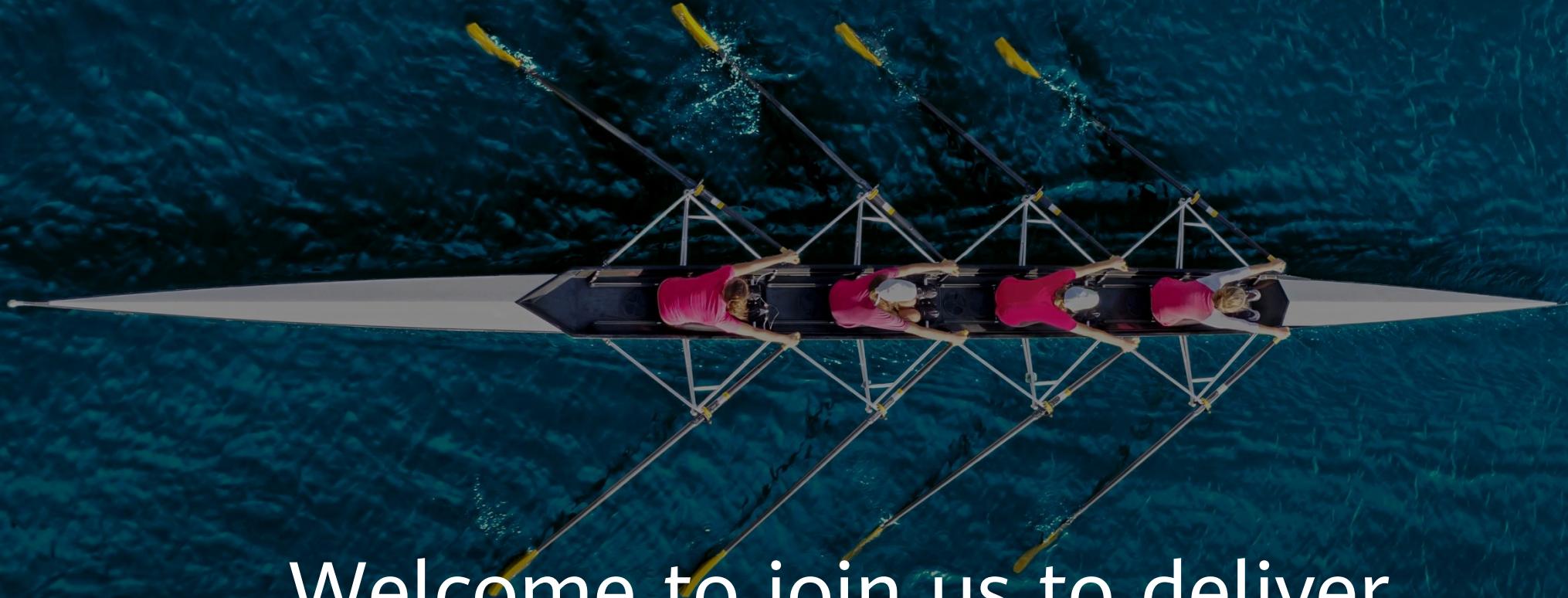
- Salons in cities
- HiAI open courses
- Technical symposiums

## USD1 billion investment, stimulating innovations in all scenarios

- Openness and innovation of device capabilities
- All-scenario digital service innovation
- Cloud service ecosystem co-construction

## Innovation competition for continuous development

- AI application innovation contest
- Creativity contest for future application
- AR application innovation contest



Welcome to join us to deliver  
ultimate AI user experience

# Summary

- We believe that AI can make life better by bringing unprecedented convenience for both back end and devices. However, this requires actual application scenarios that allow more enterprises and developers to play a part in improving user experience substantially. Huawei is willing to work with partners to jointly promote intelligent transformation of industries with more developers and enterprises based on the HiAI3.0 platform.

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



# HUAWEI CLOUD Enterprise Intelligence Application Platform



# Objectives

On completion of this course, you will be able to:

- Know the HUAWEI CLOUD enterprise intelligence (EI) ecosystem and services.
- Know the Huawei ModelArts platform and how to perform operations on the platform.

# Contents

- 1. Overview of HUAWEI CLOUD EI**
2. ModelArts
3. HUAWEI CLOUD EI Solutions

# HUAWEI CLOUD EI Services

- HUAWEI CLOUD EI is a driving force for enterprises' intelligent transformation. Relying on AI and big data technologies, HUAWEI CLOUD EI provides an open, trustworthy, and intelligent platform through cloud services (in mode such as public cloud or dedicated cloud). It allows enterprise application systems to understand and analyze images, videos, languages, and texts to satisfy the requirements of different scenarios, so that more and more enterprises can use AI and big data services conveniently, accelerating business development and contributing to society progress.

The screenshot shows the HUAWEI CLOUD website interface. At the top, there is a navigation bar with the HUAWEI logo, search bar, language selection (Intl-English), contact links (Contact Sales, After-Sales, Console), and user options (Log In, Register). Below the navigation bar, there is a secondary navigation menu with links: About Us, Products (selected), Solutions, Pricing, Marketplace, Learn, Partners, Careers, and More. A modal window is open, centered on the 'Storage' service. The modal has a header with a search bar ('Enter a product name') and a link to 'All products'. On the left side of the modal, there is a sidebar with a tree view of services: Compute, Storage (selected), Network, Database, Security, Application, Intelligent Cloud Acceleration, Enterprise Intelligence, and Management & Deployment. The main content area of the modal lists several storage services: Object Storage Service (stable, secure, efficient, and easy-to-use cloud storage), Elastic Volume Service (persistent block storage for ECSs and BMSS), Dedicated Distributed Storage Service (provides you with dedicated, physical storage resources), Cloud Backup and Recovery (backups for cloud servers, disks, and on-premises VMware virtual environments), Volume Backup Service (live, online backups for EVS disks), Cloud Server Backup Service (consistent online EVS disk backups for ECSs), and Storage Disaster Recovery Service. To the right of the main content, there is a 'Recommended Links' section with links to 'Fighting COVID-19 HOT' (Let Us Help You with Technology) and 'Product Introduction' (Elastic Volume Service Overview). A vertical sidebar on the far right also lists 'Fighting COVID-19' and 'Product Introduction'.

# HUAWEI CLOUD EI

**Industry know-how**

Deep understanding of industry pain points, driving AI implementation

**Industry data**

Processing and mining of abundant industry data to create huge values

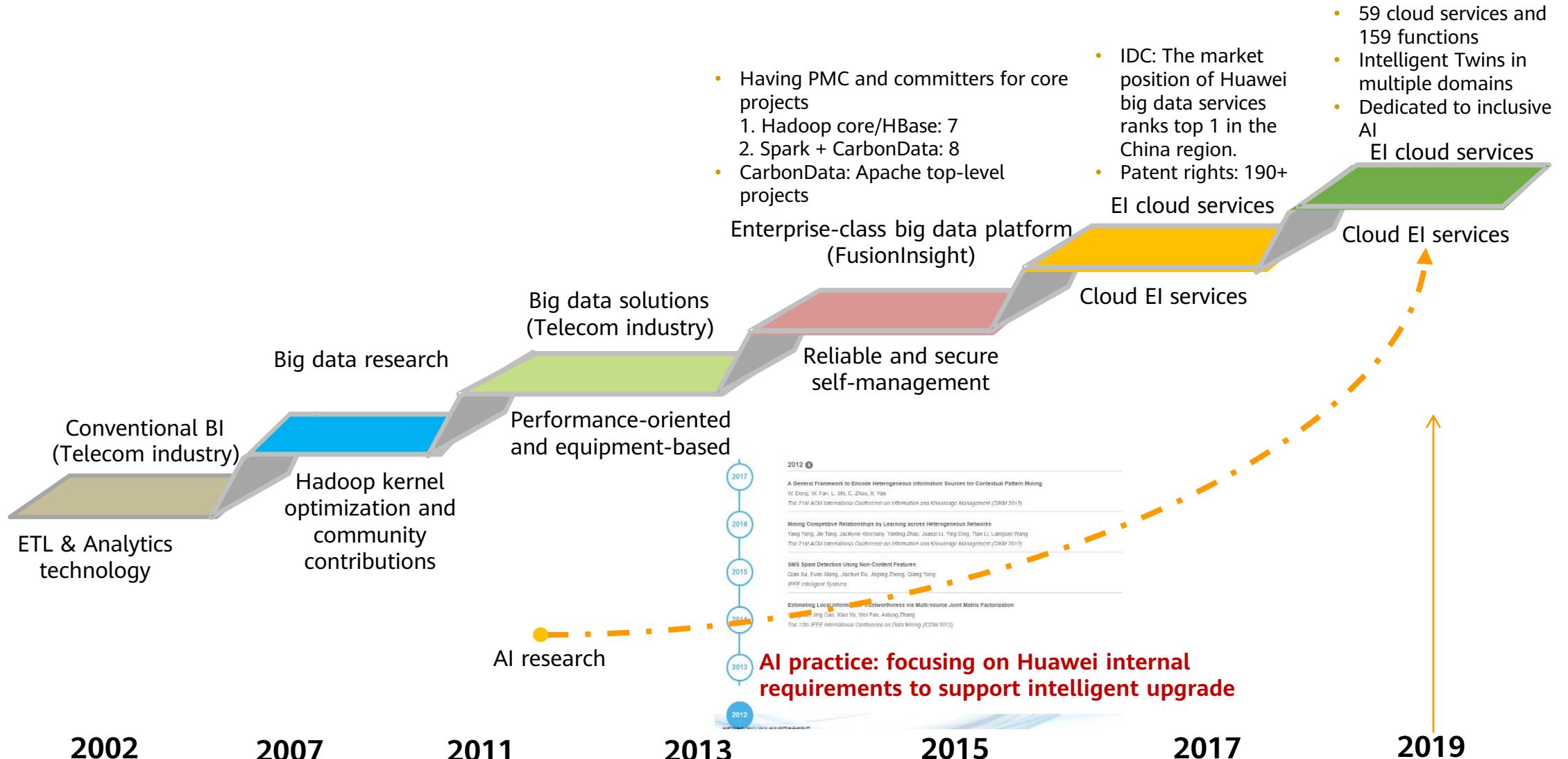
**Algorithms**

Various algorithm libraries and model libraries, general AI services and one-stop development platform

**Computing power**

30 years of information and communications technology (ICT) accumulation for most powerful and cost-effective AI computing power

# HUAWEI CLOUD EI Development History



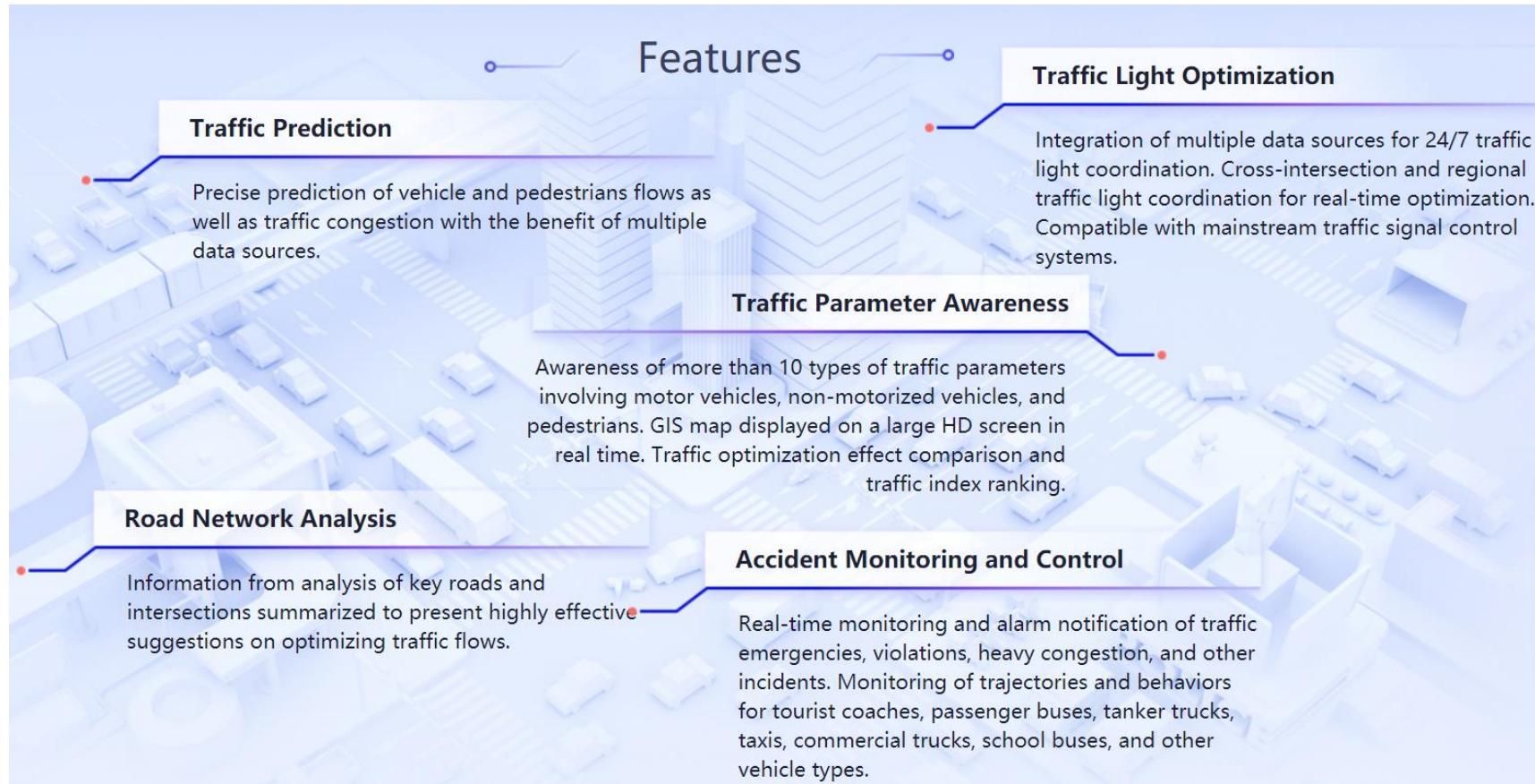
# EI Intelligent Twins

- The EI Intelligent Twins integrates AI technologies into application scenarios of various industries and fully utilizes advantages of AI technologies to improve efficiency and experience.



# Traffic Intelligent Twins (TrafficGo)

- The Traffic Intelligent Twins (TrafficGo) enables 24/7 and all-area traffic condition monitoring, traffic incident detection, real-time traffic signal scheduling, traffic situation large-screen display, and key vehicle management, delivering an efficient, environment-friendly, and safe travel experience.



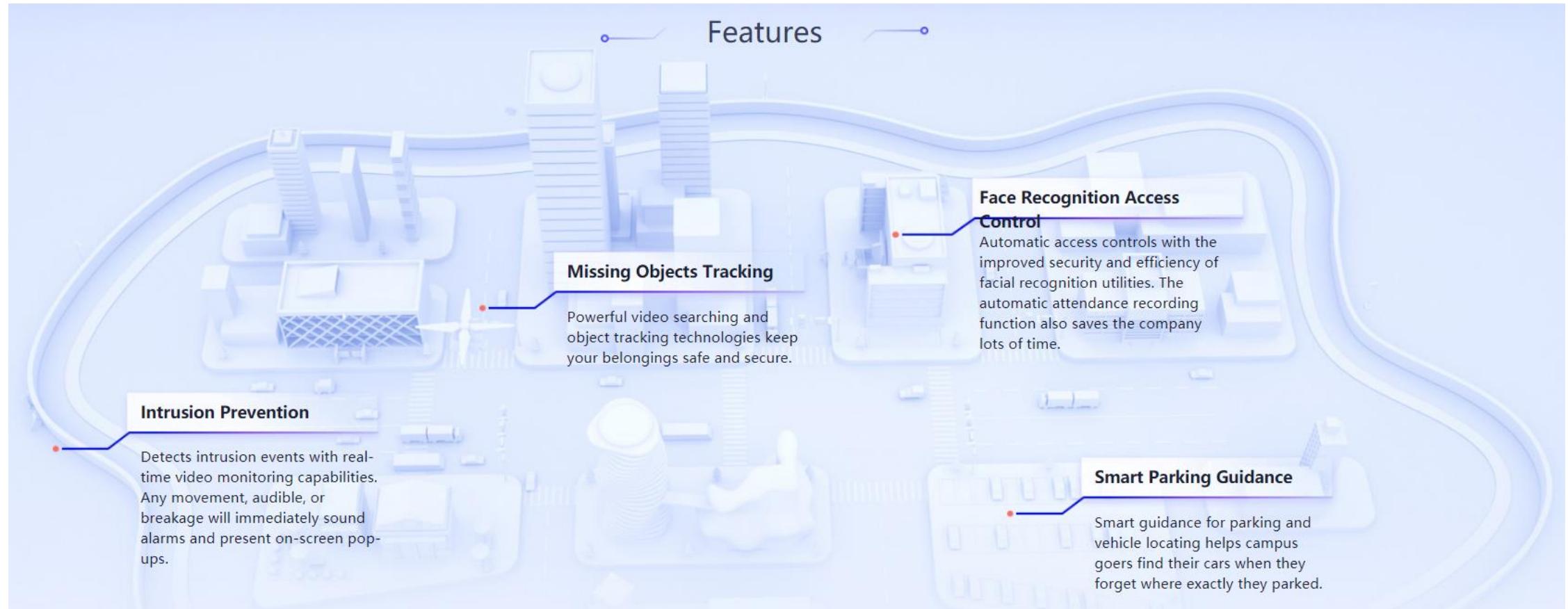
# Industrial Intelligent Twins

- The Industrial Intelligent Twins uses big data and AI technologies to provide a full series of services covering design, production, logistics, sales, and service. It helps enterprises gain a leading position.



# Campus Intelligent Twins

- The Campus Intelligent Twins manages and monitors industrial, residential, and commercial campuses. It adopts AI technologies such as video analytics and data mining to make our work and life more convenient and efficient.



# EI Products and Services

 <h3>ModelArts</h3> <p>ModelArts is a one-stop development platform that helps AI developers build models and manage the AI development lifecycle with data preprocessing, semi-automated data labeling, and distributed training.</p>	 <h3>Graph Engine Service</h3> <p>Graph Engine Service (GES) facilitates querying and analysis of graph-structure data based on various relationships. It is specifically suited for scenarios requiring analysis of rich relationship data.</p>	 <h3>Data Lake Insight</h3> <p>Data Lake Insight (DLI) is a Serverless big data compute and analysis service that is fully compatible with Apache Spark and Apache Flink ecosystems and supports batch streaming.</p>
 <h3>Video Ingestion Service</h3> <p>Video Ingestion Service (VIS) ingests massive volumes of video data in real time. Its superb data collection, real-time transmission, and video retention capabilities allow easy intelligent video analysis.</p>	 <h3>Data Warehouse Service</h3> <p>Data Warehouse Service (DWS) is a fast, easy-to-use, and reliable enterprise-class converged data warehouse service that can extend queries and analysis to your data lake with the help of DWS Express.</p>	 <h3>Cloud Stream Service</h3> <p>Cloud Stream Service (CS) is designed to process streaming data in real time. Computing clusters are fully managed by CS, allowing you to run StreamSQL or custom jobs without learning any programming skills.</p>
 <h3>MapReduce Service</h3> <p>MapReduce Service (MRS) provides enterprise-level big data clusters on the cloud. Tenants can fully control clusters and easily run big data components such as Hadoop, Spark, HBase, Kafka, and Storm.</p>	 <h3>Question Answering Bot</h3> <p>Question Answering Bot (QABot) helps enterprises quickly build, publish, and manage intelligent Q&amp;A bots.</p>	 <h3>Image Recognition</h3> <p>Image Recognition uses deep learning technologies to accurately identify objects, scenes, and concepts in images using a pool of visual content tags.</p>

# AI Essential Platform

## ModelArts

One-stop AI development platform

## Huawei HiLens

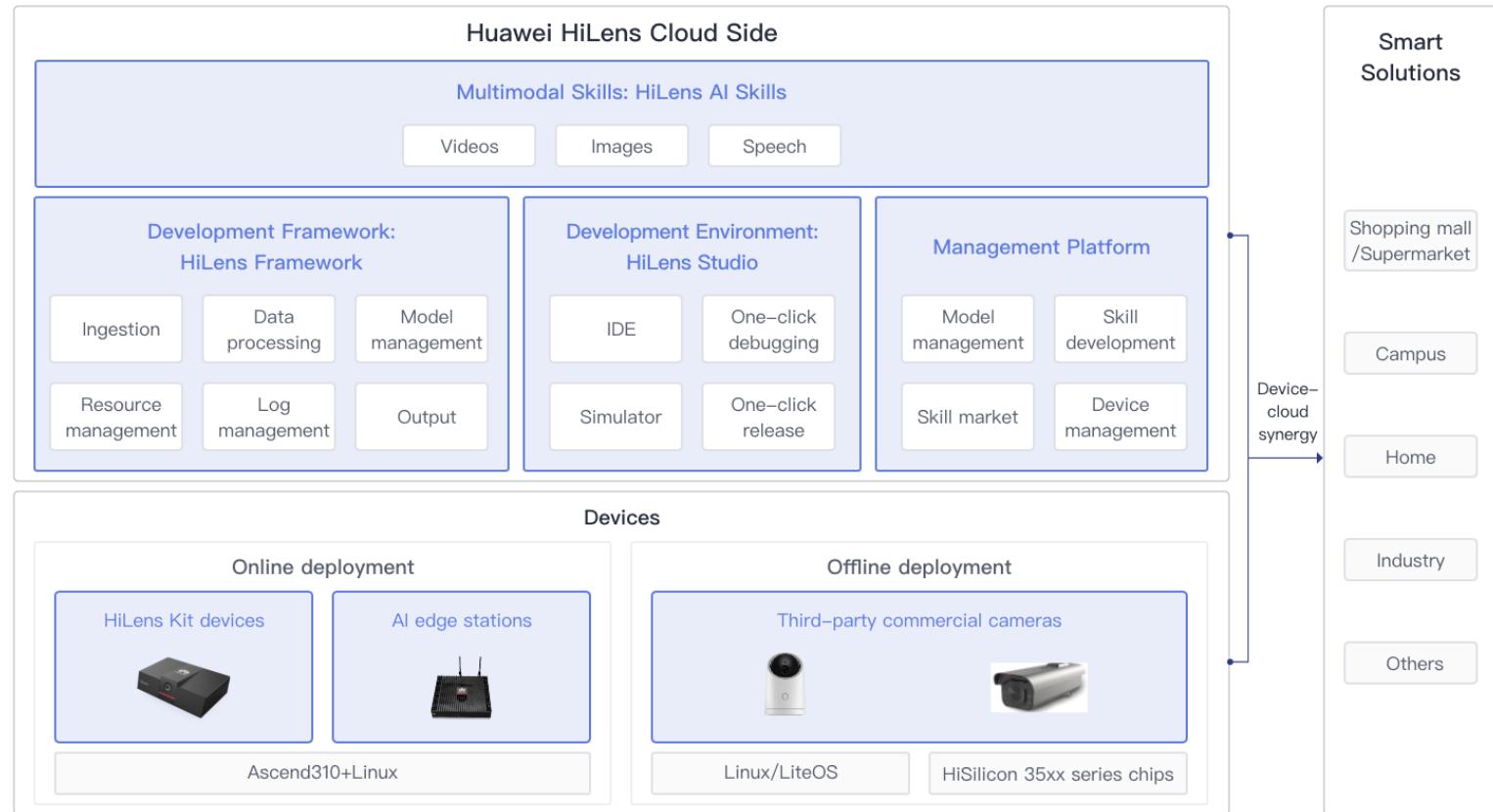
Multimodal AI development platform  
that enables device-cloud synergy

## Graph Engine Service (GES)

First commercial self-built distributed  
native graph engine with independent  
intellectual property rights in China

# Huawei HiLens

- Huawei HiLens consists of computing devices and a cloud-based development platform, and provides a development framework, a development environment, and a management platform to help users develop multimodal AI applications and deliver them to devices, to implement intelligent solutions in multiple scenarios.



# GES

GES facilitates **query and analysis** of graph-structure data based on various **relationships**. It uses the **high performance graph engine EYWA** as its kernel, and is granted many independent intellectual property rights. GES plays an important role in scenarios such as social apps, enterprise relationship analysis applications, logistics distribution, shuttle bus route planning, enterprise knowledge graph, and risk control.

- Social relationships
- Transaction records
- Call records
- Information propagation
- Browsing records
- Traffic networks
- Communications networks
- ...

**The massive and complex associated data is graph data in nature.**

- Diversified data independent of structures
- Data association and propagation capability
- Dynamic data changes and real-time interactive analysis without training
- Visualized and interpretable results

**Individual analysis**



**User profile**



**Opinion leader mining**

**Group analysis**



**Friend/Group recommendations**

**Link analysis**



**Hot topic identification**



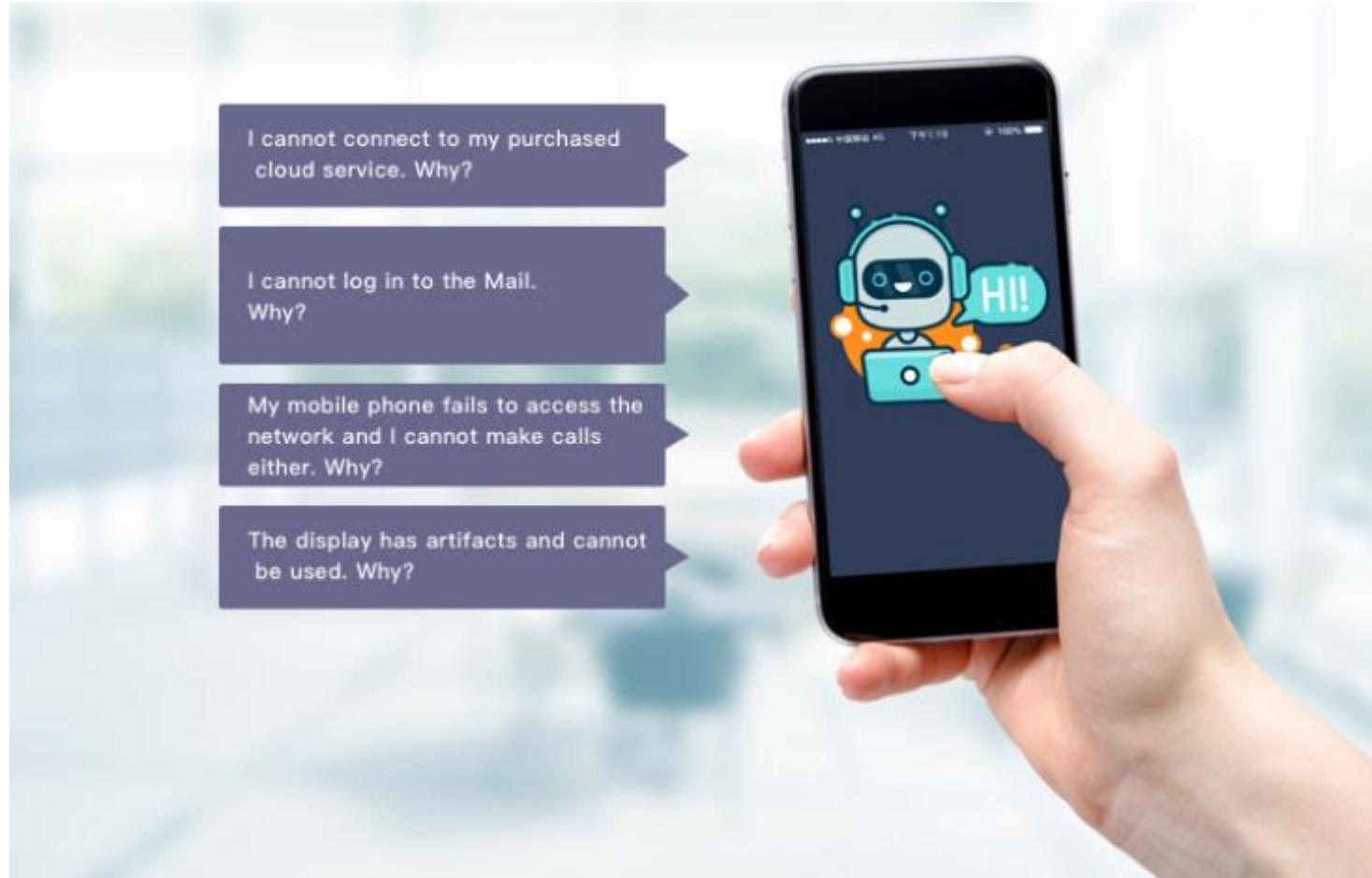
**Intelligent user grouping**



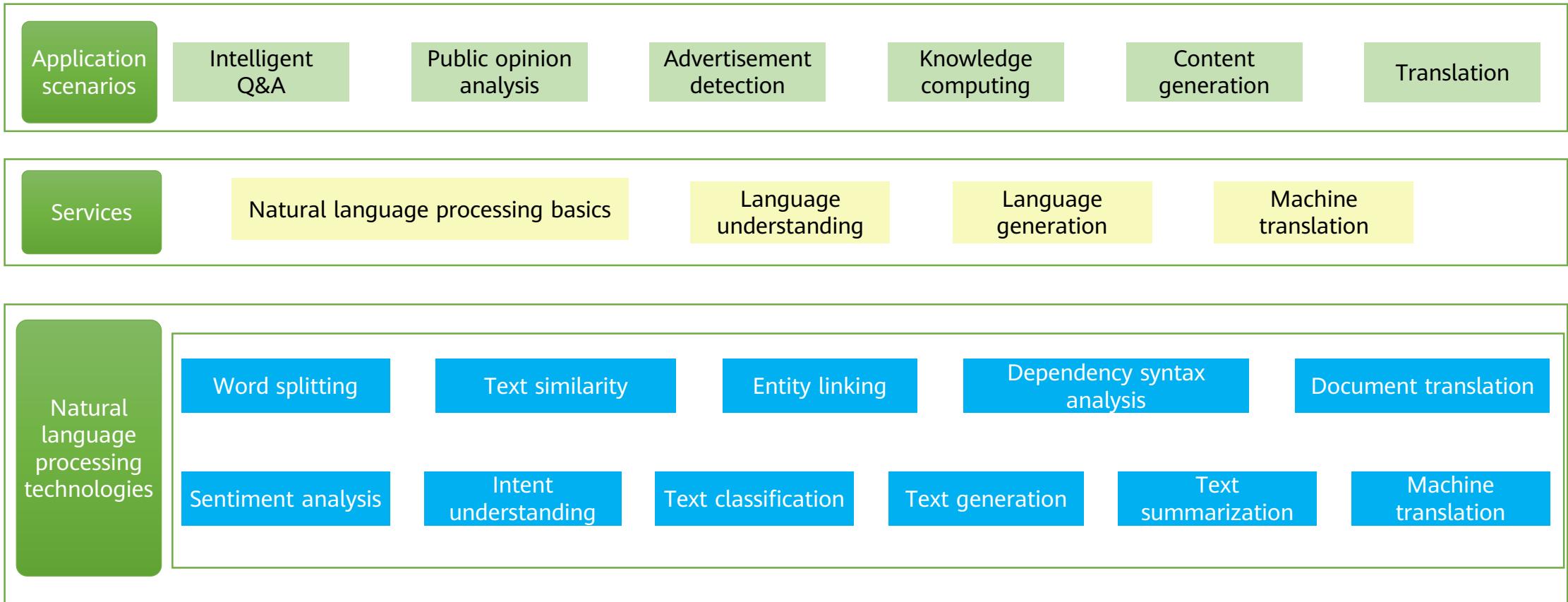
**Highlight search**

# Conversational Bot Service (CBS)

- Question-Answering bot (QABot)
- Task-oriented conversational bot (TaskBot)
- Speech analytics (CBS-SA)
- CBS customization



# Natural Language Processing



# Voice Interaction



Short sentence/speech recognition



Audio recording recognition

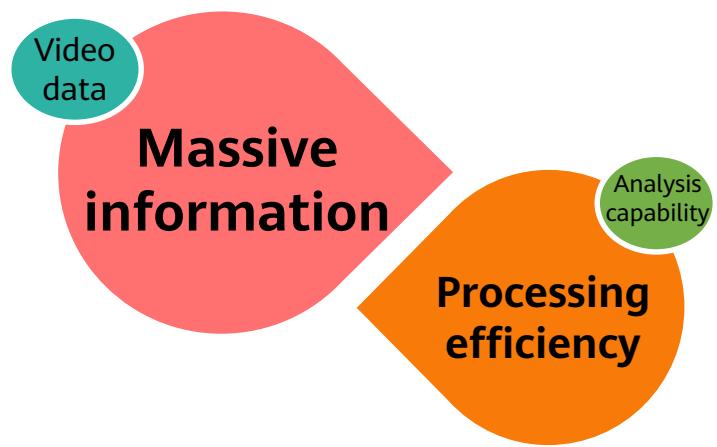


Real-time speech recognition

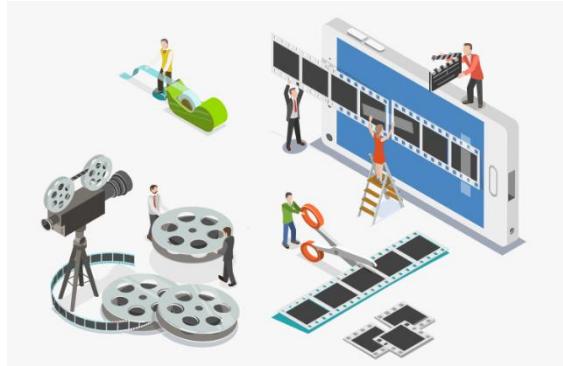


Audiobooks

# Video Analytics



**Video content analysis**



**Video editing**

Provide the cover, splitting, and summarization capabilities based on the overall video analytics.

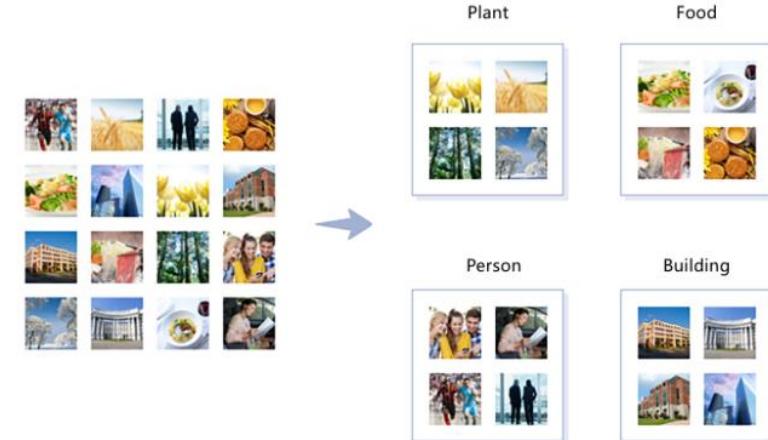
# Image Recognition



Scenario analysis



Object detection



Smart album

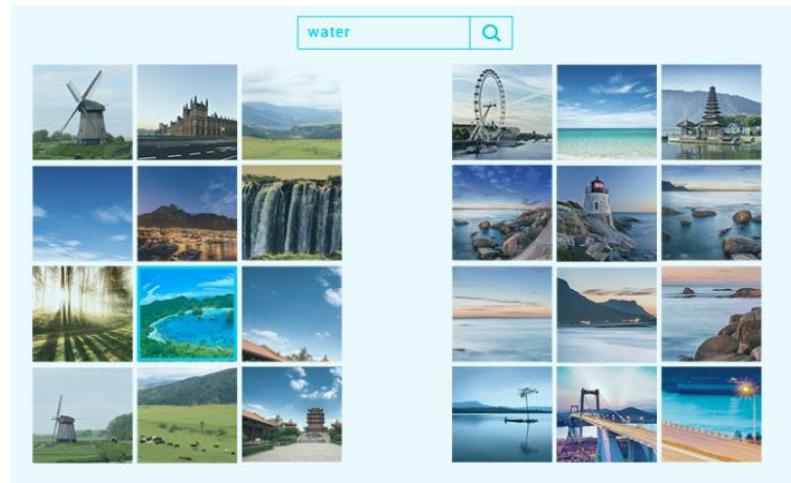


Image retrieval

# Content Moderation

Content moderation adopts cutting-edge image, text, and video detection technologies that precisely detect advertisements, pornographic or terrorism-related material, and sensitive political information, reducing non-compliance risks in your business.

## Moderation (image & text)

Sexy



Pornographic



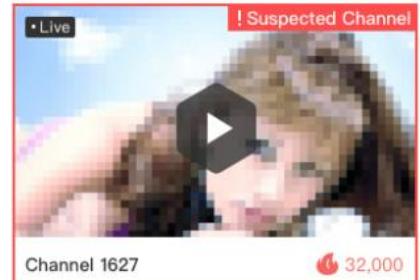
Obscene content identification



Terrorism identification

Political information detection

## Moderation (video content)



Determine whether a video has non-compliance risks and provide non-compliance information from multiple dimensions such as image, sound, and subtitle.

# AI Experience Center

- The AI Experience Center is an AI experience window built by Huawei, dedicated to lowering the threshold for using AI and making AI ubiquitous.

Image Tagging      Dark Enhance      Defog      Video Content Tagging

**Result**

Peacock	97.7%	Feather	95.3%
Birds	93.4%	Color	89.1%
Beautiful	88.8%	Bird	87.1%
Feathers	86.9%	Gorgeous	86.5%
Colorful ph...	84.7%	Wing	81.8%
Background	76.4%	Animal	75.3%
Close-up	73.1%	Gorgeous	71.5%
Colorful	71.4%		



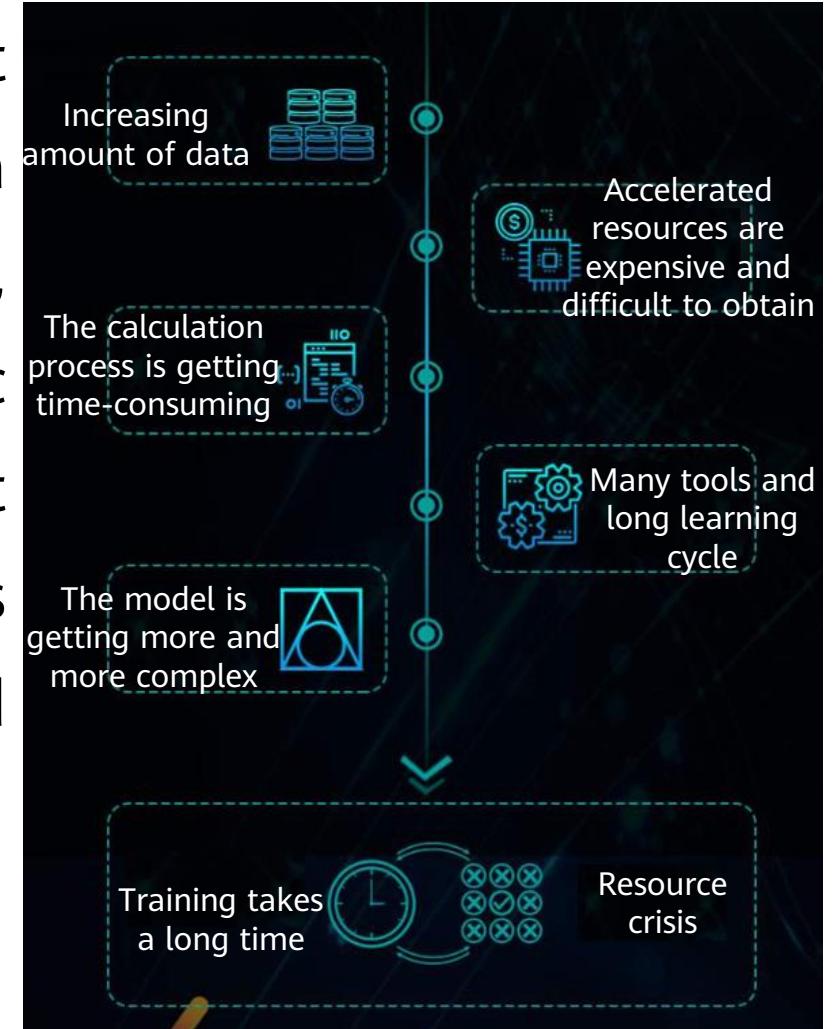
 Upload Image

# Contents

1. Overview of HUAWEI CLOUD EI
- 2. ModelArts**
3. HUAWEI CLOUD EI Solutions

# ModelArts

- ModelArts is a one-stop development platform for AI developers. With data preprocessing, semi-automatic data labeling, large-scale distributed training, automatic modeling, and on-demand model deployment on devices, edges, and clouds, ModelArts helps AI developers build models quickly and manage the AI development lifecycle.



# ModelArts Functions



## Data Management

ModelArts manages data preparation, such as collection, filtering, and labeling, and dataset versions, especially for deep learning datasets.



## Rapid and Simplified Model Training

Huawei's MoXing deep learning framework enables high-performance distributed training. To accelerate model development, it uses automatic hyperparameter tuning and standalone and distributed training.



## Model Deployment

ModelArts deploys models in various production environments such as devices, the edge, and the cloud, and supports online and batch inference jobs.



## ExeML

ModelArts supports code-free modeling and auto learning with image classification, object detection, and predictive analytics.



## Visualized Workflow

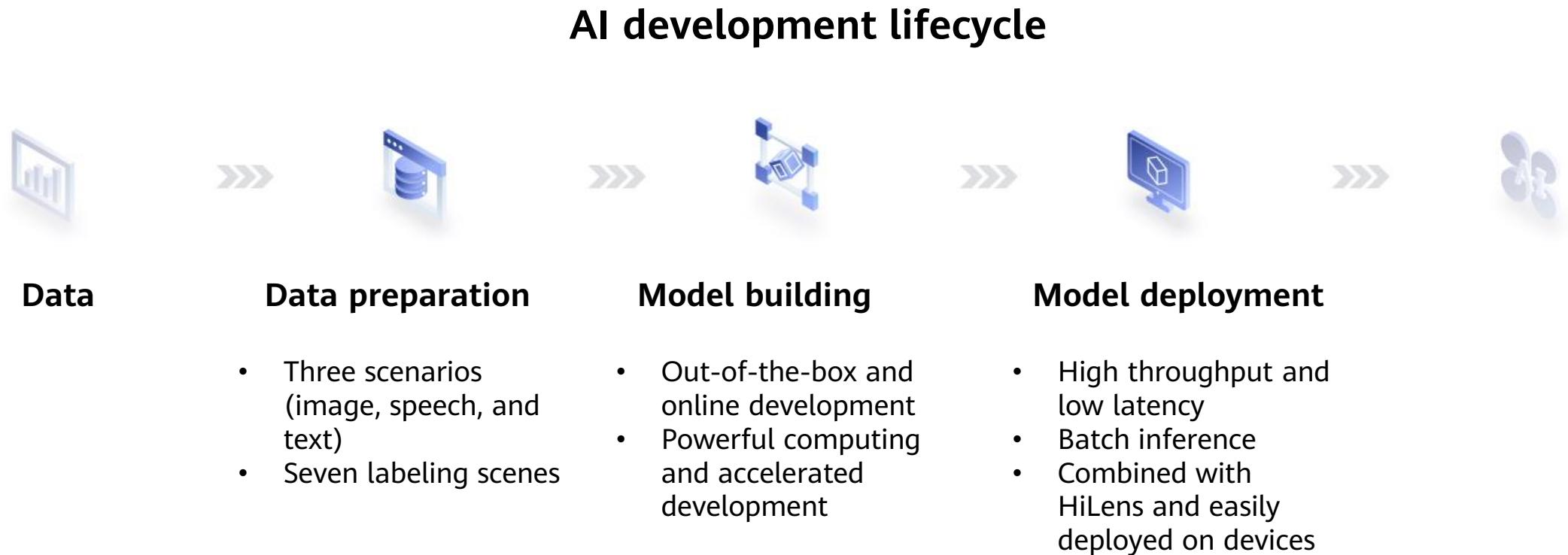
ModelArts works with Graph Engine Service (GES) to manage and visualize the lifecycle of AI development workflows, implementing data and model lineage.



## AI Marketplace

ModelArts supports common models and datasets, and internal or public sharing of enterprise models in the marketplace.

# ModelArts Applications

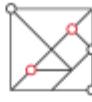


# ModelArts Highlights



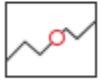
## One-Stop Platform

The out-of-the-box and full-lifecycle AI development platform provides one-stop data processing, model development, training, management, and deployment.



## Easy to Use

Various built-in open source models and automatic hyperparameter tuning help you start model training from scratch. Models can be deployed on the device, edge, and cloud with just one click.



## Excellent Performance

The Huawei-developed MoXing framework delivers high-performance algorithm development and training. GPU utilization is optimized for online inference. Huawei Ascend chips significantly accelerate inference.



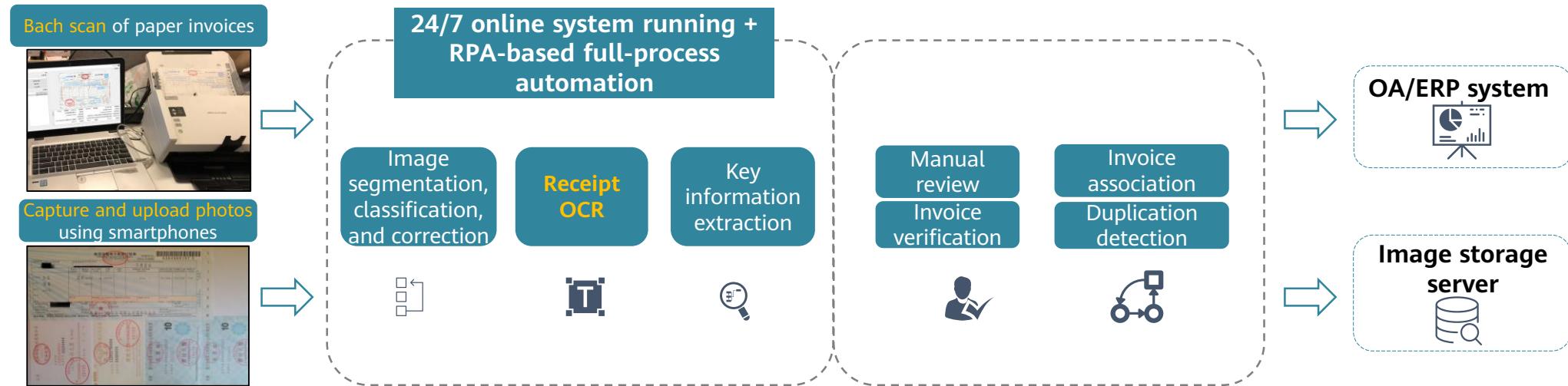
## High Flexibility

ModelArts supports multiple mainstream open source frameworks, such as TensorFlow and Apache Spark MLlib, mainstream GPUs, and the Huawei-developed Ascend AI chips. Exclusive use of resources and custom images ensure flexible development experience.

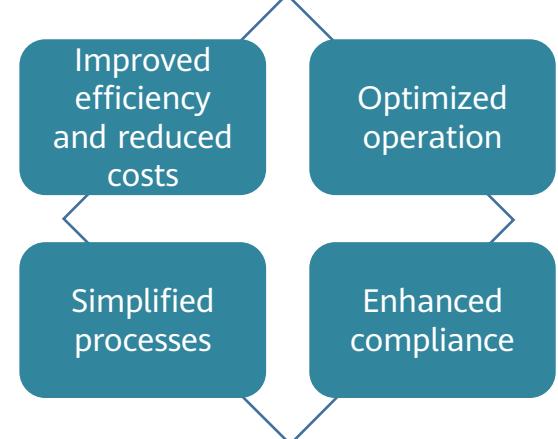
# Contents

1. Overview of HUAWEI CLOUD EI
2. ModelArts
- 3. HUAWEI CLOUD EI Solutions**

# Case: OCR Implements Full-Process Automation for Reimbursement Through Invoices.

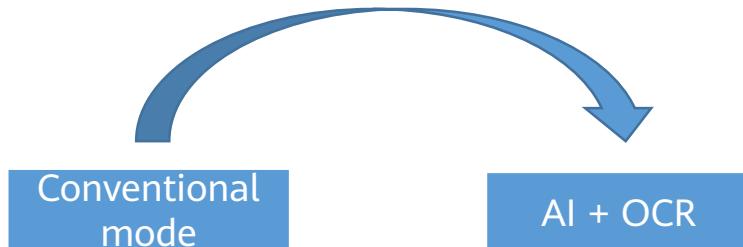
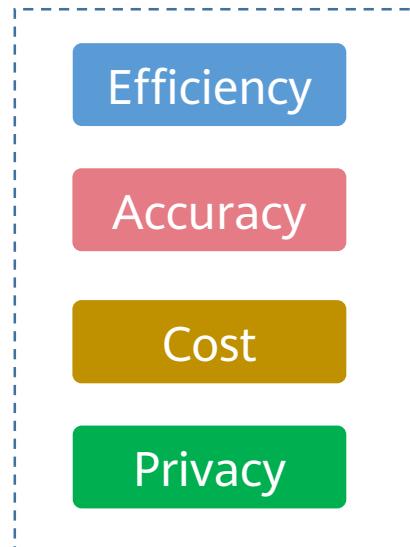
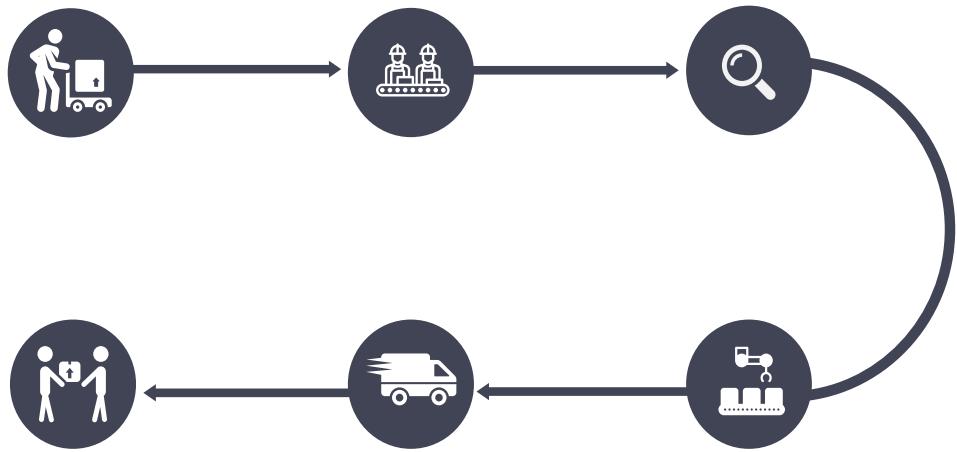


- **Multiple access modes:** automatic connection to scanners to obtain images in batches; image capture by using high-speed document scanners and mobile phones
- **Flexible deployment:** multiple deployment modes such as public cloud, HCS, and appliance, and unified standard APIs
- **Support for various invoices:** regular/special/electronic/ETC/roll value-added tax (VAT) invoices, and taxi/train/flight itinerary/quota/toll invoices
- **One image for multiple invoices:** automatic classification and identification of multiple types of invoices
- **Visualized comparison:** return of OCR character location information and conversion of such information into an Excel file for statistics collection and analysis

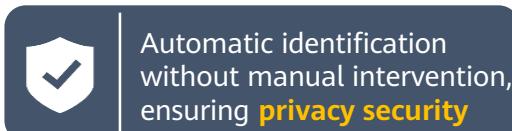
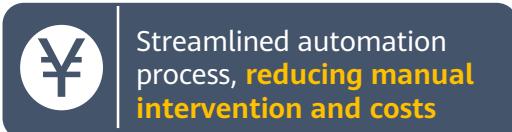
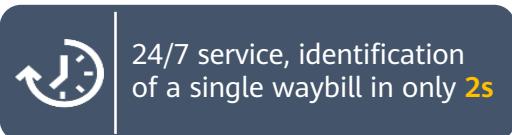


# Case: Intelligent Logistics with OCR

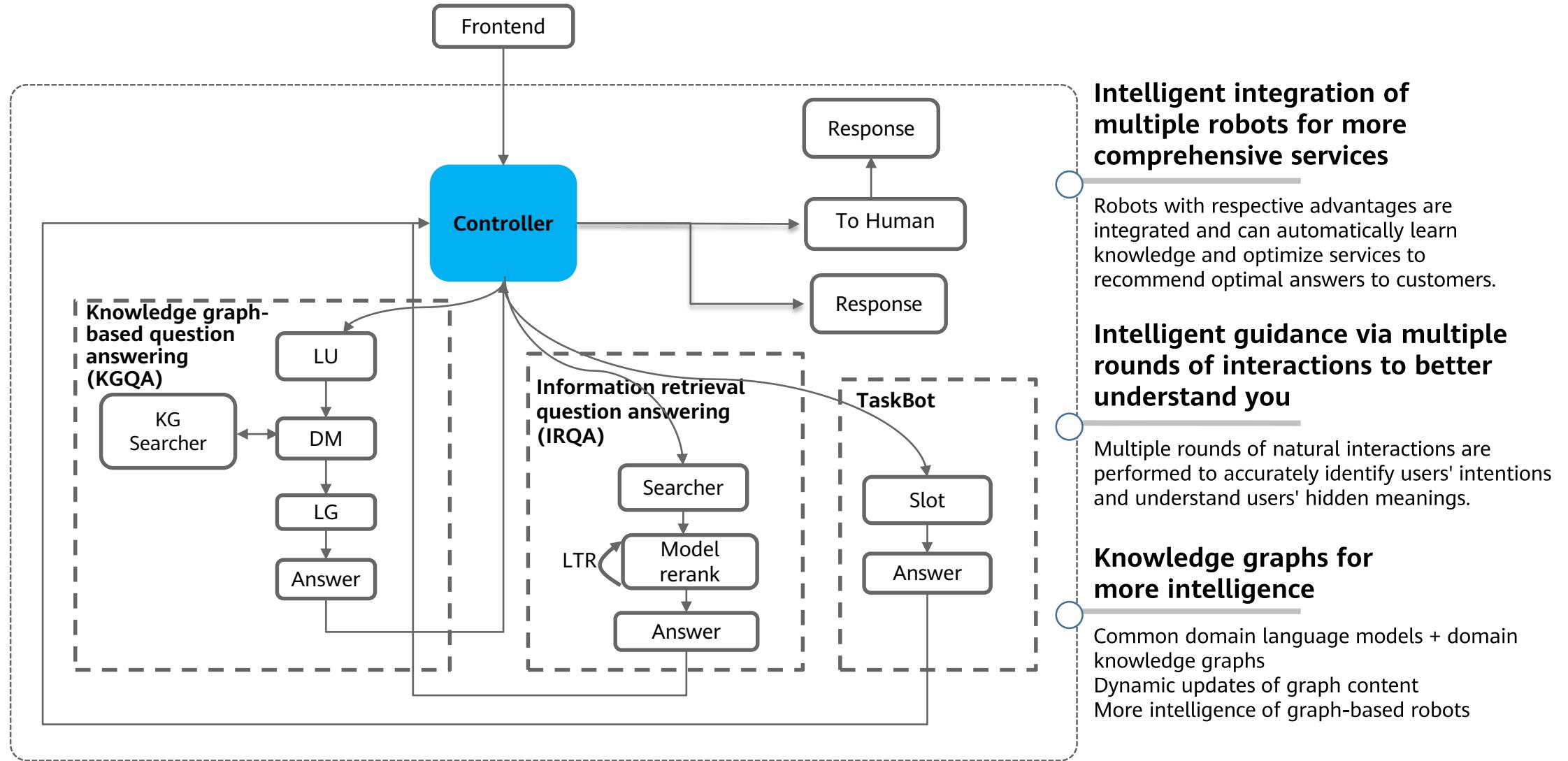
- **ID card OCR**
  - ID card photographing, recognition, and verification with mobile apps
- **Screenshot OCR**
  - After an e-commerce platform receives a buyer's address and chat screenshots, OCR recognizes and extracts the information automatically.



- **Electronic waybill OCR**
  - Automatic extraction: waybill number, and name, phone number, and address of the receiver/sender
- **Paper waybill OCR**
  - Text and seal detection
- **Receipt OCR**
  - Invoice information recognition



# CBS



# Case: Conversational Bot with Vehicle Knowledge

## Recommendations

- ✓ Please recommend a car priced at about CNY30,000.
- ✓ Please recommend a commercial vehicle.
- ✓ Please recommend vehicle models with a torque greater than 310 Nm.

## Open consulting

- ✓ How about Mercedes-Benz E?
- ✓ How is the security of S90?
- ✓ How is Audi A6L's interior?

## Precise answers based on vehicle knowledge graphs

How about the Mercedes-Benz A200 sports sedan?

The Mercedes-Benz A200 sports sedan... is equipped with... (details)

How is the security performance?

Active braking is provided in the standard configuration... (Introduce the security advantages.)  
The unique xxx solution is additionally provided...

## Comparison

- ✓ Comparison between Mercedes-Benz E300L Sedan and BMW 530Li xDrive
- ✓ Which of Audi A6L and the BMW 5 series is better?

## Performance

- ✓ What is the wheel-base of 530 Li?
- ✓ What is the fuel consumption of Mercedes-Benz E300L?
- ✓ Can seats of Lexus ES be heated?

## Multiple rounds of interactions and multimodal input

How is the interior?

...xxx top leather... (description of the interior)



## After-sales

- ✓ What can I do if the steering wheel does not move?
- ✓ Why does the indicator keep blinking? (Figure)
- ✓ How long will the insurance be paid?

## Pre-sales

- ✓ Is test driving available for S90?
- ✓ Which 4S store has stock?
- ✓ Can I try Audi A4L next Wednesday?

## Proactive guidance and preferential answers

What is the difference between S90 T5 and BMW 530Li?

The price of S90 T5 is CNY410,800, and that of the BMW 530Li is CNY519,900. BMW 530Li has 9 standard configurations as its selling points, while S90 T5 has 10. **The cost of BMW 530Li further increases if one additional configuration needs to be added for BMW 530Li to equal S90 T5.**

# Case: Intelligent Q&A of Enterprises in a Certain District



## Agent Assistant

Improves productivity and customer satisfaction with real-time analysis and helps on improving the interaction between agents in the call center and customers. During a call, the bot automatically extracts keywords, coils problems, searches for and displays the best answers to matching semantics, and provides real-time support for agents.

## Advantages

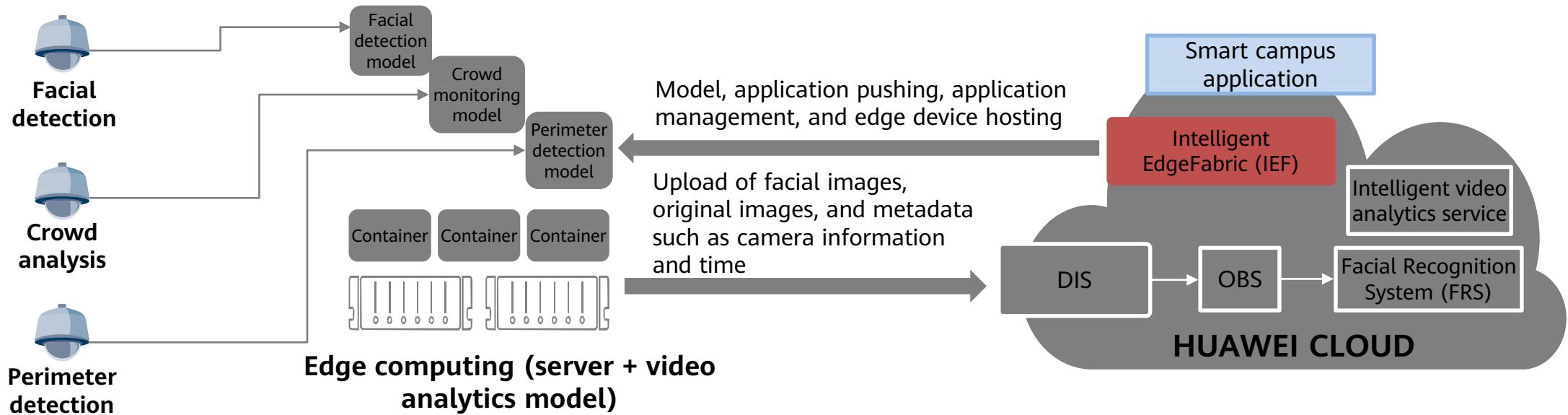
- **Real-Time Support**

Offers real-time support for human agents by attempting to understand customer questions and matching the answers with high relevancy.

- **Improved Efficiency & Satisfaction**

Enables human agents to answer customers' questions at a faster speed.

# Case: Smart Campus



## Device-side common HD IPCs:

- Face capturing
- Video analytics at the edge

## Campus surveillance



## Edge side:

- GPU servers are recommended.
- IEF pushes the facial detection, crowd monitoring, and perimeter detection algorithms for deployment on edge nodes.
- IEF manages the application lifecycle (with the algorithms iteratively optimized).
- IEF centrally manages containers and edge applications.

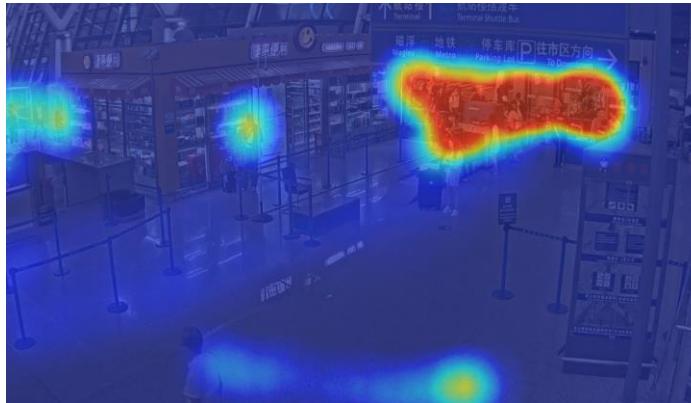
## Competitiveness and values of edge video analytics

- Service values:** Intelligently analyze surveillance videos to detect abnormal security events such as intrusions and large crowd gatherings in real time, reducing labor costs.
- Edge-cloud synergy:** Perform full-lifecycle management and seamless upgrade of edge applications.
- Cloud model training:** Implement automatic training using algorithms that have good scalability and easy to update.
- High compatibility:** Reuse existing IPCs in campuses as smart cameras through edge-cloud synergy.

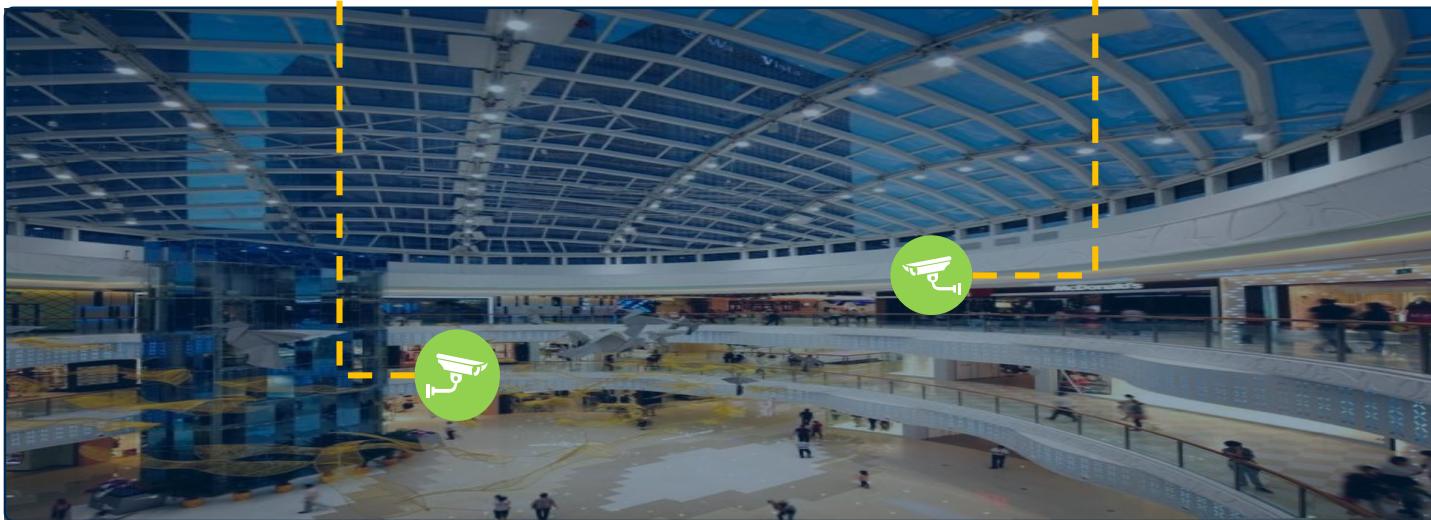
# Case: Crowd Statistics and Heat Map



Region crowd statistics



Region crowd heat map



- **Functions:**

- Counting the crow in an image.
- Collecting popularity statistics of an image.
- Supporting customized time settings.
- Enabling configurable intervals for sending statistics results.

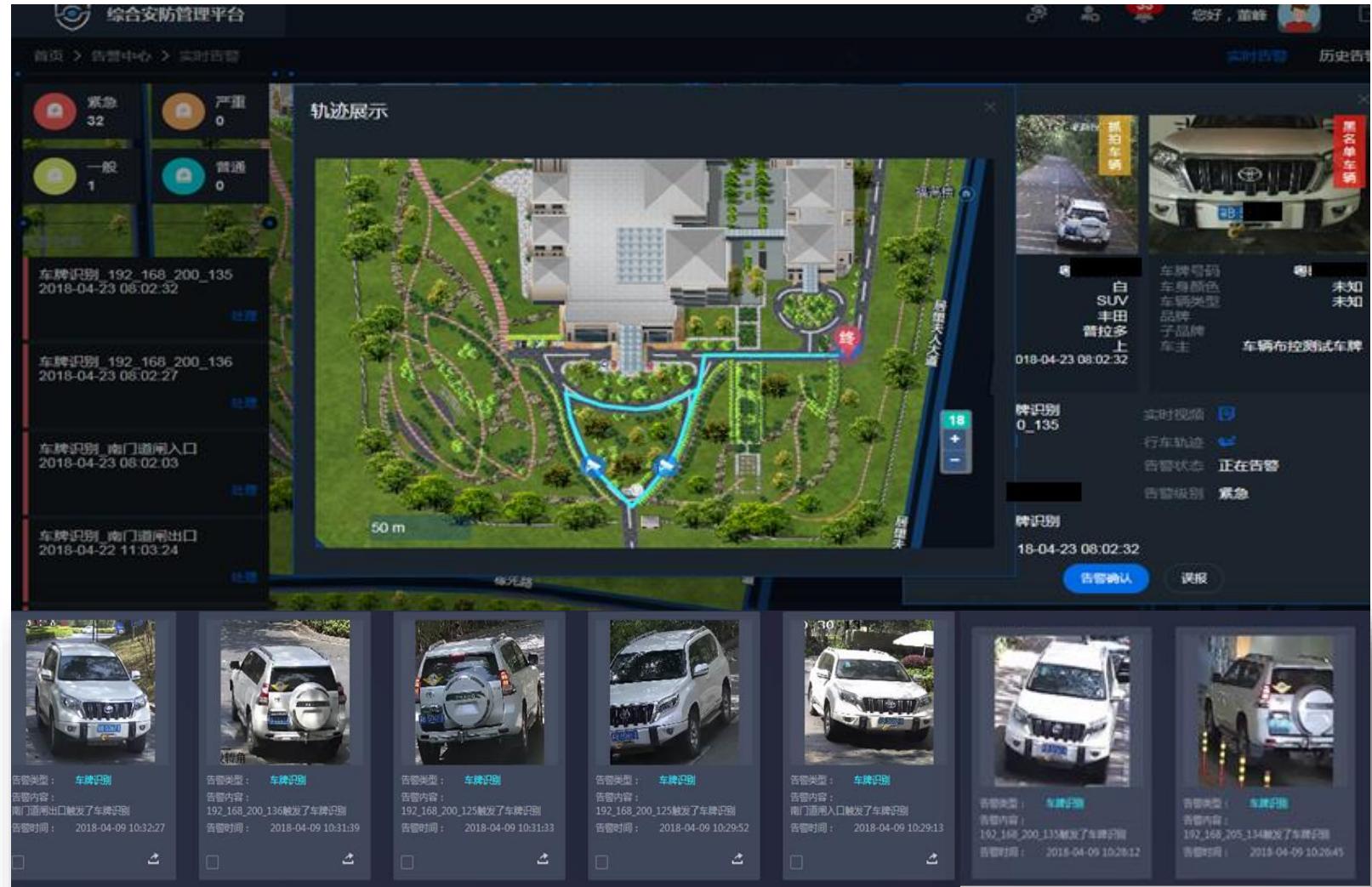
- **Scenarios:**

- Customer traffic statistics
- Visitor statistics
- Business district popularity identification

- **Advantages:**

- Strong anti-interference performance: crowd counting in complex scenarios, such as face blocking and partial body blocking
- High scalability: concurrent sending of pedestrian crossing statistics region statistics, and heat map statistics
- Ease-of-use: compatible with any 1080p surveillance camera

# Case: Vehicle Recognition



- **Functions:**

- Vehicle model detection
- Vehicle color recognition
- License plate recognition (LPR)

- **Scenarios:**

- Campus vehicle management
- Parking lot vehicle management
- Vehicle tracking

- **Advantages:**

- Comprehensive scenarios: recognition of vehicle models, styles, colors, and license plates in various scenarios such as ePolice and checkpoints
- Ease-of-use: Compatible with any 1080p surveillance camera

# Case: Intrusion Detection



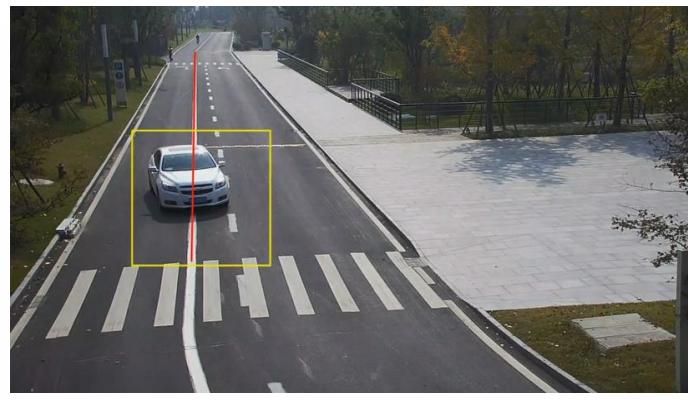
Personnel tripwire crossing detection



Area intrusion detection



Climbing detection



Vehicle tripwire crossing detection

## Functions:

- Extracting moving objects from a camera's field of view and generating an alarm when an object crosses a specified area.
- Setting the minimum number of people in an alarm area.
- Setting the alarm triggering time.
- Setting the algorithm detection period.

## Scenarios:

- Identification of unauthorized access to key areas
- Identification of unauthorized access to dangerous areas
- Climbing detection

## Advantages:

- ✓ **High flexibility:** settings of the size and type of an alarm object
- ✓ **Low misreporting rate:** people/vehicle-based intrusion alarm, without interference from other objects
- ✓ **Ease-of-use:** compatible with any 1080p surveillance camera

# Summary

- This chapter describes the following content:
  - HUAWEI CLOUD EI ecosystem, helping you understand the HUAWEI CLOUD EI services
  - ModelArts services in combination with experiments, helping you understand the ModelArts services more efficiently
  - EI cases

# Quiz

1. Which of the following scenarios can EI be applied to? ( )
  - A. Intelligent government
  - B. Intelligent city
  - C. Intelligent manufacturing
  - D. Intelligent finance

# More Information

---

Huawei Talent Online Website

<https://e.huawei.com/en/talent/#/home>

WeChat public accounts:



EMUI



Huawei Device  
Open Laboratory



HUAWEI  
Developers



Smart-E

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

Copyright©2020 Huawei Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

