

Artificial Intelligence Overview



 HUAWEI

Objectives

- Upon completion of this course, you will understand:
 - Basic concepts of AI
 - AI technologies and development history
 - Application technologies and fields of AI
 - Huawei's AI development strategy
 - AI development trends

Contents

- 1. AI Overview**
2. Application Fields of AI
3. Huawei's AI Development Strategy
4. Controversies Over AI and Its Future

Insights from Prominent Computer Scientists

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

— Alan Turing, 1950

"...making a machine behave in ways that would be called intelligent if a human were so behaving."

— John McCarthy et al., 1956

"Artificial intelligence is the science of making machines do things that would require intelligence if done by men."

— Marvin Minsky

- "...the artificial intelligence problem is taken to be that of making a machine behave in ways that would be called intelligent if a human were so behaving." - A popular definition of artificial intelligence (AI), and an earlier definition in this field proposed by John McCarthy at the Dartmouth Conference in 1956. However, it seems that this definition ignores the possibility of strong AI. In another definition, AI is the intelligence demonstrated by artificial machines (weak AI).
- Alan Turing discussed the question of "Can machines think?" in his seminal paper *Computing Machinery and Intelligence*.

What is intelligence?

- According to the theory of multiple intelligences proposed by Professor Howard Gardner, multiple intelligences are manifested by eight capabilities:
 - Linguistic-verbal intelligence
 - Logical-mathematical intelligence
 - Visual-spatial intelligence
 - Bodily-kinesthetic intelligence
 - Musical-rhythmic and harmonic intelligence
 - Interpersonal intelligence
 - Intrapersonal intelligence
 - Naturalistic intelligence

- Howard Gardner is an American educator and psychologist.
- 1. Language intelligence

It refers to the ability to express thoughts and understand others by using oral languages or in words, and to master speech, semantics, and grammar flexibly, with the ability to think in words, express in words, and appreciate the deep meaning of languages. Ideal professions for people mastering language intelligence include politicians, hosts, lawyers, speakers, editors, writers, journalists, and teachers.

- 2. Logical-mathematical intelligence

It refers to the ability to calculate, measure, infer, conclude, and classify, and to carry out complex mathematical operations. This intelligence includes sensitivity to logical ways and relationships, statements and propositions, functions, and other related abstract concepts. Ideal professions for people mastering logical mathematical intelligence include scientists, accountants, statisticians, engineers, and computer software developers.

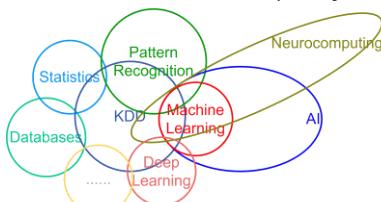
- 3. Spatial intelligence

It refers to the ability to accurately perceive the visual space and surroundings and to present the perception in the form of graphics. This intelligence includes sensitivity to colors, lines, shapes, forms, and spatial relationships. Ideal professions for people mastering spatial intelligence include interior designers, architects,

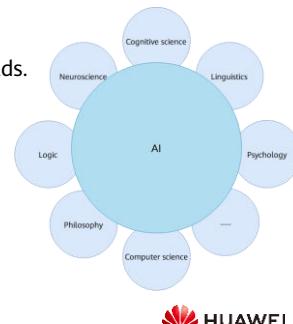
photographers, painters, and pilots.

What Is Artificial Intelligence?

- "Artificial" in artificial intelligence (AI) means that it is designed by and is created for humans.
- AI is a scientific discipline that studies and develops theories, techniques, and application systems used to simulate and extend human intelligence. The term was first coined by John McCarthy in 1956, who defined it as the "science and engineering of making intelligent machines, especially intelligent computer programs". The very premise of AI technology is to enable machines to learn from collected data, and make human-like decisions.
- Today, AI has become an interdisciplinary course that involves various fields.



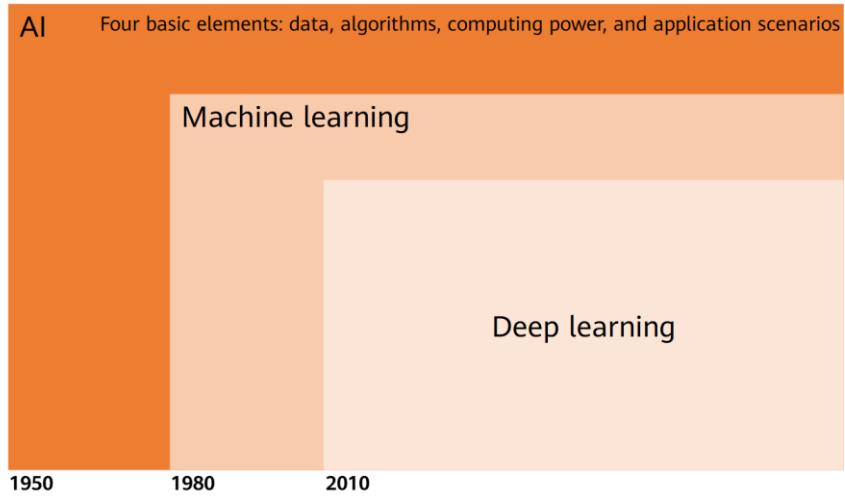
7 Huawei Confidential



 HUAWEI

- Machine learning can be understood from multiple aspects. Tom Mitchell, a global machine learning scientist, provided a widely quoted definition: "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." These definitions are simple and abstract. However, as we deepen our understanding of machine learning, we will find that the connotation and extension of machine learning are changing over time. Because a variety of fields and applications are involved and machine learning develops rapidly, it is not easy to define machine learning simply and clearly.
- In general knowledge, processing systems and algorithms of machine learning (ML) are an identification mode that performs prediction by finding a hidden mode in data. ML is an important subfield of AI, which also intersects with data mining (DM) and knowledge discovery in database (KDD), for better understanding and distinguishing of artificial intelligence, machine learning, data mining, pattern recognition, statistics, and neural computing.

Relationship Between AI, Machine Learning, and Deep Learning



Relationship Between AI, Machine Learning, and Deep Learning

- AI is a scientific discipline that studies and develops theories, techniques, and application systems used to simulate and extend human intelligence.
- Machine learning (ML) refers to the ability of computers to learn, simulate, or implement human behavior to acquire new knowledge or skills, and continuously update existing knowledge structures to improve performance.
- Deep learning (DL) is a research field in ML and originates from artificial neural network (NN) studies. Multilayer perceptron (MLP) is a deep learning structure. Deep learning uses higher level features derived from the lower level features to form a hierarchical representation, in which it simulates the mechanisms of the human brain to interpret data, such as images, voice, and text.

- The term "artificial intelligence" was previously used to describe machines that imitate and demonstrate "human" cognitive skills related to human thinking.
- Machine learning algorithms build models based on sample data (training data) to make predictions or decisions without explicit rules.
- Deep learning (DL) is a research field in ML and originates from artificial neural network (NN) studies. Multilayer perceptron (MLP) is a deep learning structure. Deep learning uses higher level features derived from the lower level features to form a hierarchical representation. Deep learning aims to establish neural networks that simulate the human brain to analyze and interpret data, such as images, voice, and text.

Major Schools of AI - Symbolism

- Symbolicism, also called logicism, psychologism, or computerism, refers to the symbolic AI that is derived from mathematical logic.
- It suggests the basic cognition units of humans are symbols, and human cognition is a reasoning process based on various symbols. As humans and computers are both physical symbol systems, computers can be used to simulate intelligent human behaviors.
- This theory, proposed by McCarthy et al in 1956, first used the term "artificial intelligence" and made dominant contributions to the field's development, especially the success of expert systems.

- Since 1956, symbolism developed heuristic algorithms -> expert systems -> knowledge engineering theory and technology and made great progress in the 1980s.
- The successful development and application of expert systems is of great significance for AI to go from theory to engineering applications.
- Symbolicism remained to be a mainstream even after other AI theories emerged.

Major Schools of AI - Connectionism

- Connectionism, also known as bionicsm or physiologism, argues AI originates from bionics, that is, it is the study of human brain.
- Researchers believe the neuron, and not symbol processes, is the basic thinking unit. Connectionism starts with neurons and studies NNs and brain models, to create a new development path for AI.
- The McCulloch-Pitts (MP) neuron model was proposed in 1943. However, the study of brain models was limited due to biological prototypes and technical conditions in the 1970s and 80s. It was not until the proposal of hardware-simulated neural networks that the trend of connectionism reemerged again.
- Nowadays, the artificial neural network (ANN) is a common technique but its complexity and scale has brought many interpretability problems.

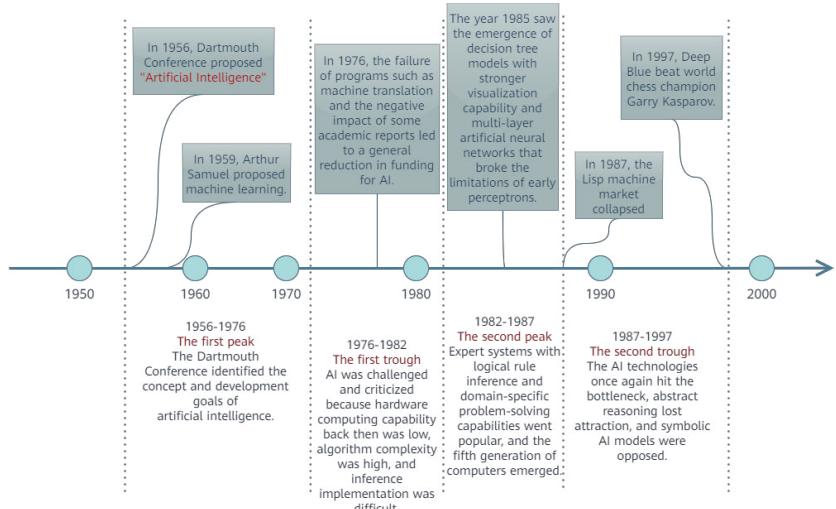
- Neurons and synapses in human brains are represented by units and connections in the network.
- In 1986, David Rumelhart et al. proposed the backpropagation (BP) algorithm in multi-layer neural networks. Since then, research in connectionism became active again. Connectionism went from model to algorithm and from theoretical analysis to engineering implementation.

Major Schools of AI - Actionism

- Actionism is also known as evolutionism or cyberneticsism and states that AI originates from cybernetics.
- It suggests intelligence depends on perception and actions, and that intelligence does not require knowledge, representation, or reasoning. AI can evolve like human intelligence, and intelligent behavior can only be manifested in the real world by interacting with the surrounding environment.
- Early research on actionism focused on simulating intelligent behavior of people in the control process. It led to the birth of intelligent control and robotics in the 1980s.

- Actionism puts more focus on practice. It believes in constantly adapting actions by learning from the environment.
- The representative achievement of actionism is Rodney Brooks' six-legged robot, a control system that simulates insect behavior based on the "perception-action" mode.
- Actionism considers AI as an adaptive control system that collects environment data using sensors and performs actions.

AI Development History (1)

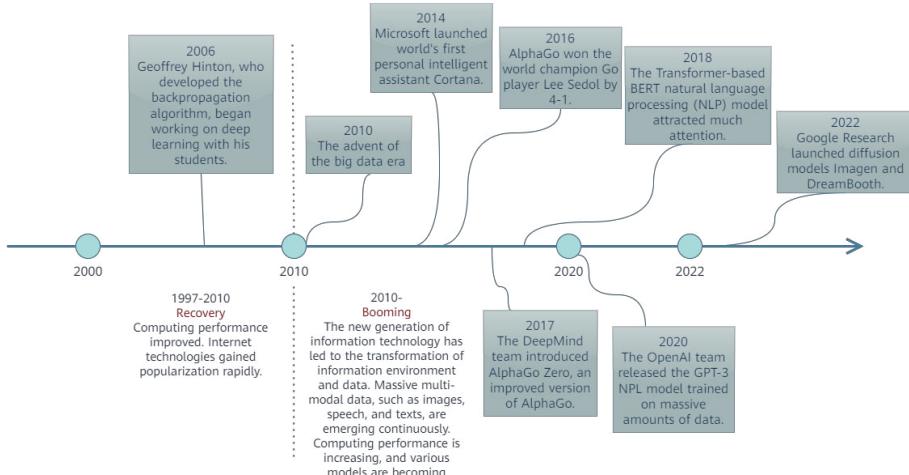


13 Huawei Confidential



- In August 1956, a group of scientists gathered at Dartmouth College in the US to discuss an otherworldly topic: how to make computers simulate learning behavior and other intelligent actions of human beings. Many well-known scientists attended the Dartmouth Conference, including John McCarthy (developer of LISP), Marvin Minsky (expert in AI and cognition), and Claude Shannon (founder of the information theory), Allen Newell (computer scientist), and Herbert Simon (Nobel laureate in economics).
- The workshop ran for two months. No consensus was reached, but they picked the name artificial intelligence for the field they discussed about. Then, **the year 1956 marked the birth of AI**.
- Lisp: [https://en.wikipedia.org/wiki/Lisp_\(programming_language\)](https://en.wikipedia.org/wiki/Lisp_(programming_language))
- Lisp Machine: https://en.wikipedia.org/wiki/Lisp_machine

AI Development History (2)



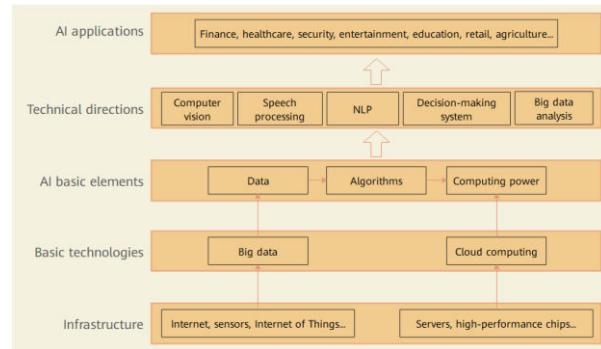
Types of AI

- Strong AI:
 - This hypothesis aims to create intelligent machines that replicate human functions, such as reasoning and problem-solving, and are perceptive and self-conscious. Strong AI will be able to think independently and teach itself to solve new problems, and have its own values and worldviews, and will even have the same instincts as creatures, such as survival and safety needs. In a sense, strong AI can be seen as a new species.
- Weak AI:
 - Weak AI aims to build intelligent machines that can perform specific tasks but rely heavily on human interference. These machines may seem intelligent but are not self-conscious.

- A key counter of AI is to achieve a superhuman level in challenging fields through self-learning without any prior knowledge.
- Strong AI can compete with humans in all aspects. Therefore, it aims to enable robots to implement human-like capabilities in all aspects rather than a specific field. Strong AI can think, make plans, solve problems, perform abstract thinking, understand complex concepts, quickly learn, and learn from experience. Currently, it is believed that if we can simulate the human brain and copy all its neurons and synapses on the same scale, strong AI will naturally occur.
- Now we are in the weak AI phase. The emergence of weak AI alleviates human intellectual labor, similar to advanced bionics. Both AlphaGo and robots that can write press releases and novels fall in the weak AI phase because they are better than humans only in some ways. The roles of data and computing power are self-evident in the era of weak AI, and they promote the commercialization of AI. In the era of strong AI, these two factors are still critical. At the same time, the research on quantum computing by technology giants like Google and IBM also provides powerful support for humans to enter the era of strong AI.

AI Industry Ecosystem

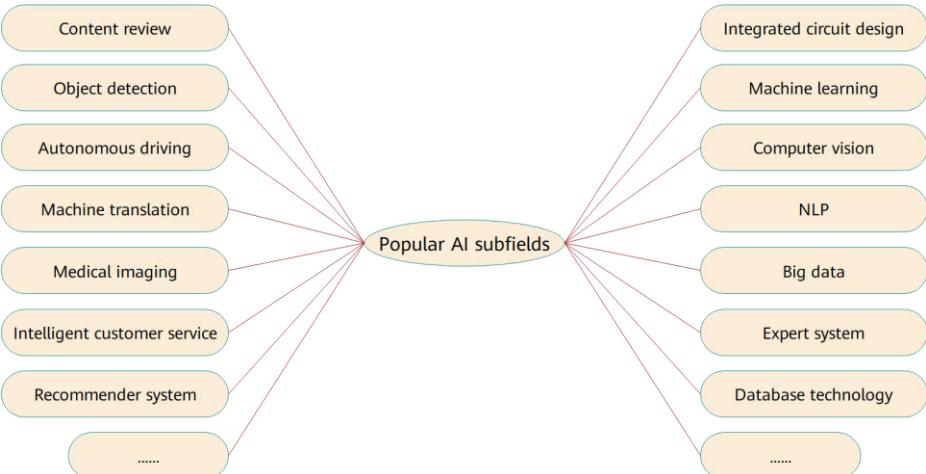
- Data, algorithms, computing power, and application scenarios are the basic elements of AI applications. We must combine AI with premium cloud computing, big data, and IoT to facilitate our intelligent society.



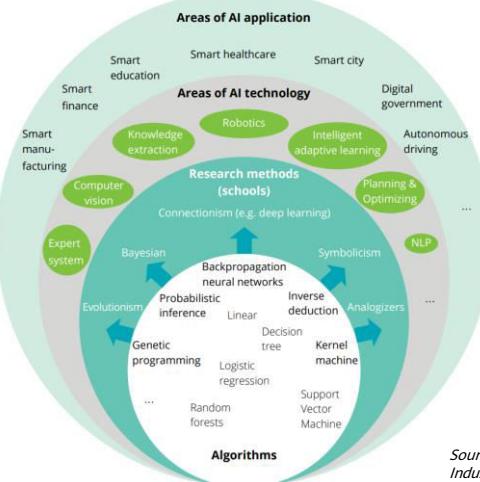
Contents

1. AI Overview
- 2. Application Fields of AI**
3. Huawei's AI Development Strategy
4. Controversies Over AI and Its Future

Popular AI Subfields



AI Technology and Application Fields

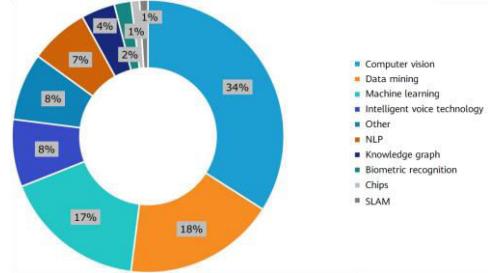


Source: Deloitte Global Artificial Intelligence Industry Whitepaper



Distribution of AI Application Technologies in China

- Computer vision, robotics, natural language processing (NLP), machine learning, and biometric recognition are the most popular technology fields for China's AI enterprises.
- Currently, the directions of general AI technologies are:
 - Computer vision
 - The study to make computers see things fast and accurately.
 - NLP
 - The study making computers understand and use natural languages.



Source: EqualOcean 2022 China AI chips Industry Research



- Technical layers: general technologies, algorithms, and technical frameworks
- General technologies: NLP, computer vision, speech recognition, machine learning, etc.
- Algorithms: machine learning, reinforcement learning, deep learning, etc.
- Technical frameworks: distributed storage, distributed computing, etc.

Smart Mining

Scenario 1: Mining chamber that houses important devices such as power and communications devices
Goal: **Automatic inspection and unattended operation.**

Challenge 1: The chamber environment needs to be monitored 24/7.



Challenge 2: Manual inspection must be performed periodically.



Scenario 2: Underground video surveillance system
Challenges

Coal conveyor belt	The length of the coal conveying belt exceeds 20 km . Manual inspection is time-consuming and insecure. Multiple belts need to work together to prevent coal accumulation.
Violation activities	Monitoring of personnel violations, such as not wearing safety helmets, smoking, passing under equipment, walking in roadways, and sitting on belts.
Violation operations	Gas drainage, water exploration and drainage.

21 Huawei Confidential

Solution for scenario 1: Automatic inspection robot

Video surveillance: Machine vision-based video surveillance for 24-hour monitoring of sensitive areas

Sound detection: AI sound detection techniques to intelligently monitor devices



Scenario 2

Belt tear monitoring



Belt deviation monitoring



Belt cleanliness monitoring



- <https://e.huawei.com/cn/material/wireless/4d008289c5424b31a055c71eaaed5790>

AI Safeguards Nature – A Protective Umbrella for Biodiversity in Chile

- The AI-powered "Nature Guardian" project has landed in the Nahuelbuta mountain range to study and protect the Darwin's fox. The "Nature Guardian", an acoustic monitoring system developed by the rainforest conservation organization Rainforest Connection (RFCx), has been deployed in several projects and is working effectively.
- It consists of solar devices equipped with microphones and antennas. These devices collect sound data from the surrounding environment and transmit it to the cloud through wireless networks for AI to analyze. Each device covers 3 square kilometers and runs around the clock.
- The "Nature Guardian" monitors gunshots from poachers, trucks and saws from illegal loggers, and cries from different animals to provide data for research.
- Trained AI analysis can identify the sounds of different animals, enabling experts to study the distribution and behavior of specific species and better implement environmental protection through adaptive management.
- If a threat is identified, the system sends a real-time alert to the ranger's mobile app, enabling the ranger to respond rapidly.



Image credit:
Buin Zoo & Nahuelbuta Foundation

Darwin's fox
Source: <https://www.huawei.com/en/tech4all/stories/nature-guardians-for-biodiversity-in-chile>



- <https://www.huawei.com/en/tech4all/stories/nature-guardians-for-biodiversity-in-chile>

AI Safeguards Nature – Protecting Wildlife in Greece with a Shield of Sound

23 Huawei Confidential



- <https://www.huawei.com/cn/tech4all/stories/wildlife-greece>

NLP Application Scenarios

- The research topics of natural language processing include machine translation, text mining, and sentiment analysis. Current NLP is technically difficult and not mature enough, and due to the highly-complex semantics, machines are unable to achieve the human understanding through deep learning based on big data and parallel computing.
- Development: From shallow semantics understanding to automatic feature extraction and deep semantics understanding; from single intelligence (ML) to hybrid intelligence (ML, DL, etc.).
- Application scenarios:

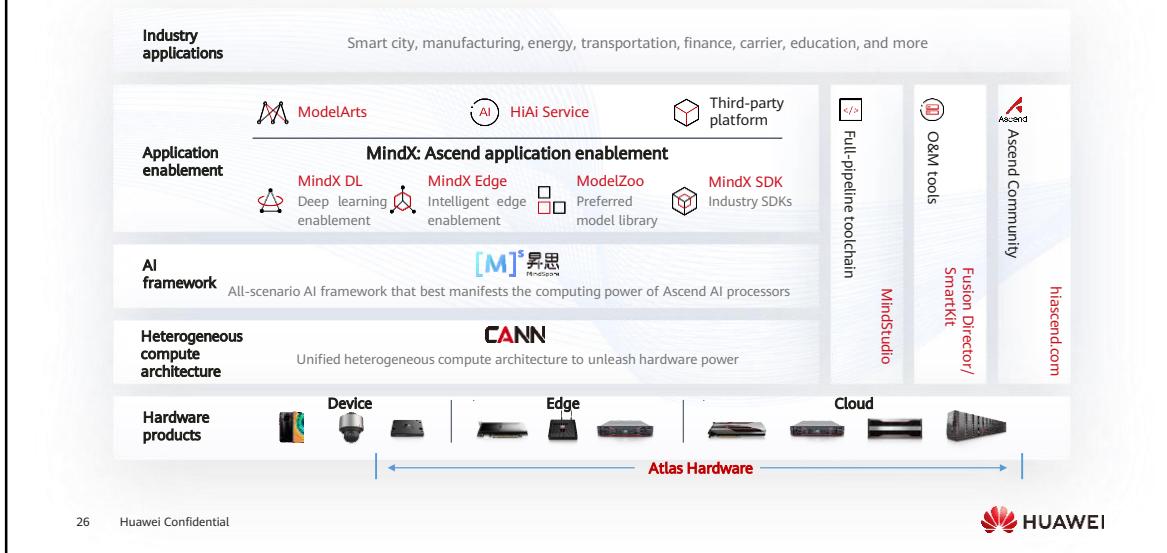


- Public opinion analysis: uses algorithm models such as keyword extraction, text clustering, and theme mining to mine emergencies and public opinion orientation, discover topics and trends, and analyze public opinions. It analyzes emotions, hot topics, trends, and propagation channels from multiple dimensions to master public opinion trends in a timely manner.
- Sentiment analysis: uses approaches such as emotion analysis and opinion extraction to extract emotional tendency and key opinion information from a large number of user comments.
- Knowledge graph: extracts knowledge based on NLP. For example, models are in Chinese, machine translation, and some medical materials are from English to Chinese. Text mining, extraction of central ideas, etc.

Contents

1. AI Overview
2. Application Fields of AI
- 3. Huawei's AI Development Strategy**
4. Controversies Over AI and Its Future

Huawei Full-Stack All-Scenario AI Solution



26 Huawei Confidential

- The MindSpore all-scenario AI computing framework went open source in the first quarter of 2020.
- "Full-stack" refers to its technical function. Huawei's full-stack portfolio includes chips, chip enablement, a training and inference framework, and application enablement.
- By "all scenarios", Huawei means different deployment scenarios for AI, including public clouds, private clouds, edge computing in all forms, industrial IoT devices, and consumer devices. As the cornerstone of the Huawei full-stack AI solution, Atlas provides modules, cards, and servers powered by Ascend AI Processors to meet customers' computing requirements in all scenarios.
- **Application enablement:** provides full-process services (**ModelArts**), layered APIs, and pre-integrated solutions.
- **MindSpore:** support for independent and collaborative device-edge-cloud training and inference frameworks
- **CANN:** a chip operator library and highly automated operator development tool.
- **Atlas:** all-scenario (device-edge-cloud) AI infrastructure solution powered by the Ascend series AI processors, available in various product forms.
- Huawei's all-scenario AI includes the deployment environments of the public cloud, private cloud, various forms of edge computing, IoT industry terminals,

and consumer terminals.

Ascend AI Processors Empower Smarter, Superior Computing



Ascend 310 series

Energy-efficient AI SoC



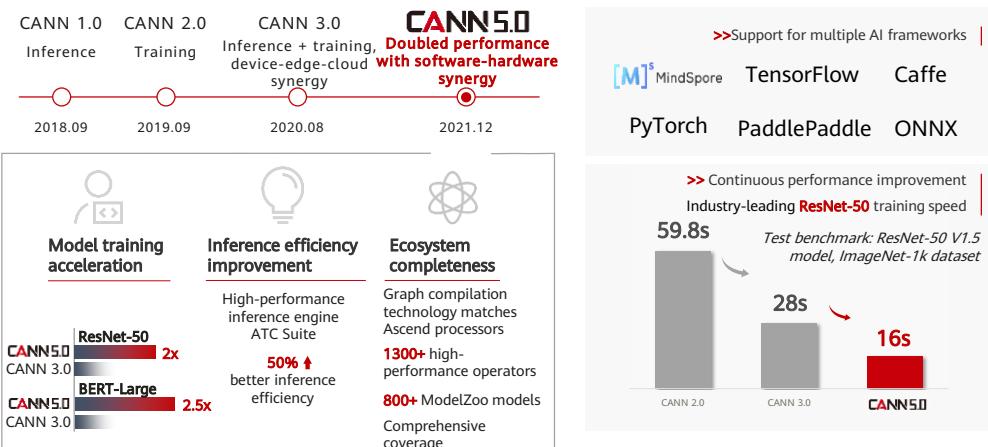
Ascend 910 series

Powerful AI Processor

- Atlas hardware that uses Ascend processors includes:
- Inference series: Atlas 200 accelerator module, Atlas 500 AI edge station, Atlas 300I Pro inference card, Atlas 300I Duo inference card, Atlas 500 Pro AI edge server, Atlas 800 inference server, Atlas 300V Pro video analysis card
- Training series: Atlas 300T Pro training card, Atlas 900 PoD, Atlas 900T PoD Lite, Atlas 800 training server, Atlas 900 AI cluster
- <https://www.hiascend.com/hardware/product>

CANN Heterogeneous Computing Architecture

Software and Hardware Synergy to Double Computing Power



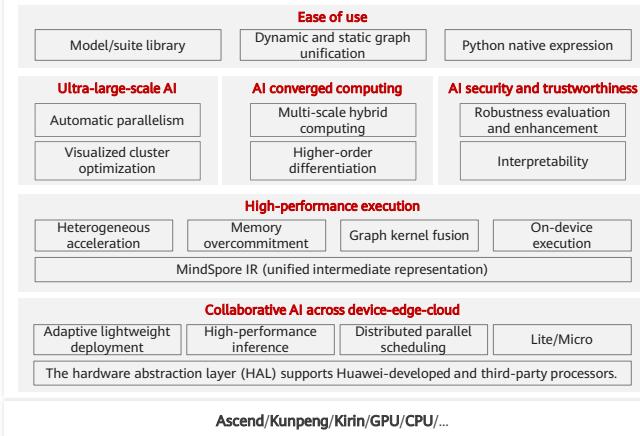
28 Huawei Confidential



- <https://www.hiascend.com/software/cann>

MindSpore AI Computing Framework

AI computing center, manufacturing, finance, Internet, transportation, government, energy, university scientific research, public safety, and carrier



29 Huawei Confidential

Key Features



Ease of use

Models for popular scenarios, Python native expression, and dynamic and static graph unification.



All-scenario AI

Diversified computing power, collaborative systems, and smart devices over edge and cloud.



High-performance execution

Full-stack collaborative acceleration and Ascend-affinitive execution engine to maximize performance.



Ultra-large-scale AI

Distributed parallel training of ultra-large trillion-parameter models, with a series of pre-trained influential models available.



AI converged computing

X-fold acceleration of AI+scientific computing converged applications for electromagnetic, biopharmaceutical, and other usages.



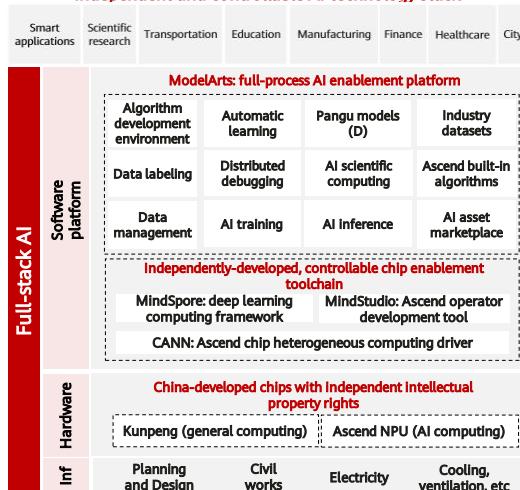
AI security and trustworthiness

Secure and trustworthy training, evaluation, and deployment of models.



ModelArts: Ascend-based Full-Stack Independence and Controllability

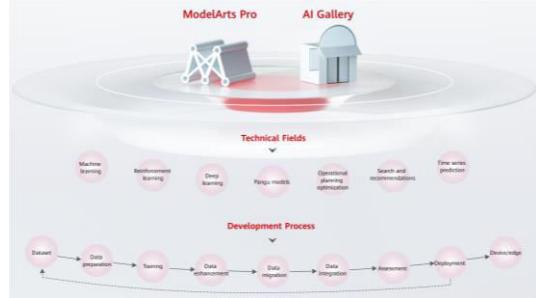
Independent and controllable AI technology stack



30 Huawei Confidential

ModelArts is a **one-stop** AI development platform that offers tools to help developers and data scientists of all skill sets build, train, and deploy models (cloud to edge), and manage full-lifecycle AI workflows. The platform makes it easier to develop AI by providing key capabilities, including data preprocessing and auto labeling, distributed training, automated model building, and one-click workflow execution.

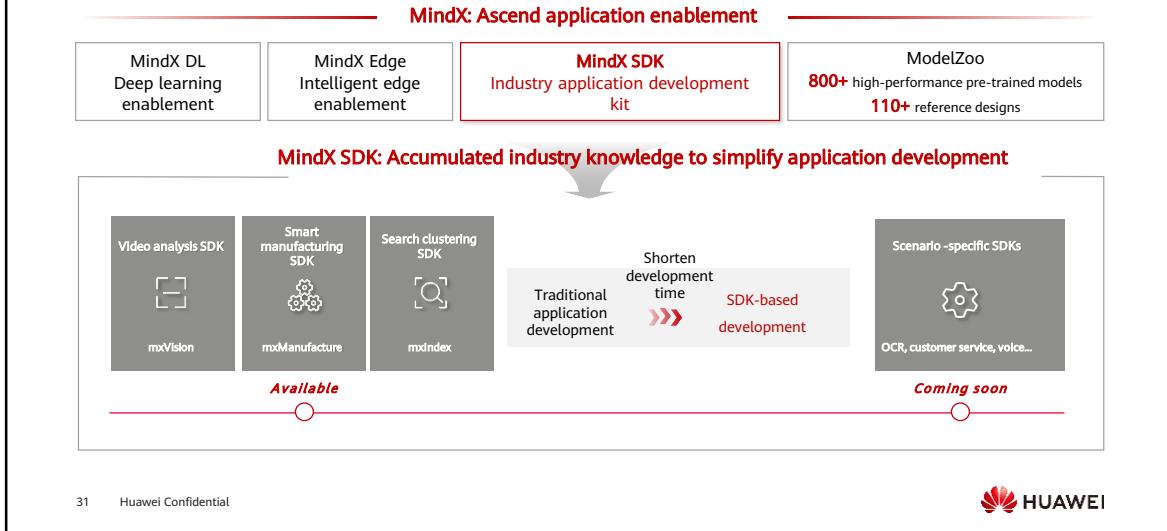
ModelArts Pro: Enterprise-Grade Development Platform for Specific Industry Needs



HUAWEI

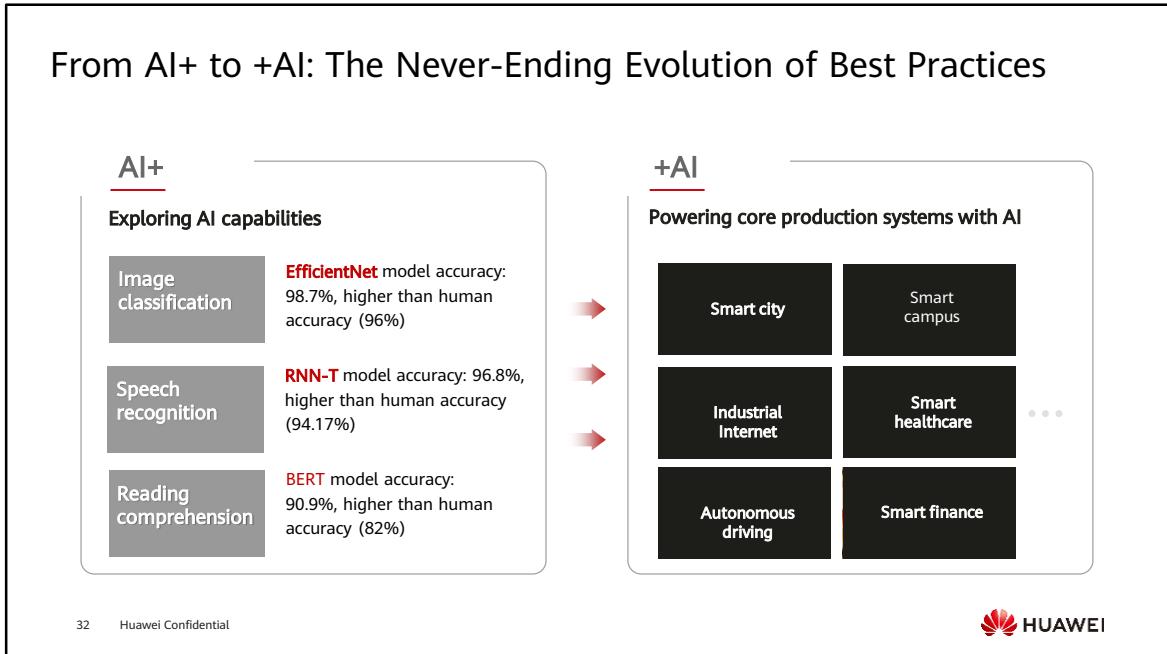
- For details about the ModelArts training process, visit <https://www.huaweicloud.com/product/modelarts.html>.

MindX: Application Enablement with Ascend



- Ascend promotional theme slides:
- MindX DL is an Ascend deep learning reference architecture that helps you develop deep learning systems. Running on Atlas 800 training and inference servers, it manages and monitors Ascend AI Processor resources, schedules Ascend AI Processors, and generates collective communication configurations for distributed training.
- ModelZoo** provides developers with a rich array of scenario-specific pre-trained models, freeing developers from model selection, training, and tuning.
- MindX SDK is a complete AI development suite oriented to industry scenarios. It provides easy-to-use APIs and a graphical user interface (GUI), enabling developers to quickly develop industry AI applications with little code.
- MindX Edge is an edge component for compute resource management and O&M. It enables industry customers to quickly build edge-cloud collaborative inference platforms.

From AI+ to +AI: The Never-Ending Evolution of Best Practices



- Traditional AI applications are single-technology applications, such as image recognition and classification and speech recognition and cloning. These applications represent the exploration of AI capabilities.
- As AI applications transform from single-modal to multi-modal and knowledge computing solutions go from perception to cognition, AI has entered core production systems of enterprises. The deep integration of AI and enterprise digitization has inspired systematic intelligent solutions such as smart city and industrial Internet, creating greater value. Image classification models: EfficientNet, YOLO, and more. In 2019, Google launched EfficientNet.
- Speech recognition: Recurrent neural network transducer (RNN-T).
- An acoustic model based on connectionist temporal classification (CTC) does not require the audio and text sequences for training to be aligned. Such a model has preliminarily possessed the end-to-end acoustic modeling capability. However, the CTC model has two severe bottlenecks when performing acoustic modeling. One is that the CTC model lacks the language modeling capability and cannot integrate language models for joint optimization. The other is that the CTC model cannot model the dependency between model outputs. RNN-T is developed to make up for weaknesses of CTC, so that the model gains outstanding advantages such as end-to-end joint optimization, language modeling capability, and easy implementation of online speech recognition. RNN-T is more suitable for speech tasks.

Contents

1. AI Overview
2. Application Fields of AI
3. Huawei's AI Development Strategy
- 4. Controversies Over AI and Its Future**
 - **Controversies Over AI**
 - The Future of AI

Controversies Over AI

- Every year we are gainin a clearer understanding of underlying AI concepts. However, many are concerned about the future of AI:
 - Do we still see and hear reality?
 - What are the ethical issues of AI?
 - Will AI take over our jobs?
 - Will our privacy be violated? How to ensure privacy security?
 - Is future AI controllable?
 -

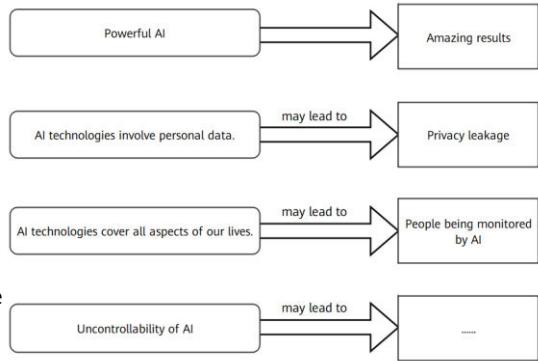
- What you see may not be true.
- Ethical issues can be constrained by policies and regulations.
- AI will take over some of the jobs.
- Privacy breach risks exist but can be mitigated through technologies such as differential privacy training, model fingerprint, and model encryption.
- Controllability - can be left to trainees to think

Seeing Is Not Believing

- Breakthroughs in computer vision, such as photoshop and generative adversarial network (GAN), means now we can generate fake images that indistinguishable from real photos, impacting the credibility of image and video data.
- For example:
 - Weight-loss shops use Photoshop to produce fake images that show fast weight loss.
 - Lyrebird is a tool that can clone human voices based on a few minutes of recording samples. Such a tool could be exploited for malicious purposes.
 - Room pictures on rental or hotel booking platforms could be fake and generated by a GAN.

Ethical Concerns of AI

- Discrimination and stigmatization: AI can be used to impact on social justice and freedom.
- The autonomous decision-making mechanism of AI: This mechanism endangers human autonomy and the results cannot be traced or reproduced to provide effective remedies for incorrect decisions.
- Defects of AI: The objectives, methods, and decisions of AI systems cannot be interpreted.
- Damage to human dignity, minds, and bodies: Robots are vulnerable to malicious use and can be altered to make environments unsafe, while AI can expose people to deception and manipulation.



Will AI take over our jobs?

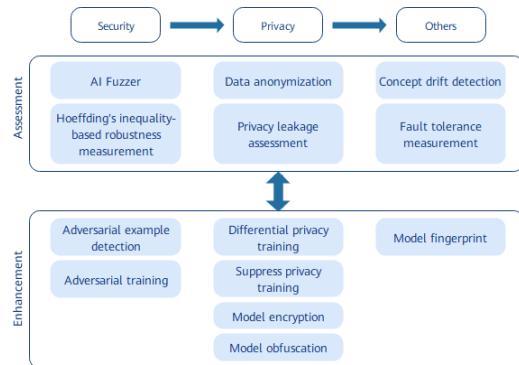
- Throughout the course of history, humans have sought ways to improve efficiency, that is, to obtain more from fewer resources. Just as sharp stones improve hunting, and steam engines reduce the demand for horses, every step towards automation has changed our society. In the AI era, what tasks will be taken over by AI?
- AI may take over:
 - Repeatable tasks
 - Non-creative tasks
 - Tasks that require weak social skills
 - Dangerous tasks
 -

Privacy and Data Security

- In the era of big data and AI, we can obtain information with convenience and efficiency. However, while we are obtaining convenience, our behavior is recorded, learned, and used for different purposes.

How can we **protect private data**?

- 1. End users need to read **privacy terms** before giving consent to a mobile app to collect user information, and limit the information available to the app.
- 2. **Privacy protection laws and regulations should be developed**, to implement industry code of conduct and strengthen supervision.
- 3. **Confidential computing, model privacy protection, federated learning, or adversarial learning**, in addition to protection frameworks, such as **MindArmour**, can prevent reverse attacks and protect user privacy.



- <https://gitee.com/mindspore/mindarmour>
- <https://mindspore.cn/mindarmour?version=/r1.5/>
- Confidential computing: Data transmission and computing are confidential. Privacy protection is costly.
- Model privacy protection: Common methods include data anonymization, differential privacy training, model encryption, and model obfuscation. (Differential privacy can measure and control the leakage of training data by models.)
- Federated learning: Multi-party joint machine learning is performed without sharing data. Federated learning is essentially a distributed machine learning framework with limited data access. Compute nodes share only gradients but not raw data.

Issues to Be Resolved

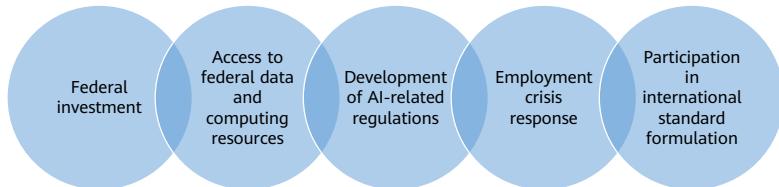
- Are AI-generated content protected by copyright laws?
- Who will grant rights to robots?
- What rights can be granted to robots?
-

Contents

1. AI Overview
2. Application Fields of AI
3. Huawei's AI Development Strategy
- 4. Controversies Over AI and Its Future**
 - Controversies Over AI
 - **The Future of AI**

Policy Trends - United States (1)

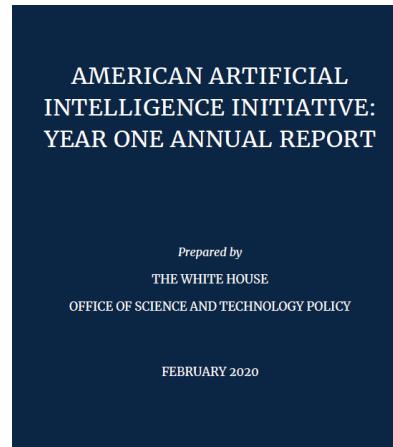
- Executive Order - American Artificial Intelligence Initiative
 - In February 2019, the US released the *American Artificial Intelligence Initiative*, which states that the US will take various ways to improve American leadership in AI. It covers federal investment, access to federal data and computing resources, development of AI-related regulations, employment crisis response, and participation in formulation of international standards. It specifies the AI development direction of the US in the future, proposes key propositions in fields such as smart healthcare and smart city, and clearly expresses the exclusion of cross-border acquisitions of key AI technologies from competitive countries.



- Maintaining American Leadership in Artificial Intelligence:
<https://www.federalregister.gov/documents/2019/02/14/2019-02544/maintaining-american-leadership-in-artificial-intelligence> (February 14, 2019)

Policy Trends - United States (2)

- Main Content of *American Artificial Intelligence Initiative: Year One Annual Report*:
 1. Invest in AI Research and Development
 2. Unleash AI Resources
 3. Remove Barriers to AI Innovation
 4. Train an AI-ready Workforce
 5. Promote an International Environment Supportive of American AI Innovation
 6. Embrace Trustworthy AI for Government Services and Missions



- <https://www.nitrd.gov/nitrdgroups/images/c/c1/American-AI-Initiative-One-Year-Annual-Report.pdf>

Policy Trends - United States (3)

- To promote technological innovation and scientific research development in the US, the US government passed the *United States Innovation and Competition Act of 2021* in June 2021. The act covers multiple topics including chips, 5G, aerospace, cyber security and AI, medical research, and American manufacturing. The act proposes to reform the **National Science Foundation (NSF)** and authorizes \$81 billion for it to conduct research in 10 key fields including AI and computer technology. In addition, \$10 billion is authorized for technology centers and innovation research institutes of universities for research in related fields.



- United States Innovation and Competition Act of 2021:
<https://www.congress.gov/bill/117th-congress/senate-bill/1260>

Policy Trends - China (1)

- In July 2017, AI was first written into the government's work report. The **New Generation of Artificial Intelligence Development Plan** issued by the State Council proposed a three-step strategic goal:
- Step 1: By 2020, the overall technology and application of AI will be in step with globally advanced levels.
- Step 2: By 2025, achieve major breakthroughs in basic theories for AI, and be a world-leading player for technologies and applications.
- Step 3: By 2030, realize world-leading status of nationally-developed AI, technologies, and applications, ensure China is the world's primary AI innovation center, achieve tangible results in intelligent economy and intelligent society, and lay a core to become a leading innovation-style nation and an economic power.

索引号：00001439/2017-00147
公开方式：主动公开
文号：国办发〔2017〕35号
发布日期：2017年07月20日
实施日期：2017年07月20日
主 题词：

国务院关于印发
新一代人工智能发展规划的通知
国发〔2017〕35号
各省、自治区、直辖市人民政府，国务院各部委、各直属机构：
现将《新一代人工智能发展规划》印发给你们，请认真贯彻执行。
国务院
2017年7月8日
(此件公开发布)

新一代人工智能发展规划

人工智能的迅速发展深刻改变人类社会生活、改变世界。为抢抓人工智能发展的重大战略机遇，构筑我国人工智能发展的先发优势，加快建设创新型国家和世界科技强国，按照党中央、国务院部署要求，制定本规划。

一、战略态势

人工智能发展进入新阶段。经过60多年的演进，特别是在移动互联网、大数据、超级计算、传感网、脑科学等新理论新技术以及经济社会发展强烈需求的共同驱动下，人工智能加速发展，呈现出深度学习、跨界融合、人机协同、群体智能、自主操控等新特征。大力建设主动学习、跨媒体协同处理、人机协同增强智能、群体集成智能、自主智能系统成为人工智能发展的主要方向。人工智能研究成果层出不穷，类脑智能蓄势待发，芯片化模块化平台化趋势更加明显，人工智能发展进入新阶段。当前，新一代人工智能相关学科发展、理论建模、技术创新、标准体系构建等整体推进，正在引发新一轮技术突破，推动经济社会各领域从数字化、网络化向智能化加速跃升。

相关报道

- * 国务院印发《新一代人工智能发展规划》

图解

- * 国务院印发《新一代人工智能发展规划》

解读

- * 总理政府工作报告带火的这件事，有了国家规划！
- * 构筑人工智能先发优势 把握新一轮科技革命战略主动 ——科技部党组书记、副部长王志刚解读我国首个人工智能发展规划纲要
- * 大智能时代的关键之举——五问AI国家战略



- http://www.gov.cn/zhengce/content/2017-07/20/content_5211996.htm

Policy Trends - China (2)

- The Outline of the 14th Five-Year Plan for Economic and Social Development and Long-Range Objectives Through the Year 2035 of the People's Republic of China was released in March 2021, which emphasized that China "will explore the establishment of regulatory frameworks for autonomous driving, online medical care, financial technology, and intelligent delivery, and improve the relevant laws, regulations, and rules for ethical reviews". The statements reflect China's priority for and determination to overhaul AI ethics and its governance.
- The infringement on personal information is becoming more rampant, and there is an urgent need to protect personal information. To protect the rights and interests on personal information, regulate its processing, and guide its reasonable use, Personal Information Protection Law of People's Republic of China was passed on August 20th, 2021 and came into force on November 1st, 2021.

- The Outline of the 14th Five-Year Plan for Economic and Social Development and Long-Range Objectives Through the Year 2035 of the People's Republic of China (2021-03-11)[2021-07-15]. http://www.gov.cn/xinwen/2021-03/13/content_5592681.htm.
- Personal Information Protection Law: http://en.npc.gov.cn/cdurl.cn/2021-12/29/c_694559.htm

AI in More Industry Practices

- With further breakthroughs in AI in various vertical industries, AI technologies will be used in an increasing number of scenarios.
 - Smart transportation - highways, airports, railways, logistics, ports...
 - Energy - electric power, oil and gas...
 - Finance - bank, insurance, securities...
 -

- Digital government: Smart city, smart government, smart emergency response, transparent kitchen, and smart water conservancy. For example, the "transparent kitchen" regulation uses AI machine vision to identify food types on meal plates and calculate prices and calories. Smart water conservancy: water governance and sewage monitoring (AI video algorithm sewage identification, satellite remote sensing monitoring, and water pump and sluice scheduling), reservoir (Eco-Cube enhanced video, intelligent power supply, and integrated pole sites), and rivers and lakes (spectrographic water quality analysis, 5G UVAs/unmanned boats)
- Smart mine: underground 5G, security monitoring, violation detection.
- Smart transportation: airport (IOC queuing management that percepts ultra-long passenger queuing using AI + video analysis, and stand allocating that uses AI algorithm + big data + IoT + GIS + video + simulation...), highway (holographic intersection, autonomous driving...)

Quiz

- 1. What are the AI application scenarios in these vertical fields? What technologies (not limited to AI) are used in these scenarios?
- 2. Based on the policy trends,summarize the core concerns for future AI development of each country.

- The core concerns include trustworthy AI, AI theory research, AI application development, privacy protection, and basic computing power development.

Summary

- This course introduces the definition and development history of AI, describes the popular fields and application scenarios of AI, briefly introduces Huawei's AI development strategy, and discusses its most pressing topics and trends.

Quiz

1. (Multiple-answer question) Which of the following are Huawei's full-stack all-scenario AI solutions?
 - A. ModelArts
 - B. MindSpore
 - C. Atlas 800 Servers
 - D. ModelZoo

- Answer:
 - ABCD

Recommendations

- Huawei Talent
 - <https://e.huawei.com/en/talent/#/home>
- Ascend official website
 - <https://www.hiascend.com/>
- MindSpore
 - <https://mindspore.cn/>
- Huawei Cloud
 - <https://www.huaweicloud.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。

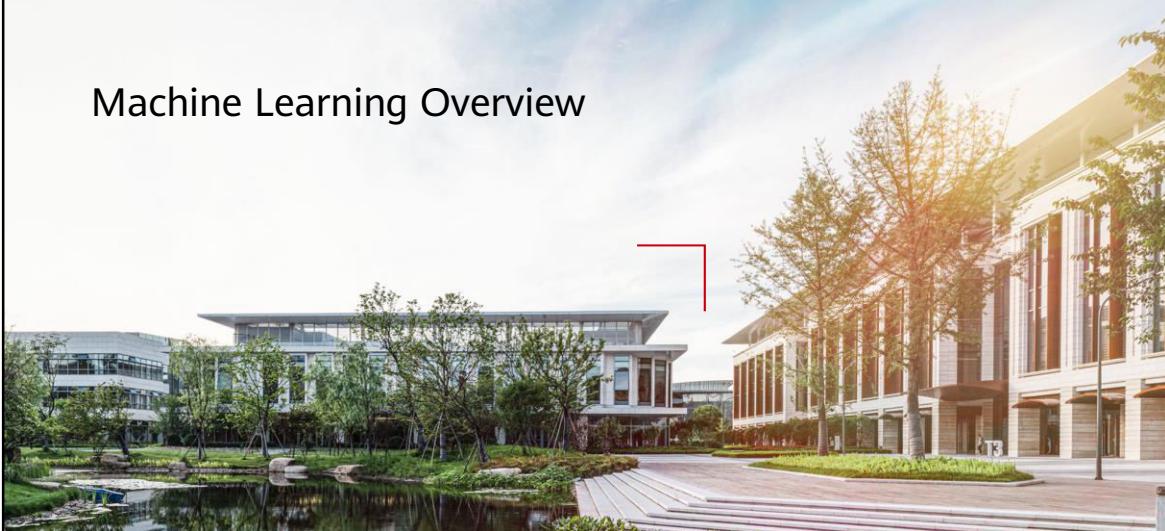
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2023 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



Objectives

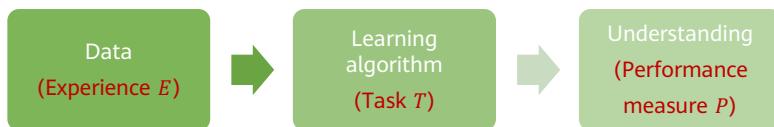
- Upon completion of this course, you will understand:
 - Learning algorithm definitions and machine learning process
 - Related concepts such as hyperparameters, gradient descent, and cross-validation
 - Common machine learning algorithms

Contents

- 1. Machine Learning Algorithms**
2. Types of Machine Learning
3. Machine Learning Process
4. Important Machine Learning Concepts
5. Common Machine Learning Algorithms

Machine Learning Algorithms (1)

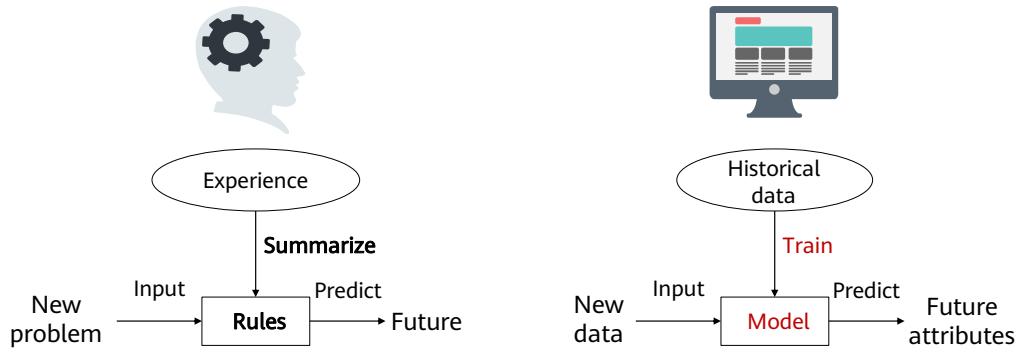
- Machine learning is often combined with deep learning methods to study and observe AI algorithms. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .



- Deep learning is a sub-field of machine learning. To understand deep learning, you need to first understand the fundamentals of machine learning.
 - Task T represents how the machine learning system should process a sample.
 - Performance measure P is used to evaluate aspects of machine learning such as accuracy and error rate.
 - Experience E : Most learning algorithms can be understood as being allowed to experience an entire dataset. Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its training process. Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process.
- To learn the game of Go:
 - Experience E_1 :Game with itself- Unsupervised and indirect learning
 - Experience E_2 :Inquiring Humans during a game with itself – Semi-supervised learning
 - Experience E_3 :Learning from human historical games - Supervised and

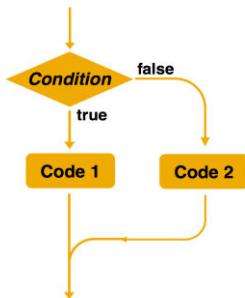
direct learning

Machine Learning Algorithms (2)



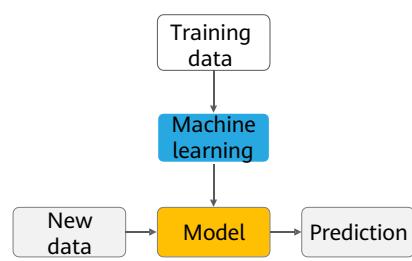
Differences Between Machine Learning Algorithms and Traditional Rule-based Methods

Rule-based method



- Explicit programming is used to solve problems.
- Rules can be manually determined.

Machine learning



- Models are trained on samples.
- Decision-making rules are complex or difficult to describe.
- Machines automatically learn rules.

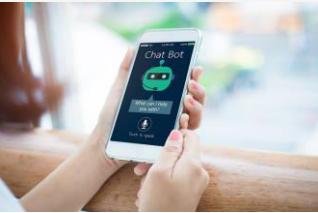
When to Use Machine Learning (1)

- Machine learning provides solutions to complex problems, or those involving a large amount of data whose distribution function cannot be determined.
- Consider the following scenarios:

Rules are complex or difficult to describe, for example, speech recognition.



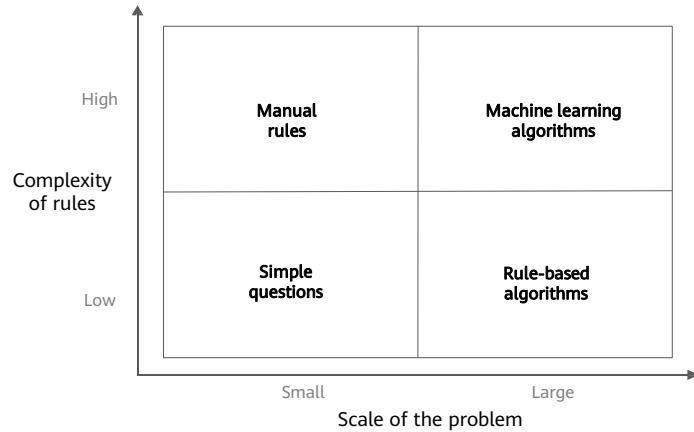
Task rules change over time, for example, part-of-speech tagging, in which new words or word meanings can be generated at any time.



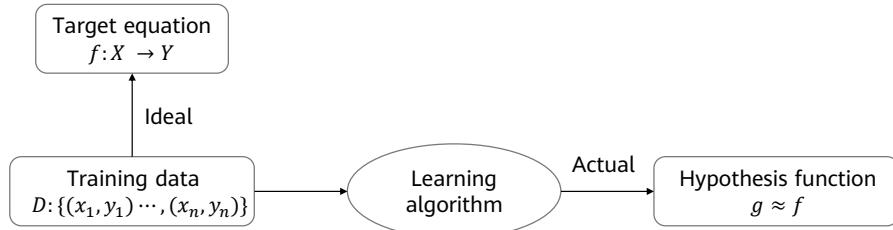
Data distribution changes over time and programs need to adapt to new data constantly, for example, sales trend forecast.



When to Use Machine Learning (2)



Rationale of Machine Learning Algorithms



- The objective function f is unknown, and the learning algorithm cannot obtain a perfect function f .
- Hypothesis function g approximates function f , but may be different from function f .

Main Problems Solved by Machine Learning

- Machine learning can solve many types of tasks. Three most common types are:
 - **Classification:** To specify a specific one of the k categories for the input, the learning algorithm usually outputs a function $f: R^n \rightarrow \{1, 2, \dots, k\}$. For example, image classification algorithms in computer vision solve classification tasks.
 - **Regression:** The program predicts the output for a given input. The learning algorithms usually output a function $f: R^n \rightarrow R$. Such tasks include predicting the claim amount of a policy holder to set an insurance premium or predicting the security price.
 - **Clustering:** Based on internal similarities, the program groups a large amount of unlabeled data into multiple classes. Same-class data is more similar than data across classes. Clustering tasks include search by image and user profiling.
- **Classification and regression are two major types of prediction tasks. The output of classification is discrete class values, and the output of regression is continuous values.**



 HUAWEI

Contents

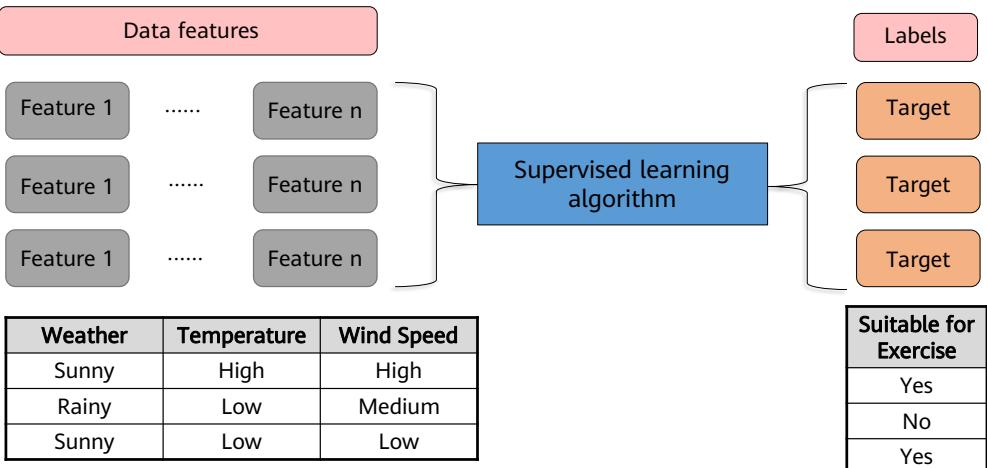
1. Machine Learning Algorithms
- 2. Types of Machine Learning**
3. Machine Learning Process
4. Important Machine Learning Concepts
5. Common Machine Learning Algorithms

Types of Machine Learning

- **Supervised learning:** The program takes a known set of samples and trains an optimal model to generate predictions. Then, the trained model maps all inputs to outputs and performs simple judgment on the outputs. In this way, unknown data is classified.
- **Unsupervised learning:** The program builds a model based on unlabeled input data. For example, a clustering model groups objects based on similarities. Unsupervised learning algorithms model the highly similar samples, calculate the similarity between new and existing samples, and classify new samples by similarity.
- **Semi-supervised learning:** The program trains a model through a combination of a small amount of labeled data and a large amount of unlabeled data.
- **Reinforcement learning:** The learning systems learn behavior from the environment to maximize the value of reward (reinforcement) signal function. Reinforcement learning differs from supervised learning of connectionism in that, instead of telling the system the correct action, the environment provides scalar reinforcement signals to evaluate its actions.
- Machine learning evolution is producing new machine learning types, for example, self-supervised learning, contrastive learning, generative learning.

- Supervised learning: We give a computer a bunch of choice questions (training samples) and provide standard answers. The computer tries to adjust its model parameters to make their predictions close to standard answers. In this way, the computer learns how to deal with this type of problem. Then the computer can help us reply to choice questions whose answers are not given (test samples).
- Unsupervised learning: We give a computer a bunch of choice questions (training samples), but do not provide standard answers. The computer tries to analyze the relationships between these questions and classify them. It does not know the answers to these questions, but it thinks that the answers to the questions in the same category should be the same.
- Semi-supervised learning: Traditional supervised learning uses a large number of labeled training samples to create a model for predicting the labels of new samples. For example, in a classification task, a label indicates the category of a sample while in a regression task, a label is a real-value output of the sample. As data storage capabilities are developing, we can easily obtain a large amount of unlabeled data from many tasks. However, labeling the data is labor-consuming and time-consuming. For example, for web page recommendation, users need to label web pages they like, but only a few users are willing to spend a lot of time doing this. Then we get limited labeled web page data and a large amount of unlabeled web page data.

Supervised Learning



Supervised Learning - Regression

- **Regression** reflects the features of sample attributes in a dataset. A function is used to express the sample mapping relationship and further discover the dependency between attributes. Examples include:
 - How much money can I make from stocks next week?
 - What will the temperature be on Tuesday?

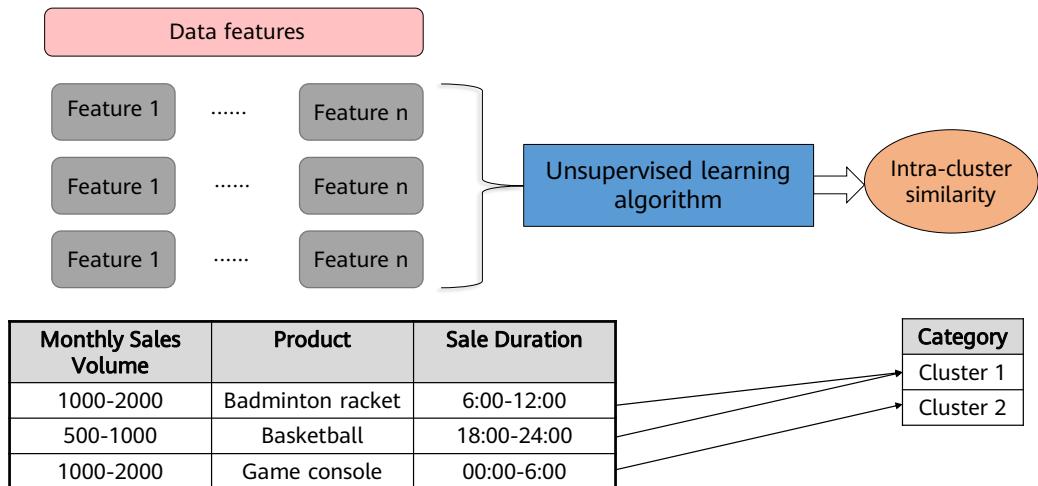
Monday	Tuesday
38°	?

Supervised Learning - Classification

- **Classification** uses a classification model to map samples in a dataset to a given category.
 - What category of garbage does the plastic bottle belong to?
 - Is the email a spam?



Unsupervised Learning

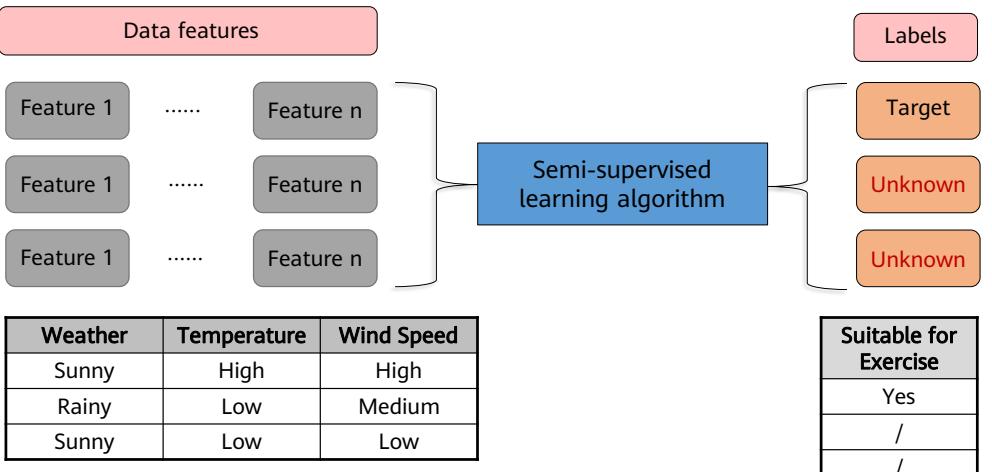


Unsupervised Learning - Clustering

- **Clustering** uses a clustering model to classify samples in a dataset into several categories based on similarity.
 - Defining fish of the same species.
 - Recommending movies for users.

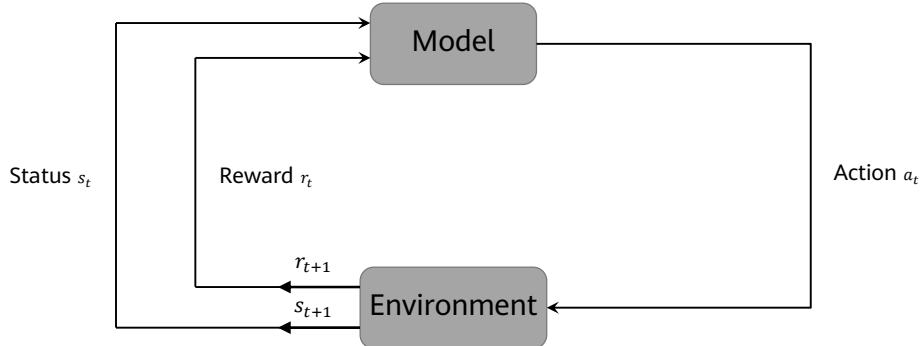


Semi-supervised Learning



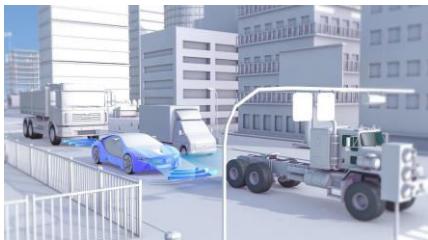
Reinforcement Learning

- A reinforcement learning model learns from the environment, takes actions, and adjusts the actions based on a system of rewards.



Reinforcement Learning - Best Action

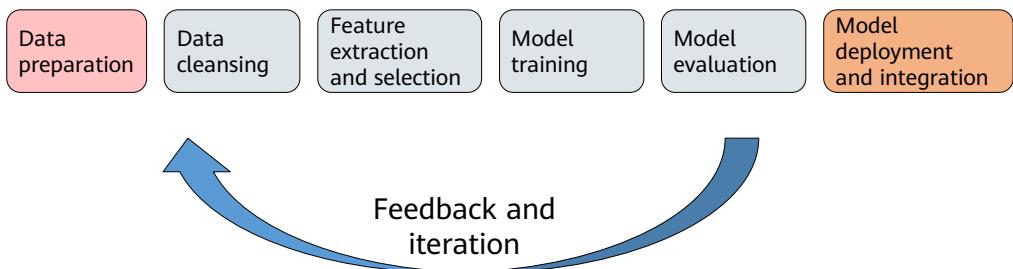
- **Reinforcement learning** always tries to find the best action.
 - Autonomous vehicles: The traffic lights are flashing yellow. Should the vehicle brake or accelerate?
 - Robot vacuum: The battery level is 10%, and a small area is not cleaned. Should the robot continue cleaning or recharge?



Contents

1. Machine Learning Algorithms
2. Types of Machine Learning
- 3. Machine Learning Process**
4. Important Machine Learning Concepts
5. Common Machine Learning Algorithms

Machine Learning Process



Machine Learning Basic Concept - Dataset

- **Dataset:** collection of data used in machine learning tasks, where each piece of data is called a sample. Items or attributes that reflect the presentation or nature of a sample in a particular aspect are called **features**.
 - **Training set:** dataset used in the training process, where each sample is called a training sample. **Learning (or training)** is the process of building a model from data.
 - **Test set:** dataset used in the testing process, where each sample is called a test sample. Testing refers to the process, during which the learned model is used for prediction.

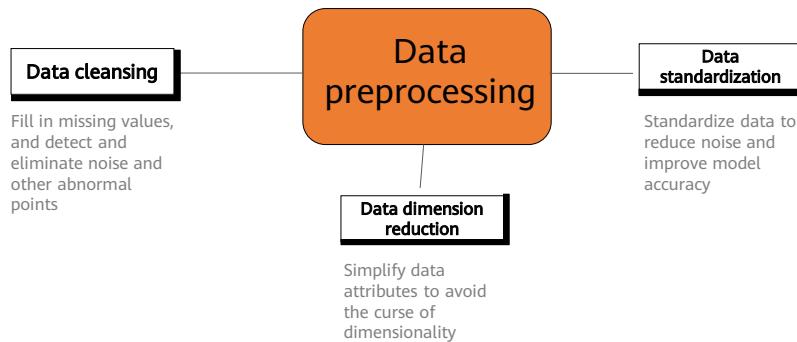
Data Overview

- Typical dataset composition

	Feature 1	Feature 2	Feature 3	Label	
No.	Area	Location	Orientation	House Price	
Training set	1	100	8	South	1000
	2	120	9	Southwest	1300
	3	60	6	North	700
	4	80	9	Southeast	1100
Test set	5	95	3	South	850

Importance of Data Processing

- Data is crucial to models and determines the scope of model capabilities. All good models require good data.



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, only preprocessed data can be used by algorithms. Data preprocessing involves the following operations:
 - Data filtering
 - Data loss handling
 - Handling of possible error or abnormal values
 - Merging of data from multiple sources
 - Data consolidation

Dirty Data

- Raw data usually contains data quality problems:
 - Incompleteness: Incomplete data or lack of relevant attributes or values.
 - Noise: Data contains incorrect records or abnormal points.
 - Inconsistency: Data contains conflicting records.

#	Id	Name	Birthday	Gender	IsTeacher?	#Students	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	Madrid
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	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Italy	Rome

Annotations pointing to specific errors:

- Invalid duplicate items: Points to row 6 where Id 555 appears again.
- Incorrect format: Points to row 6 where Birthday is in an invalid format (1983-12-01).
- Dependent attributes: Points to row 10 where City (Italy) is misspelled.
- Missing value: Points to row 2 where City is missing.
- Invalid value: Points to row 5 where Gender is set to 'A' instead of 'M' or 'F'.
- Misfielded value: Points to row 7 where City is listed as 'Italy' instead of 'Rome'.
- Misspelling: Points to row 10 where City is misspelled as 'Italy'.

Data Conversion

- Preprocessed data needs to be converted into **a representation suitable for machine learning models**. The following are typically used to convert data:
 - Encoding categorical data into numerals for classification
 - Converting numeric data into categorical data to reduce the values of variables (for example, segmenting age data)
 - Other data:
 - Embedding words into text to convert them into word vectors (Typically, models such as word2vec and BERT are used.)
 - Image data processing, such as color space conversion, grayscale image conversion, geometric conversion, Haar-like features, and image enhancement
 - Feature engineering:
 - Normalizing and standardizing features to ensure that different input variables of a model fall into the same value range
 - Feature augmentation: combining or converting the existing variables to generate new features, such as averages.

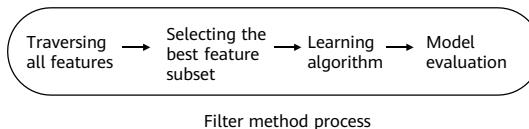
Necessity of Feature Selection

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



Feature Selection Methods - Filter

- Filter methods are independent of models during feature selection.



Filter method process

By evaluating the correlation between each feature and target attribute, a filter method scores each feature using a statistics measurement and then sorts the features by score. This can preserve or eliminate specific features.

Common methods:

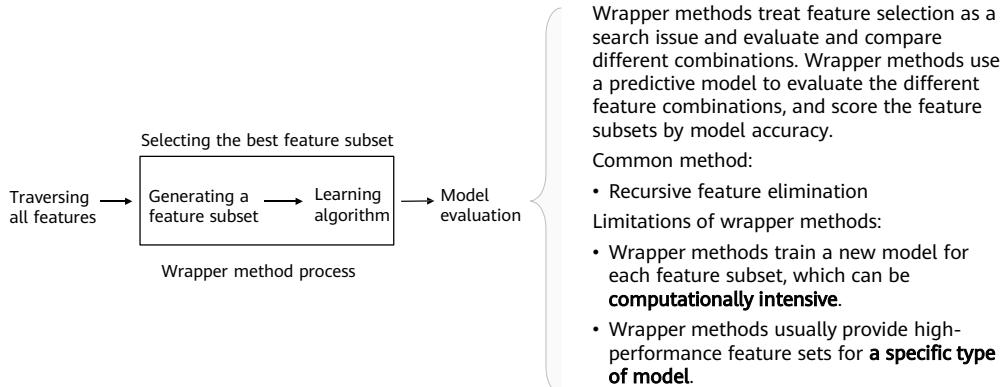
- Pearson correlation coefficient
- Chi-square coefficient
- Mutual information

Limitations of filter methods:

- Filter methods tend to select redundant variables because they do not consider the relationships between features.

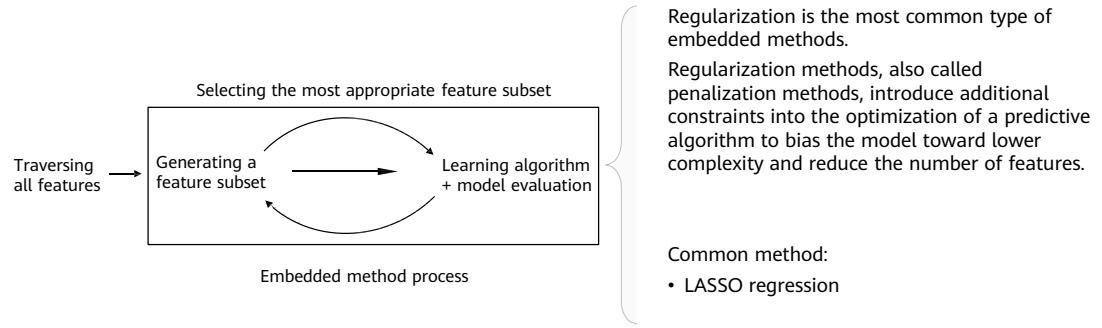
Feature Selection Methods - Wrapper

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



Feature Selection Methods - Embedded

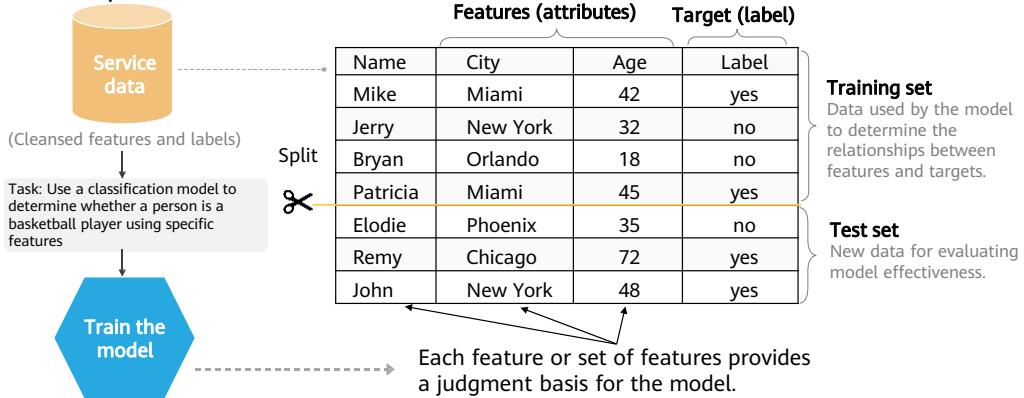
- Embedded methods treat feature selection as a part of the modeling process.



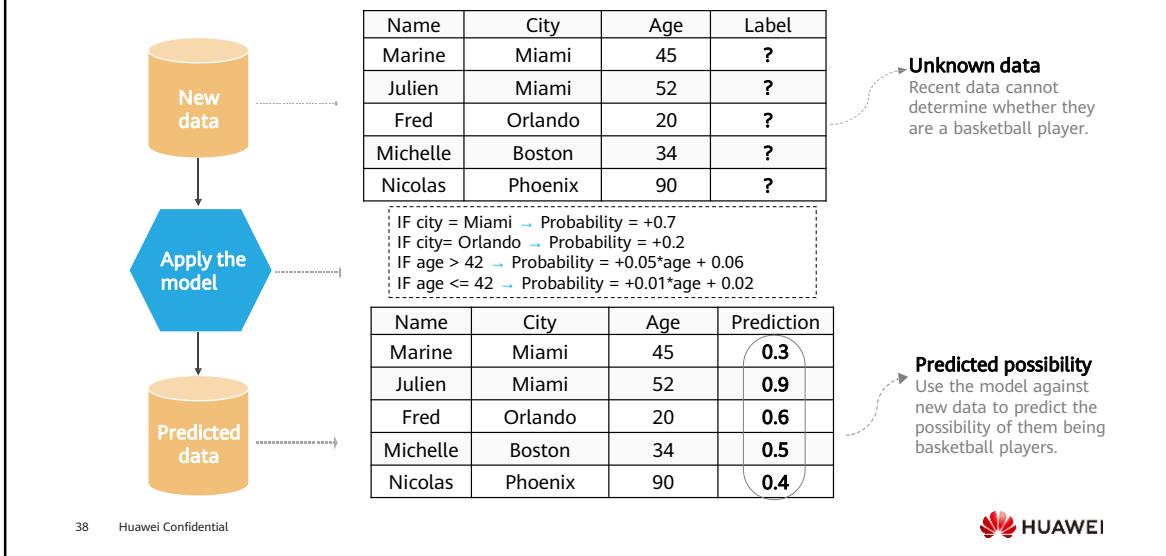
- The penalty attribute of the model is highlighted, which is used to eliminate insignificant features.

Supervised Learning Example - Learning Phase

- Use a classification model to determine whether a person is a basketball player based on specific features.



Supervised Learning Example - Prediction Phase



- Model example
- The model here is similar to the decision tree model.

What Is a Good Model?



- **Generalization**
The accuracy of predictions based on actual data
- **Explainability**
Predicted results are easy to explain
- **Prediction speed**
The time needed to make a prediction

- Which factors are used to judge a model?
- Generalization capability is the most important.

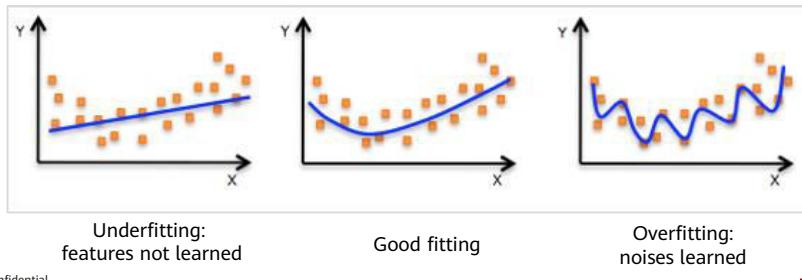
Model Effectiveness (1)

- **Generalization capability:** Machine learning aims to ensure models perform well on new samples, not just those used for training. Generalization capability, also called **robustness**, is the extent to which a learned model can be applied to new samples.
- **Error** is the difference between the prediction of a learned model on a sample and the actual result of the sample.
 - Training error is the error of the model on the training set.
 - Generalization error is the error of the model on new samples. Obviously, we prefer a model with a smaller generalization error.
- **Underfitting:** The training error is large.
- **Overfitting:** The training error of a trained model is small while the generalization error is large.

- Once the form of a problem's hypothesis is given, all possible functions constitute a space, which is hypothesis space. The problem of machine learning is searching for a suitable fitting function in a hypothesis space.
- Overfitting: It occurs frequently in complex mathematical models. To prevent overfitting, we can simplify mathematical models, end training before overfitting, or use dropout/weight decay methods.
- Underfitting: It occurs if the mathematical model is too simple or the training time is too short. To solve the underfitting problem, use a more complex model or extend the training time.

Model Effectiveness (2)

- **Model capacity**, also known as model complexity, is the capability of the model to fit various functions.
 - With sufficient capacity to handle task complexity and training data volumes, the algorithm results are optimal.
 - Models with an insufficient capacity cannot handle complex tasks because **underfitting** may occur.
 - Models with a large capacity can handle complex tasks, but **overfitting** may occur when the capacity is greater than the amount required by a task.

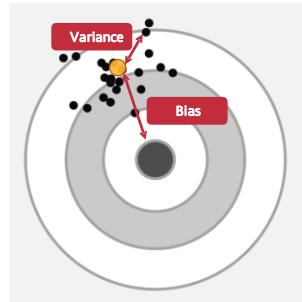


41 Huawei Confidential



Cause of Overfitting - Errors

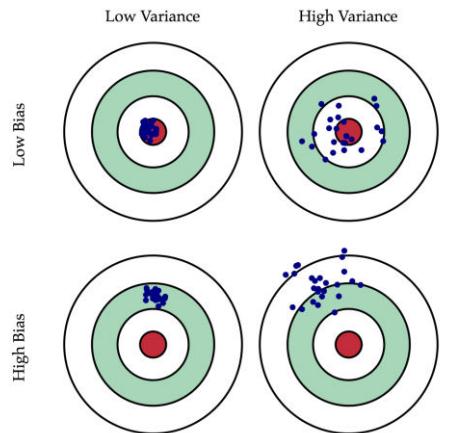
- Prediction error = **Bias²** + **Variance** + Ineliminable error
- In general, the two main factors of prediction error are variance and bias.
- Variance:
 - How much a prediction result deviates from the mean
 - Variance is caused by the sensitivity of the model to small fluctuations in a training set.
- Bias:
 - Difference between the average of the predicted values and the actual values.



- About the ineliminable error
 - Theoretically, if there is infinite amount of data and a perfect model, the error can be eliminated.
 - In practice, no model is perfect, and our data volume is limited.

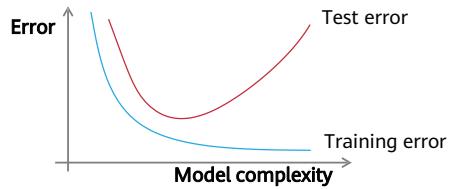
Variance and Bias

- Different combinations of variance and bias are as follows:
 - Low bias & low variance → **good model**
 - Low bias & high variance → **inadequate model**
 - High bias & low variance → **inadequate model**
 - High bias & high variance → **bad model**
- An ideal model can accurately capture the rules in the training data and be generalized to invisible (new) data. However, it is impossible for a model to complete both tasks at the same time.



Complexity and Errors of Models

- The more complex a model is, the smaller its training error is.
- As the model complexity increases, the test error decreases before increasing again, forming a convex curve.



Performance Evaluation of Machine Learning - Regression

- Mean absolute error (MAE). An MAE value closer to 0 indicates the model fits the training data better.

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

- Mean squared error (MSE).

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

- The value range of R^2 is [0,1]. A larger value indicates that the model fits the training data better. TSS indicates the difference between samples, and RSS indicates the difference between the predicted values and sample values.

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

Performance Evaluation of Machine Learning - Classification

(1)

- Terms:

- 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.
- TN : true negative, indicating the number of negative cases that are correctly classified.
- FP : false positive, indicating the number of positive cases that are incorrectly classified.
- FN : false negative, indicating the number of negative cases that are incorrectly classified.

- The confusion matrix is an $m \times m$ table at minimum. The entry $CM_{i,j}$ in the first m rows and m columns indicates the number of cases that belong to class i but are labeled as j .
 - For classifiers with high accuracy, most of the cases should be represented by entries on the diagonal of the confusion matrix from $CM_{1,1}$ to $CM_{m,m}$, while other entries are 0 or close to 0. That is, FP and FN are close to 0.

Predicted \ Actual		Yes	No	Total
Yes	TP	FN	P	
No	FP	TN	N	
Total	P'	N'	$P + N$	

Confusion matrix

Performance Evaluation of Machine Learning - Classification (2)

Measurement	Formula
Accuracy, recognition rate	$\frac{TP + TN}{P + N}$
Error rate, misclassification rate	$\frac{FP + FN}{P + N}$
True positive rate, sensitivity, recall	$\frac{TP}{P}$
True negative rate, specificity	$\frac{TN}{N}$
Precision	$\frac{TP}{TP + FP}$
F_1 value, harmonic mean of precision and recall	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
F_β value, where β is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

Performance Evaluation of Machine Learning - Example

- In this example, an ML model was trained to identify an image of a cat. To evaluate the model's performance, 200 images were used, of which 170 of them were cats.
- The model reported that 160 images were cats.

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

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

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

Predicted \ Actual		yes	no	Total
yes	140	30	170	
no	20	10	30	
Total	160	40	200	

Contents

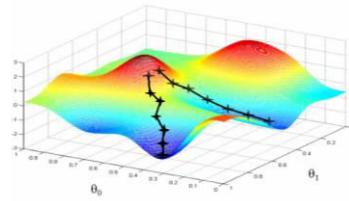
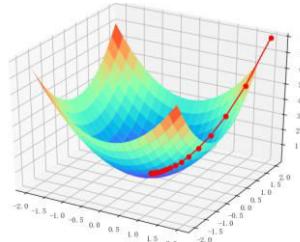
1. Machine Learning Algorithms
2. Types of Machine Learning
3. Machine Learning Process
- 4. Important Machine Learning Concepts**
5. Common Machine Learning Algorithms

Machine Learning Training Methods - Gradient Descent (1)

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

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

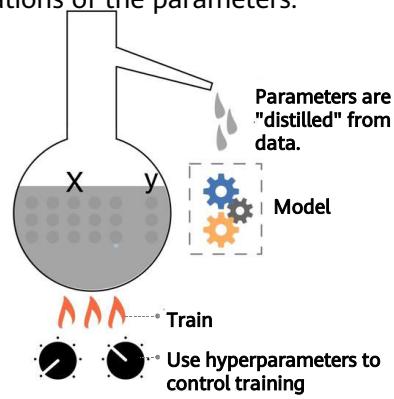
- η is the learning rate. i indicates the i -th data record. $\eta \nabla f_{w_k}(x^i)$ indicates the change of weight parameter w in each iteration.
- Convergence means that the value of the objective function changes very little or reaches the maximum number of iterations.



- Typically, the gradient descent method shows a decreasing descent in a very small range close to the target point, but the gradient may fluctuate in a specific range of the target point.

Parameters and Hyperparameters

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



Hyperparameters

- Commonly used for model parameter estimation.
- Specified by the user.
- Set heuristically.
- Often tuned for a given predictive modeling problem.

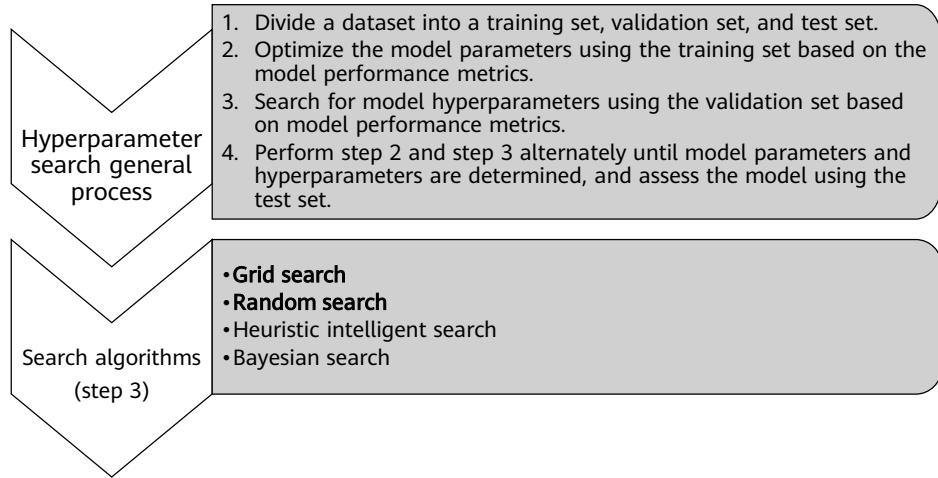
Hyperparameters are configurations outside the model.

- λ of Lasso/Ridge regression
- Learning rate, number of iterations, batch size, activation function, and number of neurons of a neural network to be trained
- C and σ of support vector machines (SVMs)
- k in the k -nearest neighbors (k -NN) algorithm
- Number of trees in a random forest

Common hyperparameters

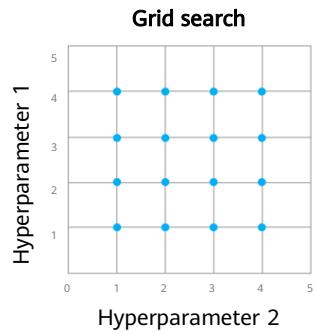
- Parameters are a part of a model that is learned from historical training data and key to machine learning algorithms. They have the following characteristics:
 - They are required by a model to make predictions.
 - Their values define the model functions.
 - They are learned or estimated from data.
 - They are often not set manually by the user.
 - They are often saved as a part of the learned model.
- Examples:
 - Weights in an artificial neural network
 - Support vectors in a SVM
 - Coefficients in linear regression or logistic regression

Hyperparameter Search Process and Methods



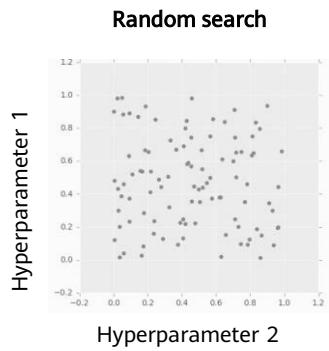
Hyperparameter Tuning Methods - Grid Search

- Grid search performs an **exhaustive search** of all possible hyperparameter combinations to form a hyperparameter value grid.
- In practice, the hyperparameter ranges and steps are manually specified.
- Grid search is expensive and time-consuming.
 - This method works well when there are relatively few hyperparameters. Therefore, it is feasible for general machine learning algorithms, but not for neural networks (see the deep learning course).



Hyperparameter Tuning Methods - Random Search

- If the hyperparameter search space is large, **random search** is more appropriate than grid search.
- In a random search, each setting item is sampled from possible parameter values to find the most appropriate parameter subset.
- Note:
 - In a random search, a search is first performed within a broad range, and then the range is narrowed based on the location of the best result.
 - Some hyperparameters are more important than others and affect random search preferences.



Cross-Validation (1)

- **Cross-validation** is a statistical analysis method used to check the performance of classifiers. It splits the original data into the training set and validation set. The former is used to train a classifier, whereas the latter is used to evaluate the classifier by testing the trained model.
- **k -fold cross-validation (k -fold CV):**
 - Divides the original data into k (usually equal-sized) subsets.
 - Each unique group is treated as a validation set, and the remaining $k - 1$ groups are treated as the training set. In this way, k models are obtained.
 - The average classification accuracy score of the k models on the validation set is used as the performance metric for k -fold CV classifiers.

- After a dataset is divided into a fixed training set and a fixed test set, if the test set error is small, there is a problem. A small test set implies statistical uncertainty around the estimated mean test error, making it difficult to claim that A algorithm works better than B algorithm on the given task. When the dataset has hundreds of thousands of samples or more, this is not a serious problem. When the dataset is too small, alternative procedures enable one to use all the samples in the estimation of the mean test error, at the price of increased computational workload.
- In k -fold CV, k is typically greater than or equal to 2. It usually starts from 3 and is set to 2 only when the original dataset is small. k -fold CV can effectively avoid over-learning and under-learning, and the final result is also persuasive.

Cross-Validation (2)

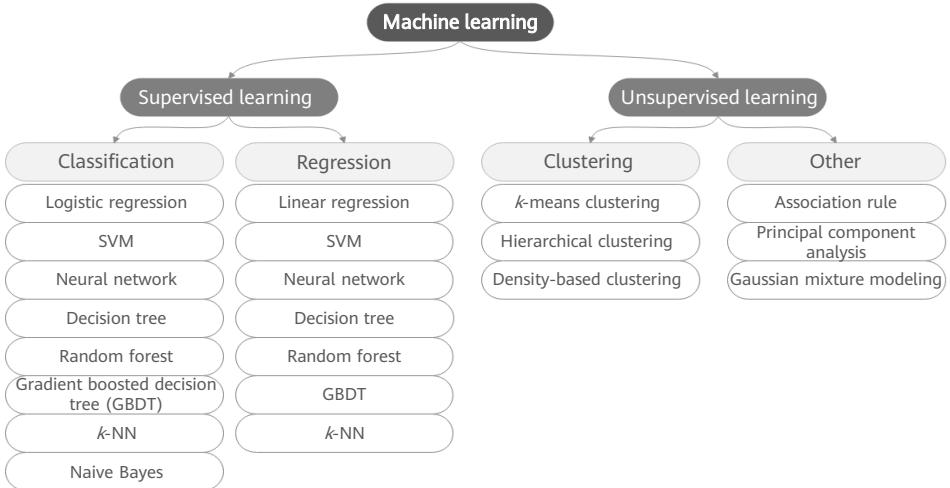


- Note: k in k -fold CV is a hyperparameter.

Contents

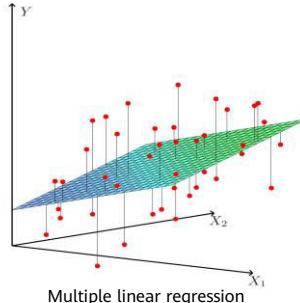
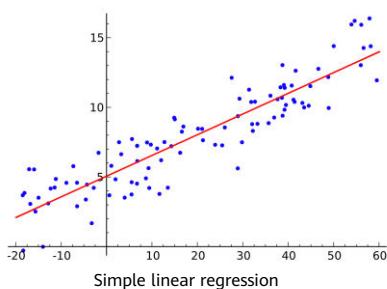
1. Machine Learning Algorithms
2. Types of Machine Learning
3. Machine Learning Process
4. Important Machine Learning Concepts
- 5. Common Machine Learning Algorithms**

Machine Learning Algorithm Overview



Linear Regression (1)

- Linear regression uses the regression analysis of mathematical statistics to determine the quantitative relationship between two or more variables.
- Linear regression is a type of supervised learning.



- Simple linear regression analysis refers to regression analysis where only one independent variable and one dependent variable exist, and their relationship can be approximately represented by a straight line. Multiple linear regression analysis involves two or more independent variables and the relationship between independent variables and dependent variables is linear. The linear regression model is a straight line only when the variable x is one-dimensional. It is a hyperplane when this variable is multi-dimensional. For example, the price of an apartment is determined by a variety of factors such as the area, layout, and location. Prediction of the apartment price based on these factors can be abstracted into a linear regression problem.
- The above figures show polynomial linear regression.

Linear Regression (2)

- The model function of linear regression is as follows, where w is the weight parameter, b is the bias, and x represents the sample:

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

- The relationship between the value predicted by the model and the actual value is as follows, where y indicates the actual value, and ε indicates the error:

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

- The error ε is affected by many independent factors. Linear regression assumes that the error ε follows normal distribution. The loss function of linear regression can be obtained using the normal distribution function and maximum likelihood estimation (MLE):

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

- We want the predicted value approaches the actual value as far as possible, that is, to minimize the loss value. We can use a gradient descent algorithm to calculate the weight parameter w when the loss function reaches the minimum, thereby complete model building.

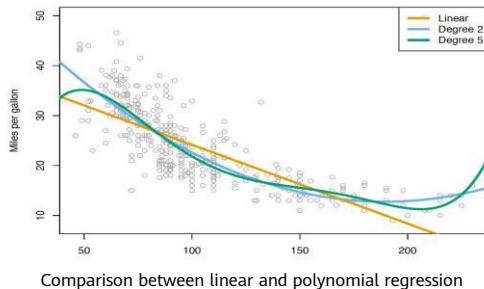
Linear Regression Extension - Polynomial Regression

- Polynomial regression is an extension of linear regression. Because the complexity of a dataset exceeds the possibility of fitting performed using a straight line (obvious underfitting occurs if the original linear regression model is used), polynomial regression is used.

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

Here, n -th power indicates the degree of the polynomial.

Polynomial regression is a type of linear regression. Although its features are non-linear, the relationship between its weight parameters w is still linear.



Preventing Overfitting of Linear Regression

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

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

- Regularization term: This regularization term is called L2-norm. Linear regression that uses this loss function is called **Ridge regression**.

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

- Linear regression with an absolute loss is called **Lasso regression**.

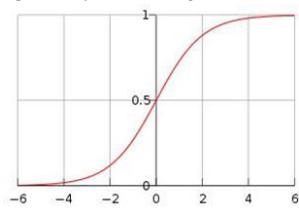
Logistic Regression (1)

- The logistic regression model is a classification model used to resolve classification problems. The model is defined as follows:

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

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

w represents the weight, b represents the bias, and $wx + b$ represents a linear function with respect to x . Compare the preceding two probability values. x belongs to the type with a larger probability value.



Logistic Regression (2)

- Logistic regression and linear regression are both linear models in broad sense. The former introduces a non-linear factor (sigmoid function) on the basis of the latter and sets a threshold. Therefore, logistic regression applies to binary classification.
- According to the model function of logistic regression, the loss function of logistic regression can be calculated through maximum likelihood estimation as follows:

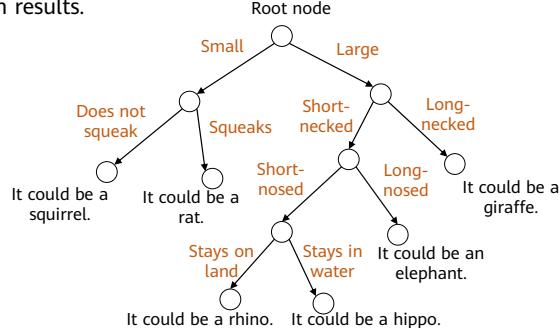
$$J(w) = -\frac{1}{m} \sum (y \ln h_w(x) + (1-y) \ln(1-h_w(x)))$$

- In the formula, w indicates the weight parameter, m indicates the number of samples, x indicates the sample, and y indicates the actual value. You can also obtain the values of all the weight parameters w by using a gradient descent algorithm.

- Overfitting of logistic regression can also be prevented by adding regularization terms.

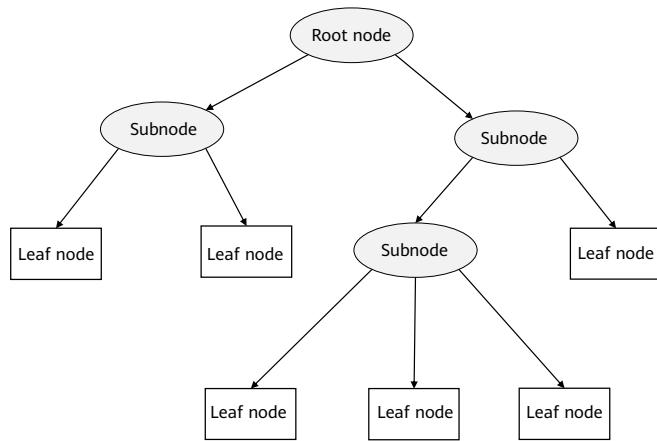
Decision Tree

- Each non-leaf node of the decision tree denotes a test on an attribute; each branch represents the output of a test; and each leaf (or terminal) node holds a class label. The algorithm starts at the root node (topmost node in the tree), tests the selected attributes on the intermediate (internal) nodes, and generates branches according to the output of the tests. Then, it saves the class labels on the leaf nodes as the decision results.



- How to construct a decision tree is very important. We should determine the topological structure of feature attributes based on attribute selection measures. The key step is to split attributes. That is, different branches are constructed based on the differences of a feature attribute on a node.
- The common learning algorithms used to generate a decision tree include ID3, C4.5, and CART.

Structure of a Decision Tree



Key to Decision Tree Construction

- A decision tree requires feature attributes and an appropriate tree structure. 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.
- Purity is measured mainly through the information entropy and GINI coefficient. The formula is as follows:

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

$$\min_{j,s} [\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

- p_k indicates the probability that a sample belongs to category k (in a total of K categories). A larger purity difference between the sample before and after division indicates a better decision tree.
- Common decision tree algorithms include ID3, C4.5, and CART.

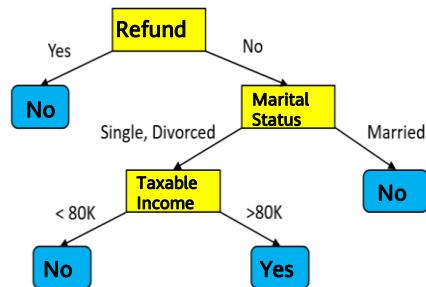
Decision Tree Construction Process

- **Feature selection:** Select one of the features of the training data as the split standard of the current node. (Different standards distinguish different decision tree algorithms.)
- **Decision tree generation:** Generate subnodes from top down based on the selected feature 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

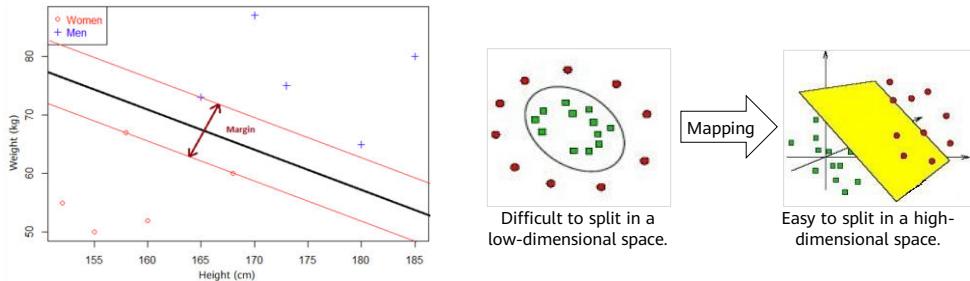
- The following figure shows a decision tree for a classification problem. The classification result is affected by three attributes: refund, marital status, and taxable income.

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



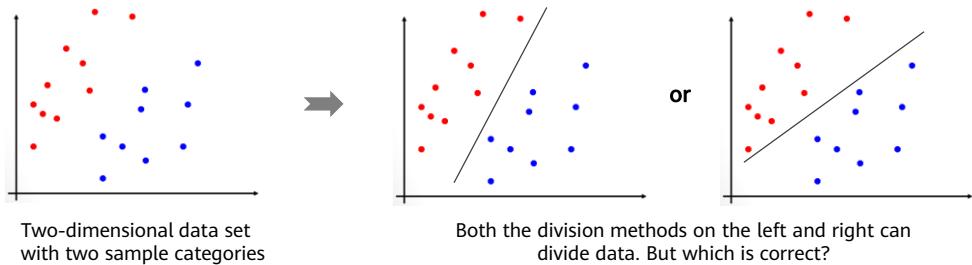
Support Vector Machine

- Support vector machines (SVMs) are binary classification models. Their basic model is the linear classifier that maximizes the width of the gap between the two categories in the feature space. SVMs also have a kernel trick, which makes it a non-linear classifier. The learning algorithm of SVMs is the optimal algorithm for convex quadratic programming.



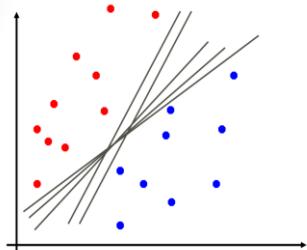
Linear SVM (1)

- How can we divide the red and blue data points with just one line?

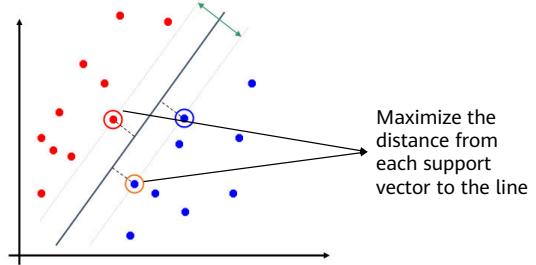


Linear SVM (2)

- We can use different straight lines to divide data into different categories. SVMs find a straight line and keep the most nearby points as **far** from the line as possible. This gives the model a strong generalization capability. These most nearby points are called **support vectors**.
- In the two-dimensional space, a straight line is used for division; in the high-dimensional space, a **hyperplane** is used for division.



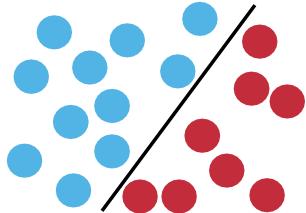
78 Huawei Confidential



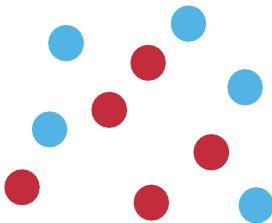
 HUAWEI

Non-linear SVM (1)

- How can we divide a linear inseparable data set?



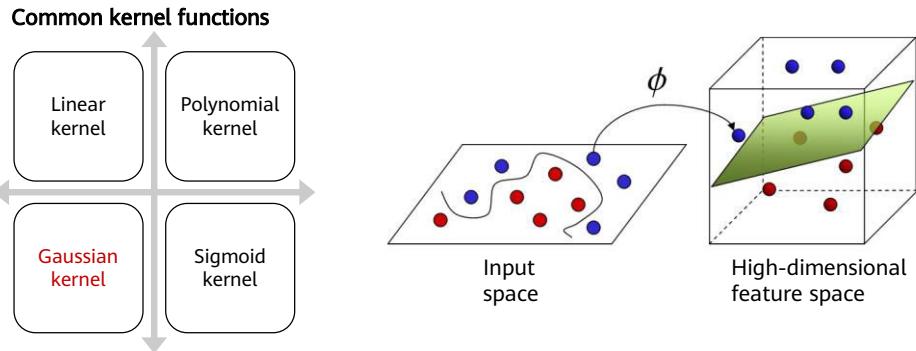
Linear SVM works well on a linear separable data set.



A non-linear data set cannot be divided using a straight line.

Non-linear SVM (2)

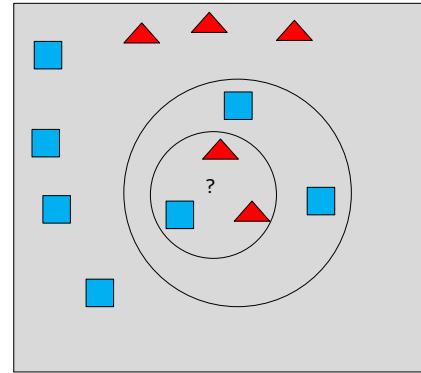
- Kernel functions can be used to create non-linear SVMs.
- Kernel functions allow algorithms to fit a maximum-margin hyperplane in a transformed high-dimensional feature space.



- The Gaussian kernel function is the most frequently used.

k -Nearest Neighbors Algorithm (1)

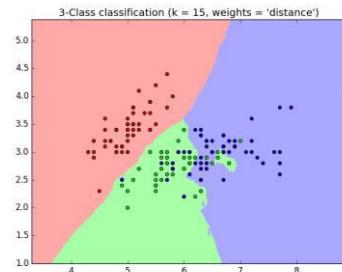
- The k -nearest neighbor (k -NN) classification algorithm is a theoretically mature method and one of the simplest machine learning algorithms. The idea of k -NN classification is that, if most of k closest samples (nearest neighbors) of a sample in the feature space belong to a category, the sample also belongs to this category.



The category of point ? varies according to how many neighbor nodes are chosen.

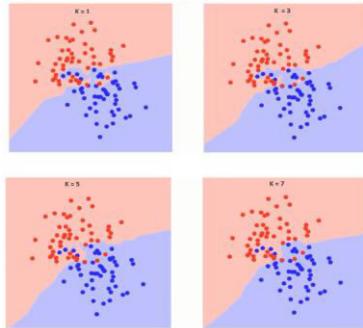
k -Nearest Neighbors Algorithm (2)

- The logic of k -NN is simple: If an object's k nearest neighbors belong to a class, so does the object.
- k -NN is a non-parametric method and is often used for datasets with irregular decision boundaries.
 - k -NN typically uses the **majority voting** method to predict classification, and uses the **mean value** method to predict regression.
- k -NN requires a very large amount of computing.



k -Nearest Neighbors Algorithm (3)

- Typically, a larger k value reduces the impact of noise on classification, but makes the boundary between classes less obvious.
 - A large k value indicates a higher probability of underfitting because the division is too rough; while a small k value indicates a higher probability of overfitting because the division is too refined.



- As seen from the figure, the boundary becomes smoother as the k value increases.
- As the k value increases, the points will eventually become all blue or all red.

Naive Bayes (1)

- **Naive Bayes** classifiers are a family of simple "probabilistic classifiers" based on **Bayes' theorem** with **strong independence assumptions between the features**. For a given sample feature X , the probability that the sample belongs to 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, X_2, \dots, X_n are data features, which are usually described by m measurement values of the attribute set.
 - For example, the attribute of the color feature may be red, yellow, and blue.
- C_k indicates that the data belongs to a specific class C .
- $P(C_k | X_1, X_2, \dots, X_n)$ is the posterior probability, or the posterior probability of H under condition C_k .
- $P(C_k)$ is the prior probability independent of X_1, X_2, \dots, X_n .
- $P(X_1, X_2, \dots, X_n)$ is the prior probability of X .

- Class conditional independence: The Bayes classifier assumes that the effect of an attribute value on a given class is independent of the values of other attributes. This assumption is made to simplify the calculation and becomes "naive" in this sense.
- Bayes classifiers show high accuracy and speed when applied to large databases.

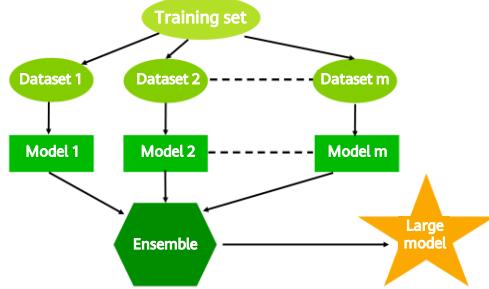
Naive Bayes (2)

- Feature independent hypothesis example:
 - If a fruit is red, round, and about 10 cm in diameter, it can be considered an apple.
 - A Naive Bayes classifier believes that each of these features independently contributes to the probability of the fruit being an apple, regardless of any possible correlation between color, roundness, and diameter features.

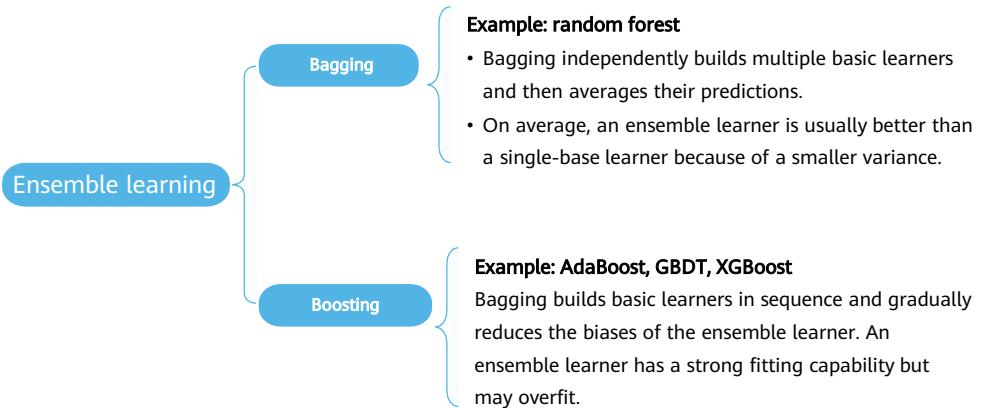


Ensemble Learning

- Ensemble learning is a machine learning paradigm in which multiple learners are trained and combined to resolve an issue. When multiple learners are used, the generalization capability of the ensemble can be much stronger than that of a single learner.
- For example, If you ask thousands of people at random a complex question and then summarize their answers, the summarized answer is more accurate than an expert's answer in most cases. This is the **wisdom of the crowd**.

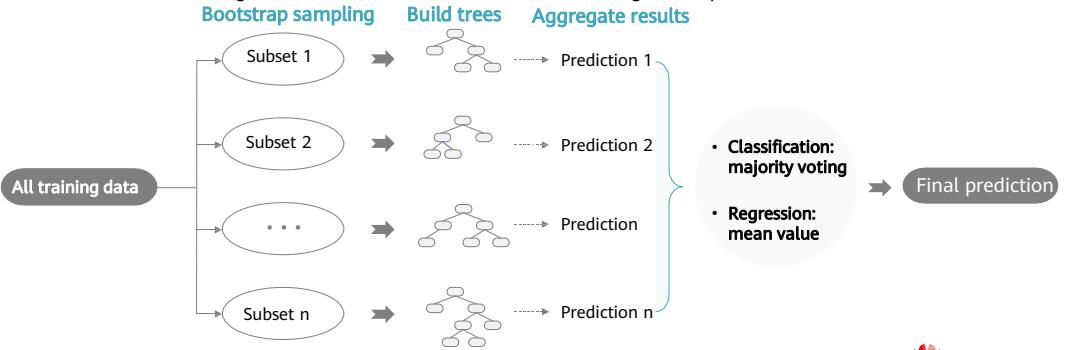


Types of Ensemble Learning



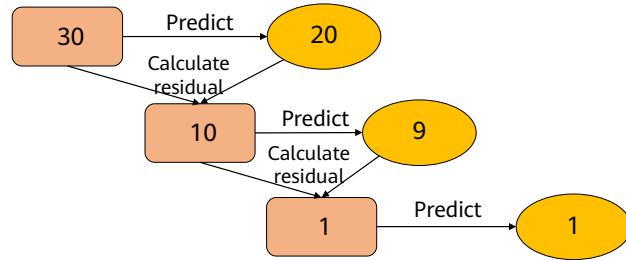
Ensemble Learning - Random Forest

- Random forest = Bagging + Classification and regression tree (CART)
- Random forest builds multiple decision trees and aggregates their results to make prediction more accurate and stable.
 - The random forest algorithm can be used for classification and regression problems.



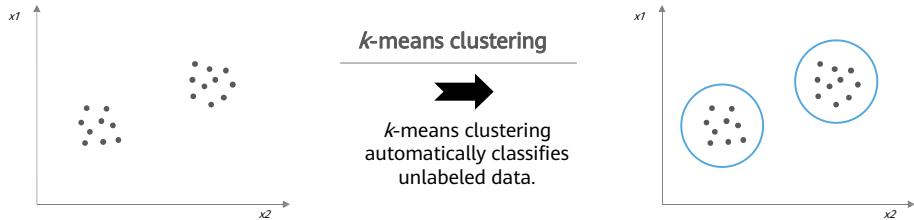
Ensemble learning - Gradient Boosted Decision Tree

- Gradient boosted decision tree (GBDT) is a type of boosting algorithm.
- The prediction result of the ensemble model is the sum of results of all base learners. The essence of GBDT is that the next base learner tries to fit the residual of the error function to the prediction value, that is, the residual is the error between the prediction value and the actual value.
- During GBDT model training, the loss function value of the sample predicted by the model must be as small as possible.



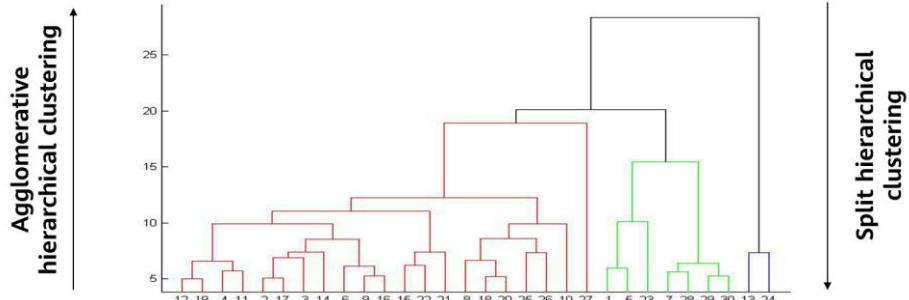
Unsupervised Learning - k -Means Clustering

- k -means clustering takes the number of clusters k and a dataset of n objects as inputs, and outputs k clusters with minimized within-cluster variances.
- In the k -means algorithm, the number of clusters is k , and n data objects are split into k clusters. The obtained clusters meet the following requirements: high similarity between objects in the same cluster, and low similarity between objects in different clusters.



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 diagram. The root is the only cluster of all samples, and the leaves are clusters of single samples.



Summary

- This course first describes the definition and types of machine learning, as well as problems machine learning solves. Then, it introduces key knowledge points of machine learning, including the overall procedure (data preparation, data cleansing, feature selection, model evaluation, and model deployment), common algorithms (including linear regression, logistic regression, decision tree, SVM, Naive Bayes, k -NN, ensemble learning, and k-means clustering), and hyperparameters.

Quiz

1. (Single-answer) Which of the following is not a supervised learning algorithm?
()
 - A. Linear regression
 - B. Decision tree
 - C. k -NN
 - D. k -means clustering
2. (True or false) Gradient descent is the only method of machine learning. ()

- Answers: 1. D 2. False

Recommendations

- Huawei Talent
 - <https://e.huawei.com/en/talent/portal/#/>
- 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。

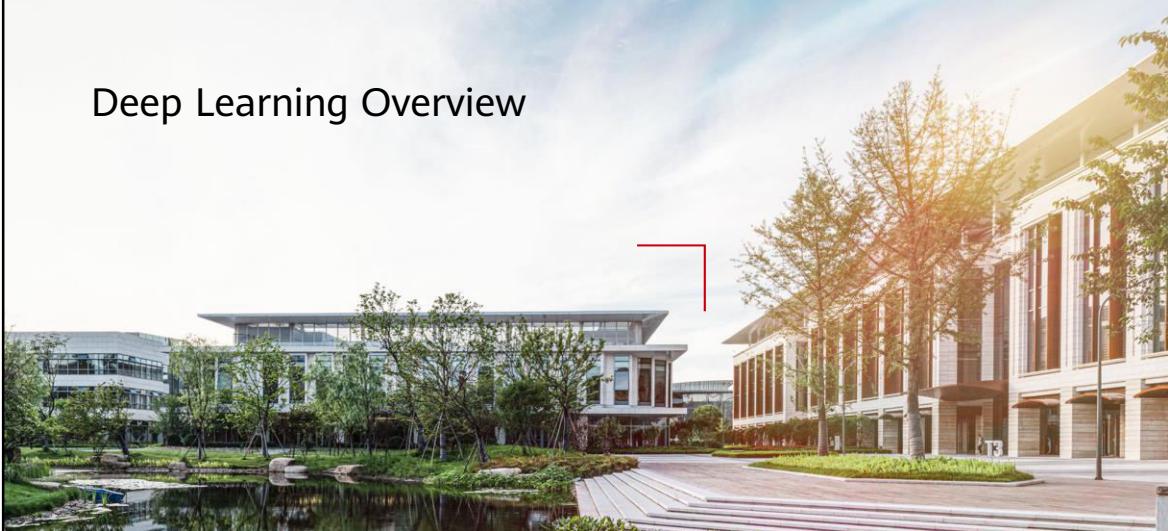
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2023 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

- This chapter describes the basic knowledge related to deep learning, including the development history, components and types of neural networks for deep learning, and common problems in deep learning engineering.

Objectives

- Upon completion of this course, you will be able to:
 - Describe the definition and development of neural networks.
 - Be familiar with the components of neural networks for deep learning.
 - Be familiar with the training and optimization of neural networks.
 - Describe common problems in deep learning.

Contents

- 1. Deep Learning**
2. Training Rules
3. Activation Functions
4. Normalization
5. Optimizers
6. Neural Network Types

Traditional Machine Learning vs. Deep Learning

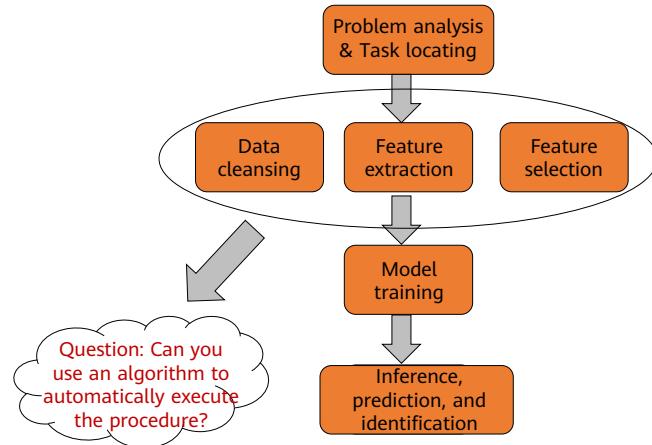
- Deep learning is a learning model based on unsupervised feature learning and the feature hierarchical structure. It has great advantages in computer vision, speech recognition, and natural language processing (NLP).

Traditional Machine Learning	Deep Learning
It has low requirements for hardware because the amount of computation is limited, and GPUs are not required for parallel computing.	It has certain hardware requirements because numerous matrix operations are needed, and also requires GPUs for parallel computing.
It is suitable for training with a small amount of data, but its performance cannot improve as the amount increases.	It achieves high performance when high-dimensional weight parameters and massive training data are provided.
Problems are located level by level.	It is an end-to-end learning method.
Features are manually selected.	Algorithms are used to automatically extract features.
Feature interpretability is strong.	Feature interpretability is weak.

Deep learning is a learning model based on unsupervised feature learning and the feature hierarchical structure. It has great advantages in computer vision, speech recognition, and natural language processing (NLP).

- The two kinds of learning are compared from five aspects.

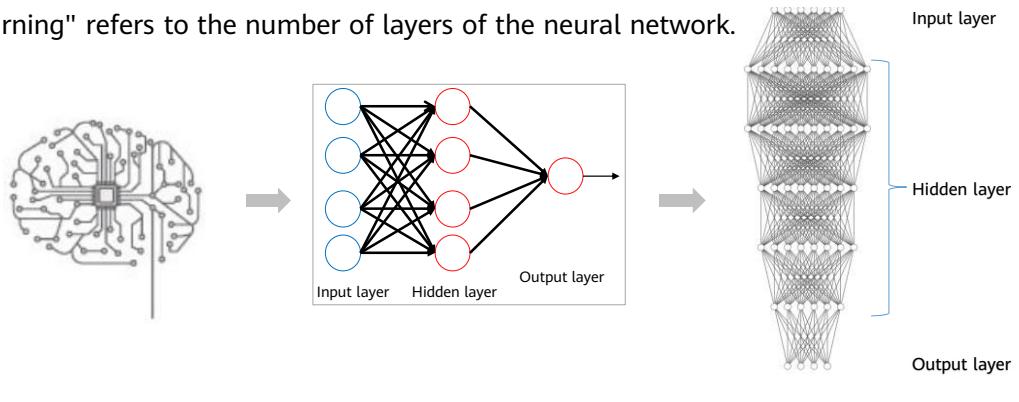
Traditional Machine Learning



- Problem analyzing and task locating
- Data cleansing
- Feature extraction
- Feature selection
- Model training
- Inference, prediction, and identification
- Generally, features in machine learning are selected manually. More features indicate more information and higher identification accuracy.
- However, more features indicate higher computation complexity and larger search space. Training data appears to be sparse in the overall feature vectors, affecting similarity determination, that is, **dimension exploration**.
- In addition, features that have nothing to do with classification may interfere with learning.
- Conclusion: More features do not mean a better learning effect. Proper feature selection is the key to successful identification. The number of required features can be determined by the **problem**.

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.



7 Huawei Confidential



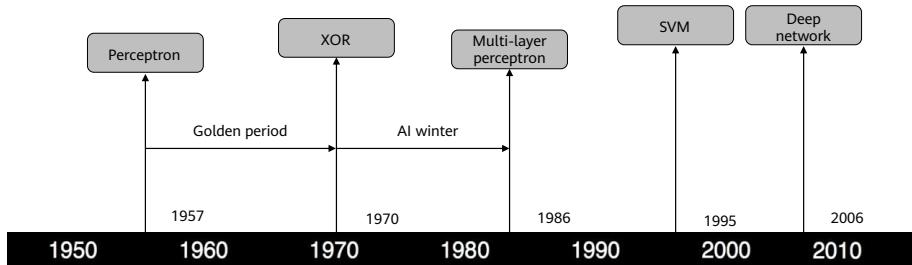
- What is deep learning?
- Generally, the deep learning architecture is a deep neural network. "Deep" in "deep learning" refers to the number of layers of the neural network.
- The network is built by simulating the human neural network.
- In the design and application of artificial neural networks (ANNs), the following factors need to be considered: neuron functions, connection modes among neurons, and network learning (training).
- Image source: <http://3ms.huawei.com/km/static/image/detail.html?fid=59876>

Neural Network

- Currently, definitions of the neural network are not unified. According to Hecht Nielsen, an American neural network scientist, a neural network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.
- Based on the source, features, and explanations of the neural network, **the artificial neural network (ANN) can be simply expressed as an information processing system designed to imitate the human brain structure and functions.**
- **ANN** is a network formed by artificial neurons connected to each other. It extracts and simplifies the human brain's microstructure and functions, and is an important approach to simulating human intelligence. It reflects several basic features of human brain functions, such as concurrent information processing, learning, association, model classification, and memory.

- Let's see what a neural network is.

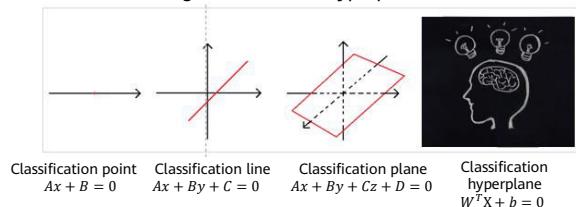
Development History of Neural Networks



- Development history of neural networks (milestones of deep learning):
- Budding neural network (1958–1969)
 - In 1958, Frank Rosenblatt invented the perceptron algorithm.
 - In 1969, Marvin Minsky, a pioneer of artificial intelligence in the US, questioned why perceptron could handle only linear classification problems but failed to classify even the simplest XOR problems. The research on the perceptron was doomed to failure.
- Developing neural network (1986–1998)
 - The second-generation neural network: In 1986, G. E. Hinton, a deep learning expert, developed a BP algorithm suitable for multilayer perceptron (MLP) and used Sigmoid for non-linear mapping, which solved the problem of non-linear classification and learning.
 - Universal approximation theorem: In 1989, Robert Hecht-Nielsen proved that a continuous function f in any closed interval could be approximated by a BP network containing a hidden layer.
- Emerging neural network (since 2006)
 - Year 2006 was the first year for deep learning. In 2006, Hinton proposed a solution to the vanishing gradient problem in the deep neural network training: weight initialization for unsupervised pre-training+weight adjustment for supervised training.
 - In 2012, the convolutional neural network (CNN) of Hinton's team overwhelmed other methods and won the first prize in the top image recognition competition ImageNet, setting off an upsurge in deep learning.
 - In 2016, the deep learning AI program AlphaGo developed by Google beat the Go world champion Lee Sedol who is a player of 9 dan rank, further promoting the popularity of deep learning.

Single-layer Perceptron

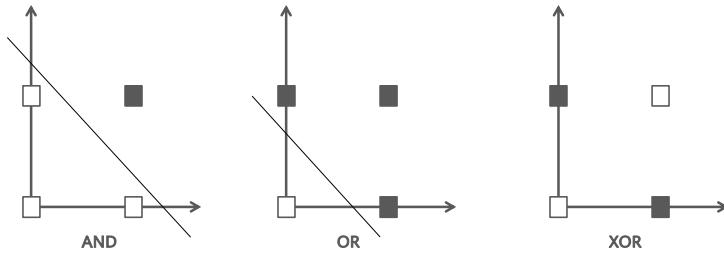
- **Input vector:** $X = [x_0, x_1, \dots, x_n]^T$.
- **Weight:** $W = [\omega_0, \omega_1, \dots, \omega_n]^T$, where ω_0 indicates an offset.
- **Activation function:** $O = sign(net) = \begin{cases} 1, & net > 0, \\ -1, & otherwise. \end{cases}$
- The preceding perceptron is equivalent to a classifier. 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.



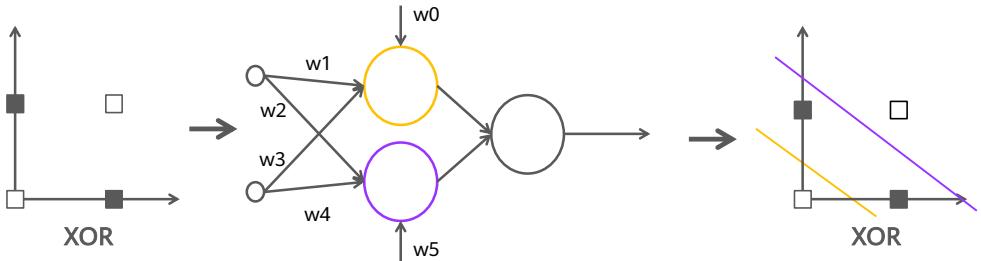
- First, let's take a look at the simplest neural network — single-layer perceptron.
- In 1958, Frank Rosenblatt invented the perceptron algorithm. Such an algorithm coexisted with machine learning for more than a decade.
- In 1969, Marvin Minsky, a pioneer of artificial intelligence in the US, questioned why perceptron could handle only linear classification problems but failed to classify even the simplest XOR problems. The research on the perceptron was doomed to failure.
- Image source: <http://3ms.huawei.com/km/static/image/detail.html?fid=63878>

XOR Problem

- In 1969, Marvin Minsky, an American mathematician and AI pioneer, proved that perceptrons are essentially a type of linear model capable of processing only linear classification.

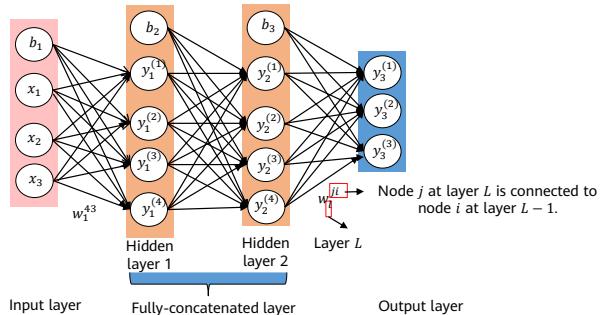


Solving the XOR Problem



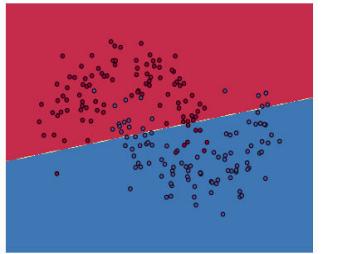
Feedforward Neural Network (FNN)

- An FNN is a typical deep learning model. It has the following features:
 - Input nodes do not provide the computing function and are used to represent only the element values of an input vector.
 - Each neuron is connected only to a neuron at the previous layer, receives an output of the previous layer as its input, and outputs the computing result to the next layer. A unidirectional multi-layer structure is used. There is no feedback in the entire network. Signals are transmitted unidirectionally from the input layer to the output layer. The network can be represented by a directed acyclic graph (DAG).

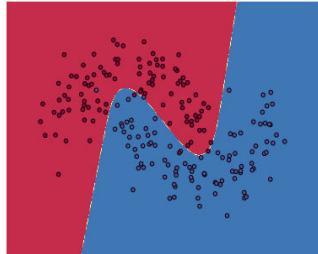


- **A feedforward neural network is a multi-layer perceptron.**
- It is the simplest neural network with neurons arranged in layers. It is more widely used and is developing faster than other ANNs.
- Input nodes do not compute but represent only the element values of an input vector.
- Nodes at each layer represent neurons that provide the computing function. Each neuron is connected only to the neurons of the previous layer.
- A unidirectional multi-layer structure is used to receive the output of the previous layer and send the output to the next layer. **Each layer includes several neurons, and the neurons at the same layer are not connected to each other.** Only unidirectional inter-layer information transmission is supported.

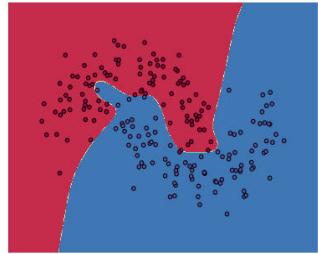
Influence of Hidden Layers on Neural Networks



0 hidden layers



3 hidden layers



20 hidden layers

Contents

1. Deep Learning
- 2. Training Rules**
3. Activation Functions
4. Normalization
5. Optimizers
6. Neural Network Types

Common Loss Functions in Deep Learning

- When training a deep learning network, we need to parameterize the error of target classification. A **loss function (error function)** is used, which reflects the error between the target output and the actual output of the perceptron.
 - For **regression tasks**, the commonly used loss function is the **quadratic cost function**. For a separate training sample x , the quadratic cost function can be written as follows:

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

d indicates a neuron at the output layer, D indicates all neurons at the output layer, t_d indicates the target output, and o_d indicates the actual output.

- For **classification tasks**, the commonly used loss function is the **cross-entropy cost function**.

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

The cross-entropy cost function depicts the distance between two probability distributions, which is a widely used loss function for classification problems.

- Generally, the quadratic cost function is used for regression problems, and the cross-entropy cost function is used for classification problems.

- This loss function has the following features:
 - It uses the weight vector as the independent variable.
 - It uses the sum of squares of the difference between the target output t_d and actual output o_d of each training sample as the main body.
 - There is a coefficient $1/2$.
- Once the training sample is given, the input and target output are constants. The actual output varies depending on W . The independent variable of the error function is W .
- Usage of the coefficient $1/2$, which is difficult to understand, is as follows: When you take a derivative of E with respect to the independent variable, the coefficient $1/2$ is multiplied by the index 2 and the number 1 is obtained. The specific computation will be described in the following text.
- X indicates the sample, n indicates the number of samples, and d indicates the neuron at the output layer.

Extremum of a Loss Function

- **Purpose:** A loss function can iteratively search for and update parameter w in the negative gradient direction to minimize the loss function.
- **Limitation:** There is no effective method for solving the extremum in mathematics on the complex high-dimensional surface of $C(W) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$.
- **Approach:** The core approach of the gradient descent method is as follows: The negative gradient direction is the fastest descent direction of the function. Therefore, the minimum point of $C(W)$ is expected to exist along the $-C()$ direction.

- **Approach:** The core approach of the gradient descent method is as follows: The negative gradient direction is the fastest descent direction of the function. Therefore, the minimum point of $C(W)$ is expected to exist along the $-C()$ direction.

Gradient Descent and Loss Function

- The gradient of a multivariable function $C(W) = f(w_0, w_1, \dots, w_n)$ at $W' = [w_0', w_1', \dots, w_n']^T$ is as follows:

$$\nabla f(w_0', w_1', \dots, w_n') = [\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_n}]^T|_{W=W'},$$

Direction of the gradient vector points to the direction in which the function grows fastest. Therefore, the direction of the negative gradient vector $-\nabla C$ points to the direction in which the function decreases fastest.

- The learning rate (LR) is a coefficient for adjusting weights according to an error gradient, and is usually denoted as η .

$$W_{t+1} = W_t - \eta \frac{dC}{dW}$$

Based on the learning rate and gradient value, updating all parameter values will reduce the network loss.

- This loss function has the following features:
 - It uses the weight vector as the independent variable.
 - It uses the sum of squares of the difference between the target output t_d and actual output o_d of each training sample as the main body.
 - There is a coefficient $1/2$.
- Once the training sample is given, the input and target output are constants. The actual output varies depending on W . The independent variable of the error function is W .
- Usage of the coefficient $1/2$, which is difficult to understand, is as follows: When you take a derivative of E with respect to the independent variable, the coefficient $1/2$ is multiplied by the index 2 and the number 1 is obtained. The specific computation will be described in the following text.

Batch Gradient Descent (BGD) Algorithm

- $\langle X, t \rangle$ indicates a sample in the training set. X is an input value vector, t is a target output, o is an actual output, η is a learning rate, and C is a loss function.
 - Initialize each w_i to a random value with a smaller absolute value.
 - Before the termination condition is met, do as follows:
 - Initialize each Δw_i to zero.
 - For each $\langle X, t \rangle$ in the training set, do as follows:
 - Input X to this unit and compute the output o .
 - For each w_i in this unit: $\Delta w_i += -\frac{1}{n} \sum_X \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$
 - For each w_i in this unit: $w_i += \Delta w_i$
- The gradient descent algorithm of this version is not commonly used because it has the following defect:
 - The convergence process is very slow because all training samples need to be computed every time the weight is updated.

Stochastic Gradient Descent (SGD) Algorithm

- To address the defect of the BGD algorithm, a common variant is developed, which is called incremental gradient descent or stochastic gradient descent. 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

- Initialize each w_i to a random value with a smaller absolute value.
- Before the termination condition is met, do as follows:
 - Randomly select $\langle X, t \rangle$ in the training set:
 - Input X to this unit and compute the output o .
 - For each w_i in this unit: $w_i += -\eta \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$

- This gradient descent algorithm goes to another extreme, that is, updating the weight based on each sample. Most training samples contain noises. As a result, when the extrema are approximated to, the gradient direction is oriented up and down near the extrema but difficult to converge to the extrema.
- ONLINE-GRADIENT-DESCENT
- Online learning
- This gradient descent algorithm goes to another extreme, that is, updating the weight based on each sample. Most training samples contain noises. As a result, when the extrema are approximated to, the gradient direction is oriented up and down near the extrema but difficult to converge to the extrema.

Mini-batch Gradient Descent (MBGD) Algorithm

- To address the defects of the previous two gradient descent algorithms, the MBGD algorithm was proposed and has been most widely used. It uses a small batch of samples with a fixed batch size to compute Δw_i and update weights.
- BATCH-GRADIENT-DESCENT
 - Initialize each w_i to a random value with a smaller absolute value.
 - Before the termination condition is met, do as follows:
 - Initialize each Δw_i to zero.
 - For each $\langle X, t \rangle$ in the samples obtained from the training set, do as follows:
 - Input X to this unit and compute 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: $w_i += \Delta w_i$
 - If it is the last batch, shuffle the training samples.

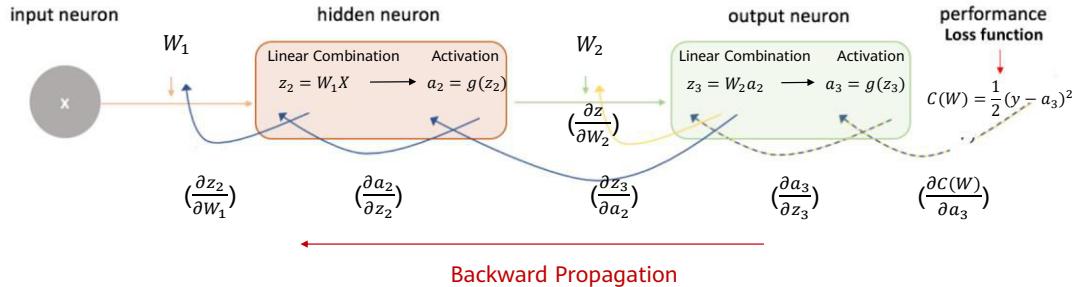
- This gradient descent algorithm considers both the efficiency and gradient stability. It is easy to overshoot the local minimum and is the most commonly used gradient descent algorithm in actual work. The batch size varies according to specific problems. Generally, the value is **32**.

Network Training Process

- Forward propagation: $C(W) = \frac{1}{2}(y - a_3)^2 = \frac{1}{2}(y - (g(W_2 g(W_1 X)))^2)$.
- If the parameter W_1 needs to be updated, according to $W_{t+1} = W_t - \eta \frac{dC}{dW}$, compute:

$$\frac{dC}{dW_1} = \frac{\partial C(W)}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial W_1}.$$

Forward Propagation



Backward Propagation

- Error backward propagation is an important algorithm in neural networks. It uses the **chain rule** to send the errors of the output layer back to the network, so that a gradient with respect to the weights of the neural network can be easily computed. The steps are as follows:
 - Backward propagate the loss function values to each compute unit.
 - Each compute unit updates the weight according to the obtained errors.

- If we use the gradient descent algorithm to optimize the model by directly using the final loss function value, that is, adjust the parameters to minimize the target function: The parameters are nested layer by layer. Therefore, the process of computing the gradient is complex and the computation workload is heavy. In this case, we use the backward propagation algorithm, which provides a method for computing the cost function gradient. By using this method, the forward errors are backward propagated to each neuron, and parameters are updated using gradient descent.
- Backward propagation process: Errors are transmitted to each compute unit (neurons at the hidden layer) based on the weight ratio to determine the responsibility of each neuron for the final error. Then, the weight is updated according to the allocated error.
- Now, let's look at the computation process of backward propagation.

Vanishing and Exploding Gradients (1)

- Vanishing gradient: As network layers increase, the derivative value of backward propagation decreases and the gradient vanishes.
- Exploding gradient: As network layers increase, the derivative value of backward propagation increases and the gradient explodes.
- Cause:

$$y_i = \sigma(z_i) = \sigma(w_i x_i + b_i), \text{ where } \sigma \text{ is a sigmoid function.}$$



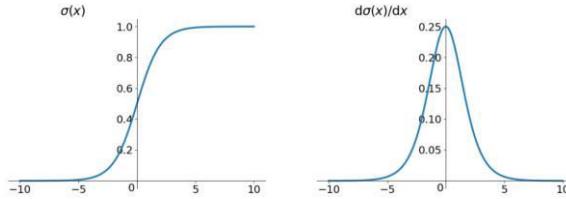
- Backward propagation can be deduced as follows:

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

- (It is assumed that each layer has only one neuron and for each layer, the activation function is a sigmoid function.)

Vanishing and Exploding Gradients (2)

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



- The network weight $|w|$ is usually less than 1. Therefore, $|\sigma'(z)w| \leq \frac{1}{4}$. When using the chain rule, the value of $\frac{\partial C}{\partial b_i}$ becomes smaller when the number of layers increases, resulting in the vanishing gradient problem.
- When the network weight $|w|$ is large, that is $|\sigma'(z)w| > 1$, the exploding gradient problem occurs.
- Solutions: The ReLU activation function and LSTM neural network are used to solve the vanishing gradient problem, and gradient clipping is used to solve the exploding gradient problem.

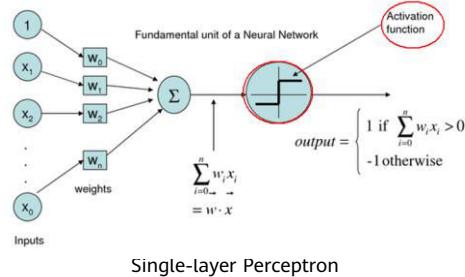
- This problem occurs when the value of w is large. However, this problem seldom occurs when the sigmoid activation function is used. It is because the value of $\sigma'(z)$ is also related to w ([equation]), unless the input value [equation] of this layer is within a small range.
- In fact, the exploding and vanishing gradient problems are caused by the deep network and unstable network weight update. In essence, they are caused by the chain rule in gradient backward propagation.
- Gradient clipping** is proposed for the exploding gradient problem. A gradient clipping threshold is set. When being updated, if the gradient exceeds the threshold, the gradient is forcibly limited within the range. This prevents the exploding gradient problem.
- Another approach for resolving the exploding gradient problem is weight normalization, among which l1 normalization and l2 normalization are commonly used. APIs in each deep learning framework may use normalization.
- ReLU:** Its idea is simple. If the derivative of the activation function is 1, the vanishing gradient or exploding gradient problem will not occur. That's why the ReLU is developed.
- For LSTM, the vanishing gradient problem does not easily occur thanks to the complex gates inside the LSTM. As shown in the following figure, LSTM can "remember" the "residual memory" of the previous training based on the internal gates when it is being updated. Therefore, it is often used in generative text. Currently, there is also CNN-based LSTM. Try it if you are interested.

Contents

1. Deep Learning
2. Training Rules
- 3. Activation Functions**
4. Normalization
5. Optimizers
6. Neural Network Types

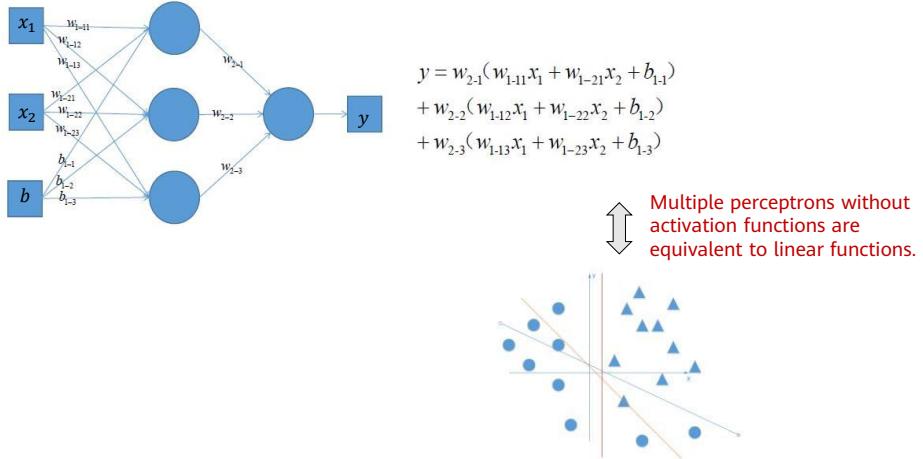
Concept of Activation Functions

- In a neural network, each neuron receives the outputs of neurons at the previous layer as its inputs and then transmits the inputs to the next layer. Neurons at the input layer directly transmit input attribute values to the next layer (a hidden or output layer). In a multi-layer neural network, there is a function between outputs of nodes at the previous layer and inputs of nodes at the next layer. Such function is referred to as an **activation function**.



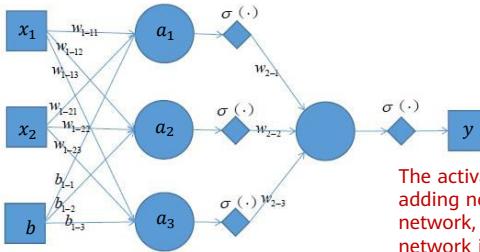
Single-layer Perceptron

Purpose of Activation Functions (1)



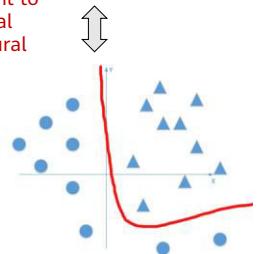
- The complexity of linear functions is limited, and the capability of learning complex function mappings from data is low.
- Reference: <https://juejin.im/entry/5ca186c4e51d4534624698ae>

Purpose of Activation Functions (2)



$$\begin{aligned}a1 &= w_{1-11}x_1 + w_{1-21}x_2 + b_{1-1} \\a2 &= w_{1-12}x_1 + w_{1-22}x_2 + b_{1-2} \\a3 &= w_{1-13}x_1 + w_{1-23}x_2 + b_{1-3} \\y &= \sigma(w_{2-1}\sigma(a1) + w_{2-2}\sigma(a2) + w_{2-3}\sigma(a3))\end{aligned}$$

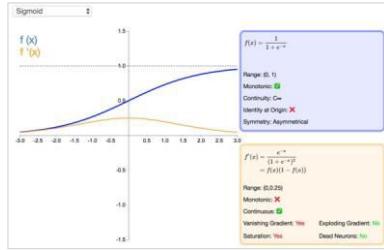
The activation function is equivalent to adding nonlinear factors to a neural network, which transforms the neural network into a nonlinear model.



- The activation function is equivalent to adding **nonlinear factors** to a neural network, which transforms the neural network into a nonlinear model.

Sigmoid Function

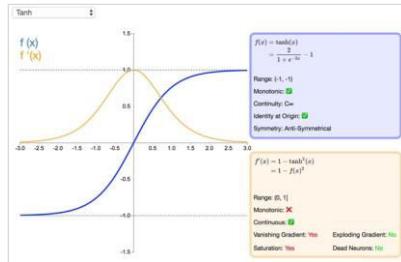
$$f(x) = \frac{1}{1 + e^{-x}}$$



- The sigmoid function is monotonic, continuous, and easy to derive. The output is bounded, and the network is easy to converge. However, we see that the derivative of the sigmoid function is close to 0 at the position away from the central point. When the network is very deep, more and more backward propagation gradients fall into the saturation area, making the gradient module smaller. Generally, if the sigmoid network has five or fewer layers, the gradient is degraded to 0, which makes training difficult. This phenomenon is a vanishing gradient. In addition, the output of the sigmoid is not zero-centered.

Tanh Function

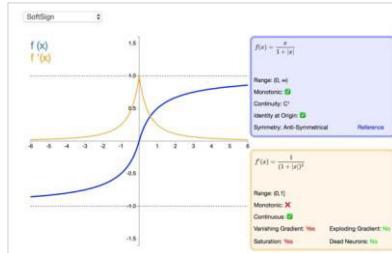
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



- The tanh function and sigmoid function have similar shortcomings. The derivative of the tanh function is nearly 0 at its extremes. However, because the tanh function is symmetric with respect to the origin, the average of the outputs is closer to 0 than that of the sigmoid function. Therefore, SGD can reduce the required number of iterations because it is closer to the natural gradient descent.

Softsign Function

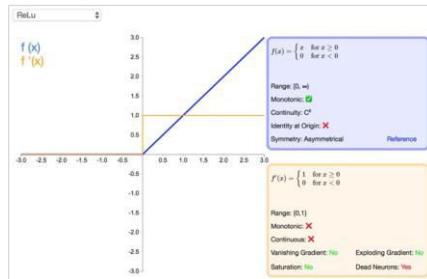
$$f(x) = \frac{x}{|x| + 1}$$



- This function saturates more slowly than the tanh function.
- When the sigmoid, tanh, and softsign functions are used to train a deep neural network, the **vanishing gradient** problem is inevitable. The derivative of each function approaches 0 at its extremes. When the network is very deep, more and more backward propagation gradients fall into the saturation area, making the gradient module smaller and finally close to 0, and the weight cannot be updated.
- Generally, if the neural network has more than five layers, the gradient is degraded to 0, which makes training difficult.

Rectified Linear Unit (ReLU) Function

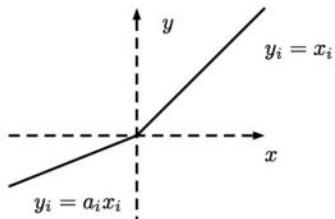
$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



- Advantages:
 - Compared with sigmoid and tanh, ReLU supports fast convergence in SGD.
 - Compared with the sigmoid and tanh functions involving exponentiation, the ReLU can be implemented more easily.
 - The vanishing gradient problem can be effectively alleviated.
 - The ReLU has a good performance during unsupervised pre-training.
- Disadvantages:
 - There is no upper bound, so it is relatively easier for the training to converge.
 - The ReLU is not differentiable at $x = 0$ and a derivative is forcibly defined at this point.
 - The surface defined at the zero point is not smooth enough in some regression problems.
- It reduces the computation workload.
- When functions such as sigmoid are used, the activation function involves exponent operation, which requires a large amount of computation. When the error gradient is computed through backward propagation, the derivation involves division and the computation workload is heavy. However, the ReLU activation function can reduce much of the computation workload.
- It effectively mitigates the vanishing gradient problem.
- The ReLU gradient is unsaturated. When the sigmoid function is close to the saturation area (far from the function center), the transformation is too slow and the derivative is close to 0. Therefore, in the backward propagation process, the ReLU function mitigates the vanishing gradient problem, and parameters of the first several layers of the neural network can be quickly updated.

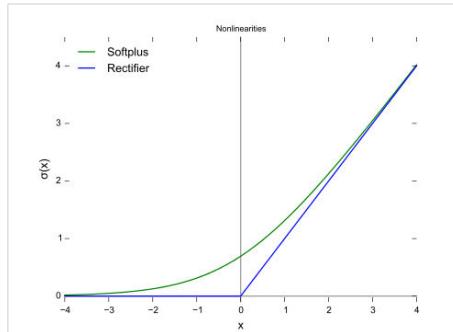
LeakyRelu Function

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \frac{x_i}{a_i} & \text{if } x_i < 0, \end{cases}$$



Softplus Function

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



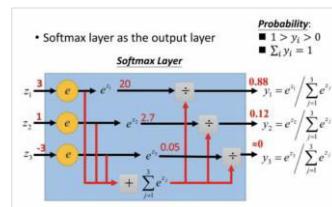
- Compared with ReLU, this function has more complex computation. However, it has a continuous derivative and defines a smooth curved surface.

Softmax Function

- Softmax function body:

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

- 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
2. Training Rules
3. Activation Functions
- 4. Normalization**
5. Optimizers
6. Neural Network Types

Overfitting

- Problem description: The model performs well in the training set, but poorly in the test set.
- Root cause: There are too many feature dimensions, model assumptions, and parameters, too much noise, but very less training data. As a result, the fitting function almost perfectly predicts the training set, whereas the prediction result of the test set of new data is poor. Training data is overfitted without considering the generalization capability of the model.

- Overfitting causes: insufficient data and complex model
 - **Obtain more data:** Obtain more data from data sources and perform data augmentation.
 - **Use a proper model:** Reduce the number of network layers and the number of neurons to limit the fitting capability of the network.
 - Dropout
 - **Normalization:** Limit the weight increase during training.
 - Limit the training time. Pass the evaluation test.
 - **Increase the noise:** Input + Weight (Gaussian initialization)
 - **Data cleansing/pruning:** Rectify or delete incorrect labels.
 - **Usage of multiple models:** Bagging uses diverse models to fit training sets of different parts. Boosting uses only simple neural networks.
- Solution to overfitting:
 - Data augmentation
 - Normalization to reduce parameter values
 - Early stopping
 - Dropout

Normalization

- Normalization is a very important and effective technology to reduce generalization errors in machine learning. It is especially useful for deep learning models which tend to have overfitting due to diverse parameters. Therefore, researchers have also proposed many effective technologies to prevent overfitting, 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

Parameter Penalty

- Many normalization methods limit the learning ability of the model by adding a parameter penalty $\Omega(\theta)$ to the objective function J . Assume that the normalized objective function is \tilde{J} .

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

where $\alpha \in [0, \infty)$ is a hyperparameter that weighs the relative contribution of the norm penalty term Ω and the standard objective function $J(X; \theta)$. If α is set to 0, no normalization is performed. A larger value of α indicates a larger normalization penalty.

- Generally, in deep learning, we add constraints only to affine parameter w but not to biases. The main reason is that biases usually require less data for more accurate fitting. Adding constraints often leads to underfitting.

L_1 Normalization

- Add L_1 norm constraints 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:

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

- The parameter optimization method is:

$$w_{t+1} = (w - \varepsilon \alpha sign(w)) - \varepsilon \nabla J(w)$$

- In special cases, for secondary optimization, assuming that the corresponding Hessian matrix is a diagonal matrix, the parameter recursive formula is $w_i = sign(w_i^*) \max(|w_i^*| - \frac{\alpha}{\lambda_i}, 0)$. According to this formula, the parameter value is reduced to 0 if $|w_i^*| < \frac{\alpha}{\lambda_i}$. This implementation is different from L_2 normalization. Compared with L_1 normalization, L_2 normalization does not directly reduce the parameter value to 0 but to a value close to 0.

L_2 Normalization

- The L_2 norm penalty term is added to the parameter constraint. This technique is used 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):

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

where ε is the learning rate. Compared with the normal gradient optimization formula, the parameter is multiplied by a reduction factor.

- If J is a secondary optimization formula, the model parameter may be further represented as $\tilde{w}_i = \frac{\lambda_i}{\lambda_i + \alpha}w_i$, that is, adding a control factor to the original parameter, where λ is an eigenvalue of the parameter Hessian matrix. Therefore:
 - When $\lambda_i \gg \alpha$, the penalty factor has a small effect.
 - When $\lambda_i \ll \alpha$, the corresponding parameter value is reduced to 0.

L_2 VS. L_1

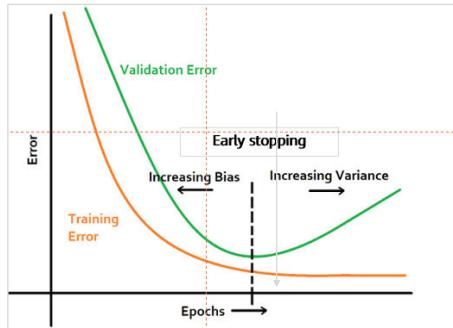
- The differences between L_2 and L_1 are as follows:
 - According to the preceding analysis, L_1 can generate a sparser model than L_2 . When the value of parameter w is small, L_1 normalization can directly reduce the parameter value to 0, which can be used for feature selection.
 - From the perspective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

Data Augmentation

- The most effective way to prevent overfitting is to add a training set. A larger training set has a smaller overfitting probability. Data augmentation is time-saving and effective but not applicable in certain 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 should not be changed through transformation. For example, for handwriting digit recognition, classes 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.

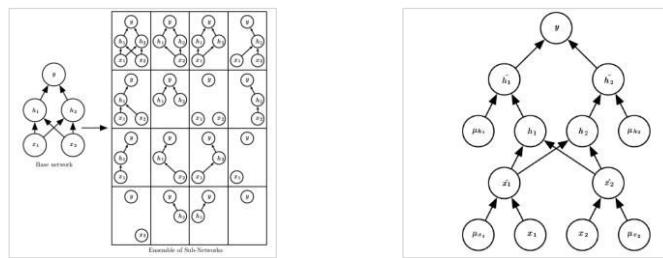
Early Stopping of Training

- A test on data of the validation set can be inserted during the training. When the data loss of the validation set increases, the training is stopped in advance.



Dropout

- Dropout is a common and simple normalization 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. Dropout is an integration method. It combines all sub-network results and obtains sub-networks by randomly dropping inputs. For example:



- The sampling probability of each entry is 0.8 for the input and 0.5 for the hidden layers.
- Advantages:
 - Compared with weight decay and norm constraints, this strategy is more effective.
 - It is computationally cheap and simple and can be used in other non-deep-learning models.
 - However, it is less effective when the training data is insufficient.
 - Stochasticity is not necessary or sufficient to achieve the normalizing effect of dropout. Invariant shielding parameters can be constructed to obtain good solutions.
- In addition to the preceding methods, we can also use semi-supervised learning, multi-task learning, early stopping, parameter sharing, ensemble methods, and adversarial training.

Contents

1. Deep Learning
2. Training Rules
3. Activation Functions
4. Normalization
- 5. Optimizers**
6. Neural Network Types

Optimizers

- There are various improved versions of gradient descent algorithms. In object-oriented language implementation, different gradient descent algorithms are often encapsulated into objects called **optimizers**.
- The **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.

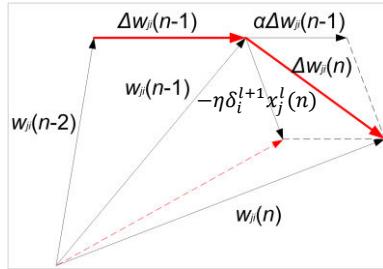
Momentum Optimizer

- A most basic improvement is to add momentum terms for Δw_{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 α is a constant ($0 \leq \alpha < 1$) called momentum coefficient and $\alpha \Delta w_{ji}^l(n-1)$ 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.



- $\eta \delta_i^{l+1} x_j^l(n)$ indicates the original gradient size and direction, j represents a j -th neuron at an l -th layer, and i represents an i -th neuron at an $(l+1)$ -th layer

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 η and momentum α need to be manually set, which often requires more experiments to determine the appropriate value.

AdaGrad Optimizer (1)

- The common feature of the stochastic gradient descent (SGD) algorithm, 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.

$$g_t = \frac{\partial C(t, o)}{\partial w_t} \quad \text{Computing gradient}$$

$$r_t = r_{t-1} + g_t^2 \quad \text{Square gradient accumulation}$$

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t \quad \text{Computation update}$$

$$w_{t+1} = w_t + \Delta w_t \quad \text{Application update}$$

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

AdaGrad Optimizer (2)

- It can be seen from the AdaGrad optimization algorithm that, as the algorithm is continuously iterated, the r becomes larger and the overall learning rate becomes smaller. This is because we hope the **LR becomes slower** 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 and closer to the optimal solution, and therefore LR can decrease.
- Advantages:
 - The learning rate is automatically updated. As the number of updates increases, the learning rate decreases.
- Disadvantages:
 - The denominator keeps accumulating. As a result, 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 of the AdaGrad optimizer ending the optimization process too early. It is suitable for non-stable target handling and has good effects on RNNs.

$$g_t = \frac{\partial C(t, o)}{\partial w_t} \quad \text{Computing gradient}$$

$$r_t = \beta r_{t-1} + (1 - \beta) g_t^2 \quad \text{Square gradient accumulation}$$

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t \quad \text{Computation update}$$

$$w_{t+1} = w_t + \Delta w_t \quad \text{Application update}$$

- g_t indicates the t th gradient, r_t is a gradient accumulation variable, and the initial value of r is 0, **which may not increase and may need to be adjusted using a parameter**. β indicates the decay factor, and η indicates the global LR, which needs to be set manually. ε is a small constant, and is set to about 10^{-7} for numerical stability.

- m indicates the number of samples used.

Adam Optimizer (1)

- Adaptive moment estimation (Adam): Developed based on AdaGrad and AdaDelta, Adam maintains two additional variables m_t and v_t for each variable to be trained:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where t represents the t -th iteration and g_t is the computed gradient. m_t and v_t are moving averages of the gradient and square gradient. From the statistical perspective, m_t and v_t are estimates of the first moment (the average value) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method.

- Adam attempts to compute adaptive learning rates for each parameter. This is very useful in complex network structures because different parts of the network have different sensitivities to weight adjustment. A very sensitive part usually requires a smaller learning rate. It is difficult or complex to manually identify the sensitive part and set a learning rate. This may be the best optimizer at present.

Adam Optimizer (2)

- If m_1 and v_1 are initialized using the zero vector, m_t and v_t are close to 0 during the initial iterations, especially when β_1 and β_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 η , β_1 , and β_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, η can be reduced. After several times of reduction, a satisfying local extrema will be obtained. Other parameters do not need to be adjusted.

Contents

1. Deep Learning
2. Training Rules
3. Activation Functions
4. Normalization
5. Optimizers
- 6. Neural Network Types**

Convolutional Neural Network (CNN)

- A CNN is a feedforward neural network. Its artificial neurons may respond to **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.

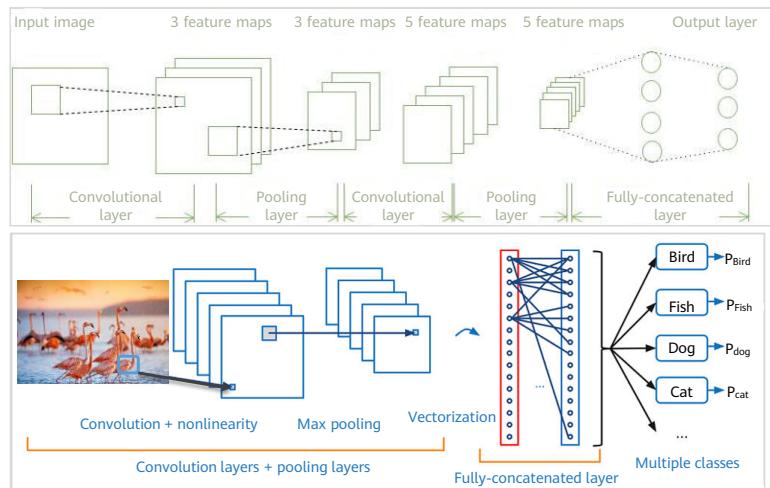
- A filter matrix is a set of fixed weight and can be seen as a constant filter (kernel). The convolution (adding each element, weighted by the kernel) is performed between an image (data from different data windows) and a kernel. This type of network is called CNN.
- Local receptive field: It is generally 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 pixels that are far away. Therefore, each neuron does not need to know the global image. It only needs to know the local image and the local information is then combined at a higher level to generate global information. The idea of local network connection is also inspired by the biological visual system structure. The neurons in the visual cortex receive local information (respond to stimuli of certain regions).
- Parameter sharing: One or more filters can be used to scan the input images. The parameters of the filter are weights. At the layers scanned by the same filter, each filter uses the same parameters to perform weighted computation. Weight sharing means that the parameter values of each filter do not change when the filter scans the entire image. For example, if we have three feature filters and each filter scans the entire image. During the scanning process, the parameter values of the filters do not change. In other words, all elements of the image share the same weights.

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 pixels that are far away.** Therefore, each neuron does not need to know the global image. It only needs to know the local image and the local information is then combined at a higher level to generate global information.
- **Parameter sharing:** One or more convolution cores may be used to scan input images. Parameters carried by the convolution cores are weights. In a layer scanned by convolution cores, each core uses the same parameters during weighted computation. Weight sharing means that when each convolution core scans an entire image, **the parameters of the convolution core are fixed.**

- Local receptive field: The idea of local network connection is also inspired by the biological visual system structure. The neurons in the visual cortex receive local information (respond to stimuli of certain regions).
- Parameter sharing: For example, if we have three feature convolution kernels and each kernel scans the entire image, the parameter values of the convolution kernels do not change during the scanning process. In other words, all elements of the image share the same weights. This means that the features learned from a part of the image can also be applied to other parts of the image or other images, which is called position invariance.

CNN Architecture



63 Huawei Confidential



- Input layer: inputs data.
- Convolutional layer: composed of several convolutional units. The parameters of each convolutional unit are obtained by optimizing the backward propagation algorithm. The purpose of convolution computation is to extract different input features. The first convolutional layer may extract only some low-level features such as edges, lines, and angles. A multi-layer network can extract more complex features based on low-level features.
- Rectified linear units layer (ReLU layer): uses ReLU $f(x) = \max(0, x)$ as the activation function.
- Pooling layer: partitions features obtained from the convolutional layer into some areas and outputs the maximum or minimum value, generating new features with a smaller spatial size.
- Fully-connected layer: integrates all local features into global features to compute the final scores for each type.
- Output layer: outputs the final result.
- Image source: smartvi
- http://image.huawei.com/tiny-lts/v1/images/0281bccdff8137c18a163beba5a2650c_4715x2927.jpg

Computing a Convolution Kernel (1)

- Convolution computation

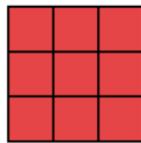
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

image 5*5

1	0	1
0	1	0
1	0	1

bias=0

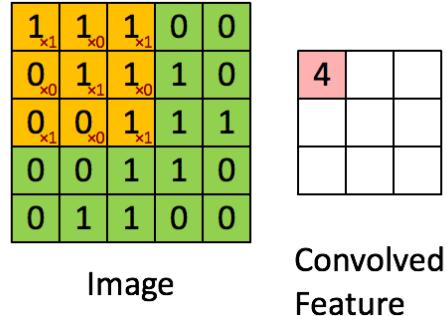
filter 3*3



feature map 3*3

Computing a Convolution Kernel (2)

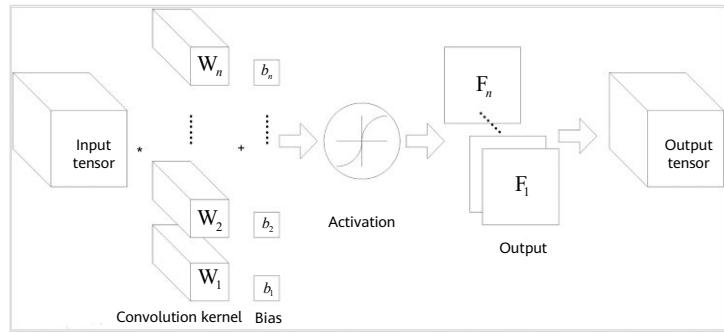
- Convolution computation result



hanbingtao, 2017, CNN

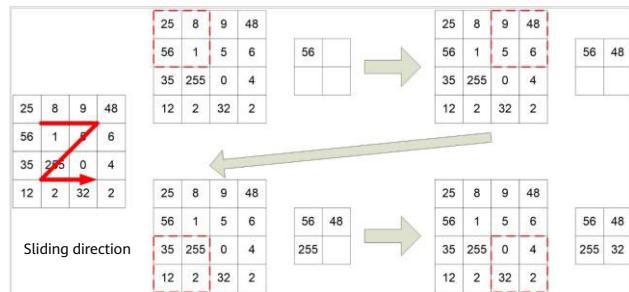
Convolutional Layer

- The basic architecture of a CNN is a 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 convolution core of the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learned. Similar to the fully-connected layer, after the convolution is complete, the result is 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, whereas 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 pool window size is 2.



- The actual classification networks are feedforward networks that are formed by interconnected convolutional and pooling layers. The pooling layer has the following functions:
 - Invariance: Max pooling ensures invariance within a certain range because the maximum value of a region is the last output value regardless of where the value is.
 - Reducing the input size for the next layer: Pooling effectively reduces the size of the input data for the next layer, the number of parameters, and the computation workload.
 - Obtaining fixed-length data: By properly setting the pooling window size and stride, we can obtain fixed-length outputs from variable-length inputs.
 - Increasing the scale: The features of the previous layer can be extracted from a larger scale.
 - Preventing overfitting: Pooling simplifies the network and reduces the fitting precision. Therefore, it can prevent overfitting (we must also pay attention to the possible underfitting).

Fully-concatenated 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 compute the score of each type.

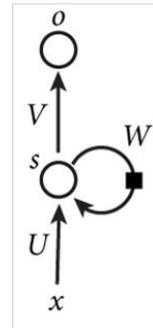
$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Recurrent Neural Network (RNN)

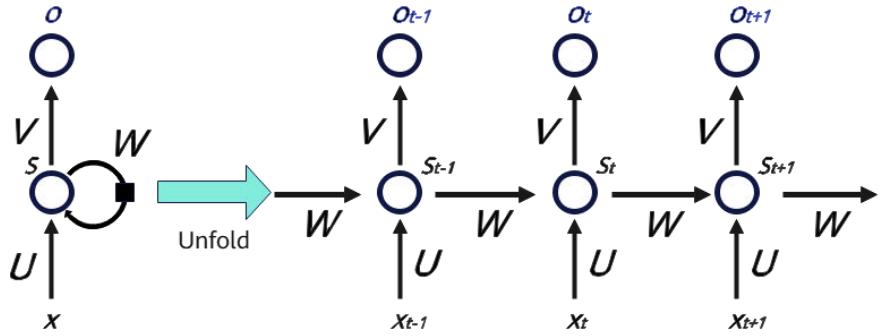
- RNN is a neural network that **captures dynamic information in sequential data** through periodical connections of hidden layer nodes. It can classify sequential data.
- Unlike FNNs, the RNN can keep a context state and even store, learn, and express related information in context windows of any length. Different from traditional neural networks, it is not limited to the space boundary **but it 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.

RNN Architecture (1)

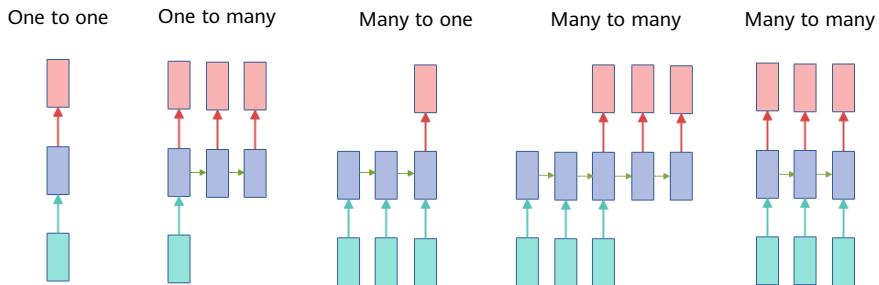
- X_t is the input of the input sequence at time t.
- S_t is the memory cell of the sequence at time t and caches previous information.
$$S_t = \sigma(UX_t + WS_{t-1}).$$
- S_t passes through multiple hidden layers, and then passes through the fully-connected layer V to obtain the final output O_t at time t.
$$O_t = \text{softmax}(S_tV).$$



RNN Architecture (2)



RNN Types



- From left to right: (1) Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). (2) Sequence output (e.g. image captioning takes an image and outputs a sentence of words). (3) Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing a positive or negative sentiment). (4) Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). (5) Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case there are no pre-specified constraints on the lengths of sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.
- Andrej Karpathy, 2015, The Unreasonable Effectiveness of Recurrent Neural Networks
- Image: original

Backward Propagation Through Time (BPTT)

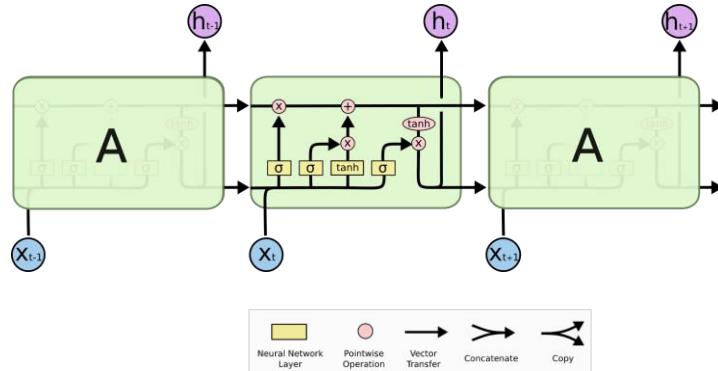
- BPTT:
 - It is an extension of traditional backward propagation based on time sequences.
 - There are two errors in a memory cell at time sequence t : 1. Partial derivative of the error C_t at time sequence t to the memory cell. 2. Partial derivative of the error at the next time sequence $t+1$ of the memory cell to the memory cell at time sequence t . The two errors need to be added.
 - If the time sequence is longer, it is more likely that the loss of the last time sequence to the gradient of weight w in the first time sequence will cause the vanishing gradient or exploding gradient problem.
 - The total gradient of weight w is the accumulation of the gradient of the weights in all time sequences.
- Three steps of BPTT:
 - Compute the output of each neuron (forward).
 - Compute the error δ_j of each neuron (backward).
 - Compute the gradient of each weight.
- Updating weights using the SGD algorithm

- z_t is the value before the memory cell enters the activation function, $\delta_t = \frac{\partial C_t}{\partial z_t} + w^T \times f'_t(z_t) \times \delta_{t+1}$

RNN Problems

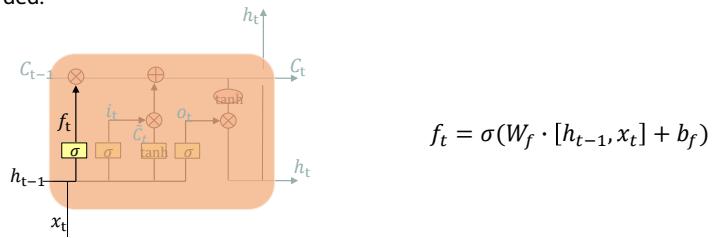
- $S_t = \sigma(UX_t + WS_{t-1})$ is extended based on time sequences.
- $S_t = \sigma\left(UX_t + W\left(\sigma\left(UX_{t-1} + W\left(\sigma\left(UX_{t-2} + W(\dots)\right)\right)\right)\right)\right)$
- Despite that the standard RNN structure solves the problem of information memory, **the information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

Long Short-Term Memory (LSTM) Network



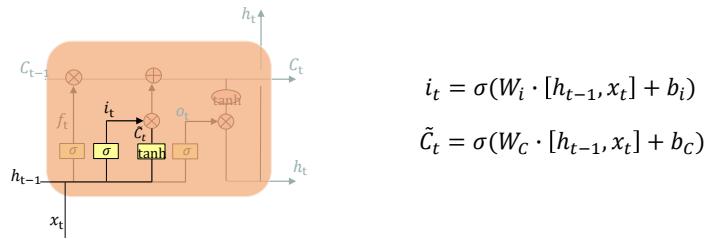
Forget Gate in LSTM

- The first step in LSTM is to decide what information to discard from the cell state.
- The decision is made through the forget gate. This gate reads h_{t-1} and x_t and outputs a numeric value in the range from 0 to 1 for each digit in the cell state C_{t-1} . The value **1** indicates that the information is completely retained while the value **0** indicates that the information is completely discarded.



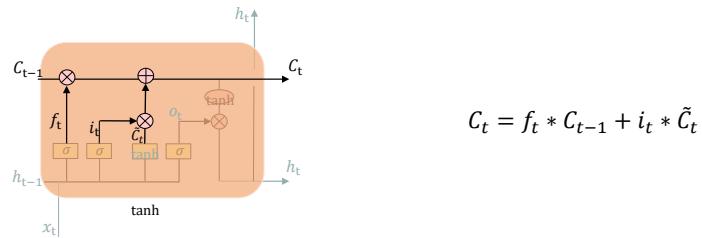
Input Gate in LSTM

- This step is to determine what information is stored in the cell state.
- It consists of two parts:
 - The sigmoid layer is called the input gate layer, which determines the value to be updated.
 - A candidate value vector is created at the tanh layer and is added to the state.



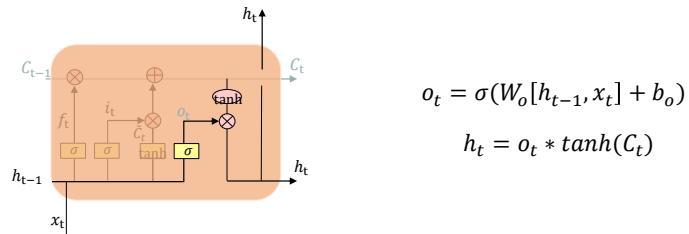
Update in LSTM

- In this step, it is time to update the state of the old cell. C_{t-1} is updated as C_t .
- We multiply the old state by f_t and discard information we decide to discard. Then add $i_t * C_t$.
This is a new candidate value, scaled by how much we decided to update each state value.



Output Gate in LSTM

- We run a sigmoid layer to determine which part of the cell state will be output.
- Then we process the cell state through tanh (a value between -1 and 1 is obtained) and multiply the value by the output of the sigmoid gate. In the end, we output only the part we determine to output.



Quiz

1. (Single-answer question) Which of the following is not a deep learning neural network?
()
A. CNN
B. RNN
C. LSTM
D. Logistic

- 1.D

Quiz

2. (Multiple-answer question) Which of the following are CNN "components"? ()
- A. Activation function
 - B. Convolutional kernel
 - C. Pooling
 - D. Fully-connected layer

- 2.ABCD

Quiz

3. (True or false) Compared with RNN, CNN is more suitable for image recognition. ()
 - A. True
 - B. False

- Answers: 3. A

Summary

- This chapter mainly introduces the definition and development of neural networks, perceptrons and their training rules, and common types of neural networks.

Recommendations

- Huawei Talent
 - <https://e.huawei.com/en/talent/#/home>
- Huawei Support Knowledge Base
 - <https://support.huawei.com/enterprise/en/knowledge?lang=en>

Thank you.

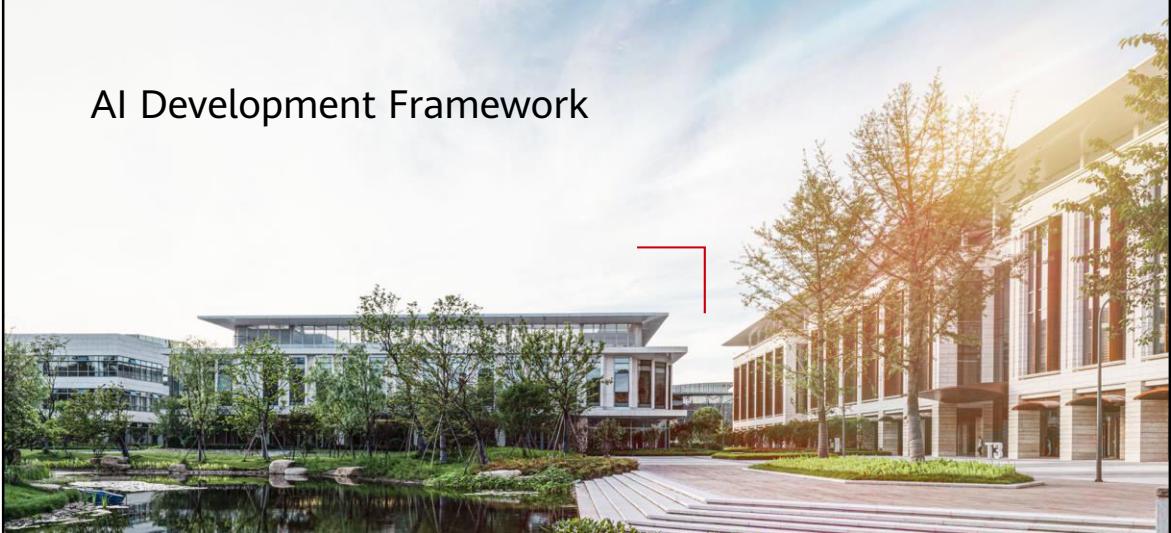
Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright©2023 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.



AI Development Framework



 HUAWEI

Foreword

- Over the past few decades, machine learning has been the subject of extensive research and application. In fields such as image recognition, speech recognition and synthesis, autonomous driving, and machine vision, deep learning has witnessed significant achievements. This also poses higher requirements on the algorithm application and dependent frameworks.
- With the continuous development of the deep learning framework, a large number of computing resources can be conveniently used when a neural network model is trained on a large dataset. This course describes how to use the AI framework and develop AI applications.

Objectives

- On completion of this course, you will be able to:
 - Understand the current mainstream AI development framework.
 - Master the AI application development process.
 - Understand the structure and features of MindSpore.
 - Understand MindSpore development components.

Contents

- 1. AI Framework Development**
2. MindSpore
3. MindSpore Features
4. MindSpore Development Components
5. AI Application Development Process

AI Development Framework

- The deep learning framework lowers the entry barriers for AI development. We can use the existing model to configure parameters as required and train the model automatically, instead of compiling the code through the complex neural network and backpropagation algorithm. We also can add the user-defined network layer based on the existing model, or select the required classifier and optimization algorithm.

TensorFlow

PyTorch

PaddlePaddle

[M]^s 昇思
MindSpore



- A deep learning framework is like a set of building blocks. A model or algorithm is the equivalent to each block. We can assemble the required building block model with different blocks and avoid repetitive work.

Current Market Environment

- Similar AI framework:
 - Outside China: TensorFlow (Google), PyTorch (Meta), MXNet (Amazon)...
 - China: Paddle (Baidu), MindSpore (Huawei)...
- Mainstream computing processors:
 - GPU
 - CPU
 - NPU

PyTorch

- PyTorch is a machine learning computing framework released by Meta, formerly known as Torch. Torch is a scientific computing framework supported by a large number of machine learning algorithms and a tensor library similar to Numpy. Torch has a flexible programming environment but is not commonly used because its programming language is Lua. Therefore, PyTorch is created based on Python.
- PyTorch is a tensor library optimized for deep learning using GPUs and CPUs, as well as a mainstream deep learning framework. Enterprises such as Twitter, GMU, and Salesforce all adopt PyTorch.

PyTorch Features

- **Python preferred:** PyTorch supports fine-grained access for Python instead of adopting Python based on C++ framework. We can use PyTorch as easily as using Numpy and Scipy.
- **Dynamic neural network:** A static computation graph is required before running the TensorFlow 1.x and then repeatedly executing the graph through feed and run operations. But PyTorch programs can dynamically build or modify computational graphs during their execution.
- **Easy to debug:** PyTorch can generate dynamic diagrams during running. Developers can stop the interpreter in the debugger and view the output of a node.
- PyTorch provides tensors that can run on CPU and GPU and greatly accelerate the computation.

TensorFlow

- TensorFlow (TF) is an end-to-end open-source machine learning platform designed by Google. Currently, the main version is TensorFlow 2.x.
 - TensorFlow has three iterative versions: TensorFlow 0.1, TensorFlow 1.0, and TensorFlow 2.0. Currently, TensorFlow 2.x is mainly used.
 - TensorFlow 2.X includes TensorFlow.js, JavaScript, TensorFlow Lite, and TensorFlow Extended. They build a complete ecosystem for TensorFlow.
 - AlphaGo Zero is trained based on TensorFlow.

- TensorFlow.js: a library used to develop and train machine models in JavaScript and deployed on browsers or Node.js.
- TensorFlow Lite: an open-source deep learning framework designed for on-device inference. TensorFlow Lite provides a set of tools that enables on-device machine learning by allowing developers to run their trained models on mobile and IoT devices and computers.
- TensorFlow Extended (TFX): a machine learning platform for creating and managing end-to-end pipelines that transfer trained models from research to production.

TensorFlow 2.x VS TensorFlow 1.x

- Disadvantages of TensorFlow V1.0:
 - In TensorFlow 1.0, the result generated for creating a tensor cannot be directly returned. Instead, a session mechanism needs to be created and the graph is included. Then the session needs to be run. This style is similar to the hardware programming language VHDL.
 - TensorFlow 1.0 is difficult to debug, APIs are disordered, and it is difficult for beginners. As a result, many researchers change to use PyTorch.

TensorFlow 2.x VS TensorFlow 1.x

- TensorFlow 2 features

- Advanced Keras API

Easy to use (the most important feature): The graph and session mechanisms are removed.

Main improvements:

- The core function of TensorFlow 2 is the dynamic graph mechanism **Eager Execution**. This allows users to compile programs and debug models and make developers easier to learn and use TensorFlow.
- Supports more platforms and languages, and improves the compatibility between components by standardizing the API exchange format and providing base lines.
- Delete discarded and duplicate APIs to prevent confusion.
- Although tf.contrib is no longer used, the valuable modules will be applied in other applications, and the remaining will be deleted.

MindSpore

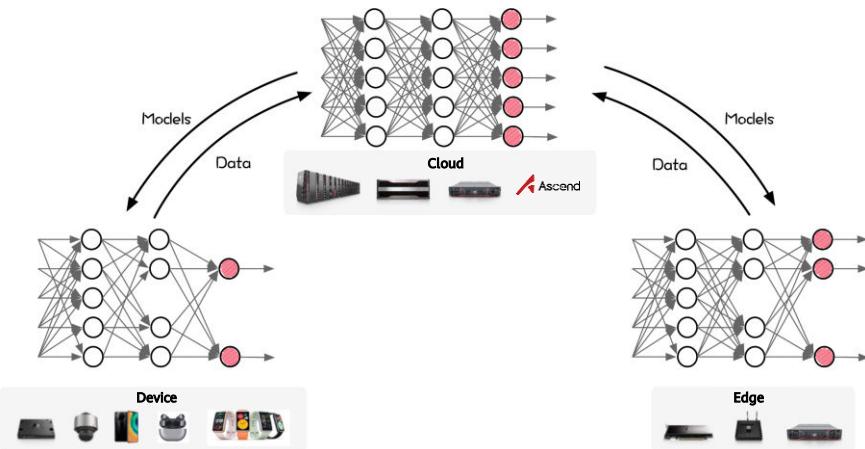
- MindSpore is Huawei's next-generation deep learning framework. By leveraging the computing power of Huawei Ascend processors, enabling flexible deployment in device, edge, and cloud scenarios, and integrating the industry's best practices, MindSpore defines a new paradigm in AI programming and lowers the threshold for AI development.
 - MindSpore aims to achieve three goals: easy development, efficient execution, and all-scenario coverage.
 - MindSpore is highly flexible and supports data parallel, model parallel, and hybrid parallel training.
 - MindSpore has the automatic parallelism capability which efficiently searches for a fast parallel strategy in a large strategy space.



Contents

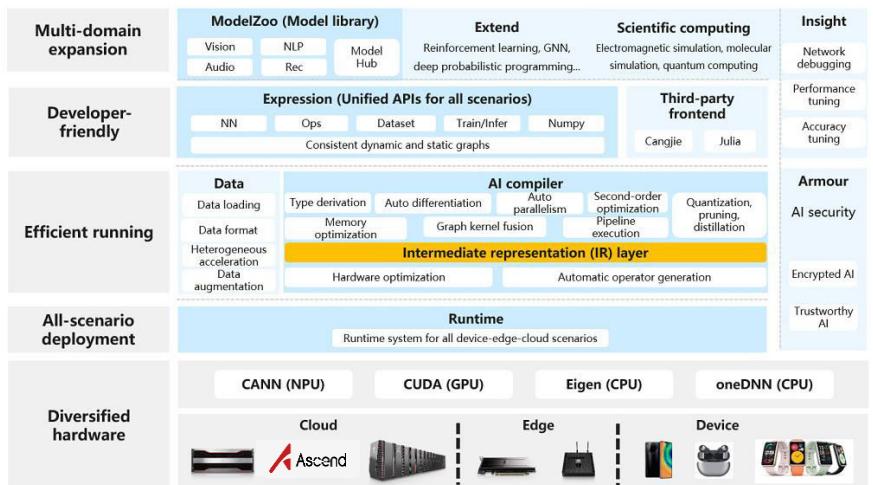
1. AI Framework Development
2. **MindSpore**
 - **MindSpore Architecture**
 - MindSpore Framework Basics
3. MindSpore Features
4. MindSpore Development Components
5. AI Application Development Process

All-Scenario AI Framework



- MindSpore aims to provide a device-edge-cloud AI framework for all scenarios. MindSpore can be deployed on the device, edge, and cloud to meet differentiated requirements. For example, MindSpore supports lightweight deployment on the device and various training functions on the cloud, including auto differentiation, mixed precision, and easy-to-use model programming.
- MindSpore has the following key all-scenario features:
 - Unified C++ inference API across the device, edge, and cloud: supports quick porting of algorithm code to different hardware environments.
 - Unified model: The device and cloud use the same model format and definition with the same software architecture. MindSpore supports multiple hardware, including Ascend, GPU, and CPU (such as x86 and ARM). It achieves "Train Once, Deploy Everywhere."
 - Diversified computing power is supported. MindSpore provides a unified southbound API for quickly adding and using new hardware.
 - Model compression technology that adapts to different hardware environments and scenarios, including model quantization, and quick application of device-edge-cloud collaborative technologies, such as federated learning.

MindSpore Architecture



15 Huawei Confidential

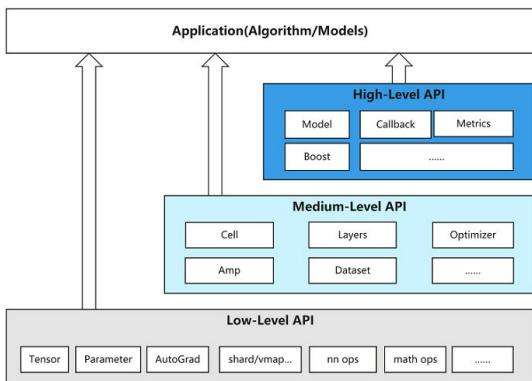


- The MindSpore architecture consists of four layers:
 - Model layer: provides out-of-the-box functions for users and contains preset models, development kit, and extended libraries for hot research fields such as graph neural network (GNN) and deep probability programming.
 - MindExpression: provides users with unified APIs for AI model development, training, and inference. Users can use the native Python syntax to develop and debug neural networks. MindSpore provides an encoding mode that unifies dynamic and static graphs, which helps developers balance the development efficiency and execution performance. Besides, all-scenario C++ interfaces are provided in the production and deployment phase.
 - MindCompiler: compiles frontend expressions into underlying languages with higher execution efficiency as the core of the AI framework based on MindIR. Besides, optimizes global performance, including hardware-independent optimization such as automatic differentiation and expression simplification, and hardware-related optimization such as graph kernel fusion and operator generation.
 - During runtime, underlying hardware operators are interconnected and invoked based on the optimized result of upper-layer compilation. Besides, the unified device-edge-cloud runtime architecture supports AI

collaboration in cloud, edge, and device, including federated learning.

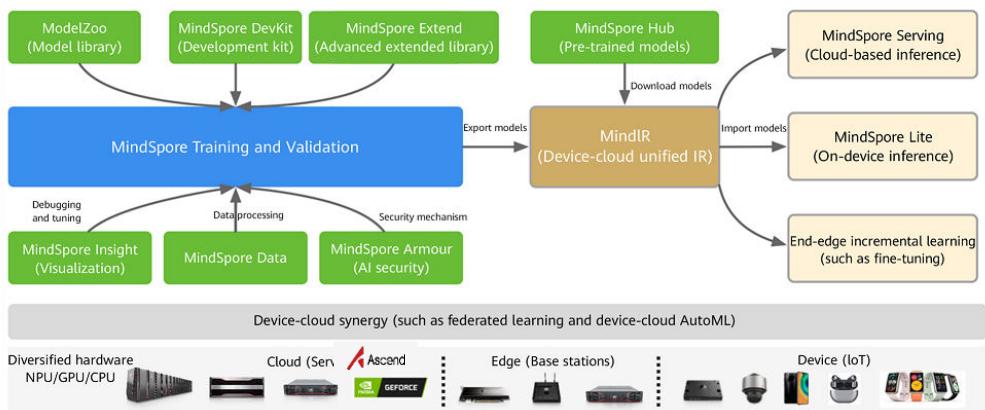
API Level Structure

- To develop AI applications, algorithms, and models, MindSpore provides users with three levels of APIs: low-level Python APIs, medium-level Python APIs, and high-level Python APIs.
- High-level Python APIs have better encapsulation, low-level APIs have better flexibility, and medium-level APIs have both flexibility and encapsulation, meeting developers' requirements in different fields and at different levels.



- MindExpression API Level Structure
- High-Level Python API:
 - High-level APIs are at the first layer, and include training and inference management, mixed precision training, debugging, tuning, and other advanced APIs that are based on medium-level APIs, enabling users to control the execution process of the entire network and implement training, inference, and optimization of the neural network. For example, by utilizing the Model API, users can specify the neural network model to be trained as well as related training settings and train the neural network mode.
- Medium-Level Python API:
 - Medium-level APIs are at the second layer, which encapsulates low-level APIs and provides modules such as the layer, optimizer, and loss function. Users can flexibly build neural networks and control execution processes through the medium-level APIs to quickly implement model algorithm logic. For example, users can call the Cell API to build neural network models and computing logic, add the loss function and optimization methods to the neural network model by using the loss module and Optimizer API, and use the dataset module to process data for model training and derivation.
- Low-Level Python API:
 - Low-level APIs are at the third layer, which includes tensor definition, basic operators, and automatic differentiation modules, enabling users to easily define tensors and perform derivative computation. For example, users can customize tensors by using the Tensor API, and use the GradOperation operator in the ops.composite module to calculate the derivative of the function at a specified position.

MindSpore All-Scenario Support Mode



17 Huawei Confidential



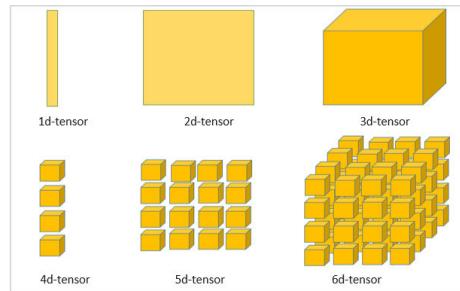
- As an all-scenario AI framework, MindSpore supports different series of hardware in the device (mobile phone and IoT device), edge (base station and routing device), and cloud (server) scenarios, including Ascend series products and NVIDIA series products, Qualcomm Snapdragon in the ARM series, and Huawei Kirin chips.
- The blue box on the left is the main framework of MindSpore, which mainly provides the basic API functions related to the training and verification of neural networks, and also provides automatic differentiation and automatic parallelism by default.
- Below the blue box is the MindSpore Data module, which enables data preprocessing, including data sampling, data iteration, data format conversion, and other operations. Many debugging and tuning problems may occur during training. Therefore, the MindSpore Insight module visualizes debugging and tuning data such as the loss curve, operator execution status, and weight parameter variables, facilitating debugging and tuning during training.
- The simplest way to ensure AI security is from the perspective of attack and defense. For example, attackers inject malicious data in the training phase to affect the inference capability of AI models. Therefore, MindSpore features the MindSpore Armour module, which provides an AI security mechanism.
- The content above the blue box is for algorithm development, including the AI algorithm model library ModelZoo, development toolkit MindSpore DevKit for different fields, and the advanced extension library MindSpore Extend. MindSciences, a scientific computing kit in MindSpore Extend, is worth mentioning. MindSpore is the first to combine scientific computing with deep learning, combine numerical computing with deep learning, and support electromagnetic simulation and drug molecular simulation through deep learning.
- After the neural network model is trained, you can export the model or load the model that has been trained in MindSpore Hub. Then MindIR provides a unified IR format for the device and cloud, which defines logical network structures and operator attributes through a unified IR, and decouples model files in MindIR format from hardware platforms to implement one-time training and multiple-time deployment. As shown in the figure, the model is exported to different modules through IR to perform inference.

Contents

1. AI Framework Development
2. **MindSpore**
 - MindSpore Architecture
 - **MindSpore Basics**
3. MindSpore Features
4. MindSpore Development Components
5. AI Application Development Process

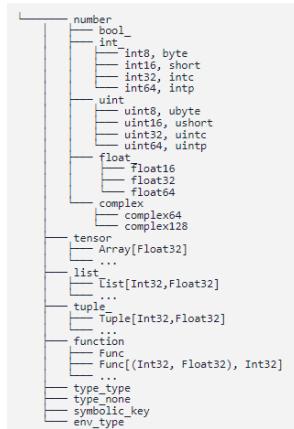
Tensor

- The most basic data structure in MindSpore is tensor. All data is encapsulated in tensors.
 - Tensor: a multi-dimensional array.
 - Zero-order tensor: scalar
 - First-order tensor: vector
 - Second-order tensor: matrix
- In MindSpore, you can use the following methods to quickly create a tensor:
 - `mindspore.Tensor`.
 - Collect a set of data to create a tensor. Dimension can be specified during creation.



Data Types in MindSpore

- MindSpore supports data types such as int, uint, and float.
 - For details, see the figure on the right.
- In MindSpore, you can use **mindspore.dtype** to create a data type object.
- MindSpore is compatible with data types in NumPy and Python, and can convert data types through **mindspore.dtype_to_npytyp** and **mindspore.dtype_to_pytype**.



Running Environment Setup

- When you run MIndSpore, you need to specify environment parameters, including the graph mode, device, and memory size. You can use related APIs to obtain detailed information about the current running environment.
- APIs related to the running environment:
 - `mindspore.context.set_context` is used to set the context of the running environment.
 - `mindspore.context.get_context` is used to obtain the attribute value in the context according to the input key.
 - `mindspore.context.ParallelMode` is used to enable the parallel mode.
 - `mindspore.context.set_ps_context` is used to set the context of the training mode in the parameter server
 -

context

- The context of MindSpore can be used to set up the current operating environment, including the execution mode, execution backend, and feature switching.
 - `mindspore.context.set_context(**kwargs)`

Parameter	Description	Value
device_id	Target device ID	Value range: [0,4096-1]. The default value is 0 .
device_target	Target device to be run	Value range: Ascend, GPU, and CPU
mode	Selects the running mode	The default value is GRAPH_MODE , GRAPH_MODE/0 or PYNATIVE_MODE/1 .
enable_sparse	Whether to enable the sparse feature	The default value is False .
save_graphs	Whether to save the computation graph	The default value is False .
runtime_num_threads	Controls the number of threads in the thread pool during running	The default value is 30 .

- Before running the program, you need to set the context. If you do not set the context, the system automatically sets the context based on the device target by default.

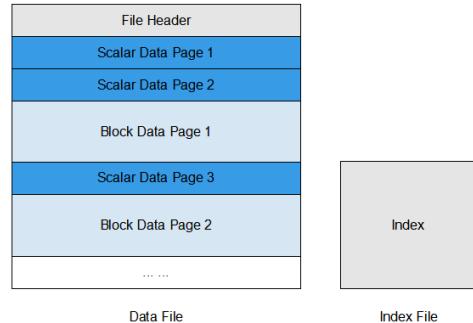
MindSpore Data Processing Module

- MindSpore provides the dataset submodule for processing datasets.
 - The **mindspore.dataset** module provides APIs for loading and processing various public datasets, such as MNIST, CIFAR-10, COCO, ImageNet, and CLUE. It can also load datasets in standard formats in the industry, such as MindRecord, TFRecord, and Manifest. You can also use this module to define and load your own datasets.
 - This module also provides the data enhancement function for data such as voice, text, and image.

MindRecord

- To read data efficiently, serialize the data and store it in a set of files that can be read linearly. Each file ranges from 100 MB to 200 MB.

- MindRecord is an efficient data format developed by MindSpore.
 - The **mindspore.mindrecord** module converts different datasets into the MindRecord format and provides some methods to read, write, and retrieve MindRecord data files.
 - MindRecord data format can reduce the disk I/O and network I/O overhead.



- The data files contain the file header, scalar data page, and block data page, which are used to store the training data after normalization. It is recommended that the size of a single MindRecord file be less than 20 GB. You can partition a big dataset into multiple MindRecord files. The index files contain index information generated based on scalar data (such as image label and image file name), which is used to conveniently search and collect dataset information. The functions of the file header, scalar data page, and block data page in data files are as follows:
 - **File header:** The metadata of the MindSpore Record file. It is used to store the file header size, scalar data page size, block data page size, schema information, index fields, statistics information, file partition information, and mapping between scalar data and block data.
 - **Scalar data page:** Used to store integers, character strings, and floating-point data, such as the image label, image file name, and image length and width. That is, the information that can be stored in scalar data is stored on this page.
 - **Block data page:** Used to store data such as a binary string and a NumPy array, for example, a binary image file and a dictionary converted from text.
- The data format of MindRecord has the following features:
 - Implement unified storage and access of user data, which simplifies data loading during training.
 - Aggregate data for storage and reading, which facilitates data management and movement during training.
 - Data encoding and decoding are efficient and transparent to users.

- Flexibly control the partition size to implement distributed training.

Neural Network Module in MindSpore

- The **mindspore.nn** module provides predefined building blocks or computing units in a neural network.
 - The module contains the following components for building the neural network:
 - Network structures such as RNN, CNN, and LSTM.
 - Loss functions such as MSELoss and SoftmaxCrossEntropyWithLogits.
 - Optimizers such as Momentum and Adam.
 - Model evaluation indicators such as F1 Score and AUC.
- **mindspore.Model** is the high-level API for model training and inference.

Callback Function

- The callback function in MindSpore is not a function but a class. You can use the callback function to observe the internal status and related information of the network during training or to perform specific actions in a specific period. For example, monitor loss functions, save model parameters, dynamically adjust parameters, and terminate training tasks in advance.
- MindSpore allows users to insert user-defined operations in a specific phase of training or inference through the following callback capabilities:
 - Callback classes provided by MindSpore framework, such as ModelCheckpoint, LossMonitor, and SummaryCollector.
 - User-defined callback.

Model File for Inference

MindSpore can execute inference tasks on different hardware platforms based on trained models.

Two types of data are supported: **training parameters and network models**.

1. Training parameters are stored in the checkpoint format.
2. Network models are stored in the MindIR, AIR, or ONNX format.

File Format	Basic Concept	Application Scenario
Checkpoint	Uses the Protocol Buffers format and stores all network parameter values	Used to resume training after a training task is interrupted or execute a fine-tuning task after training.
MindIR	A graph-based function-like IR of MindSpore which defines scalable graph structures and operator IRs.	Eliminates model differences between different backends and is generally used to perform inference tasks across hardware platforms.
ONNX	An open format built to represent machine learning models.	Used for model migration between different frameworks or used on the inference engine TensorRT.
AIR	An open file format defined by Huawei for machine learning.	Adapts to Huawei AI processors well and is generally used to execute inference tasks on Ascend 310.

Contents

1. AI Framework Development
2. MindSpore
- 3. MindSpore Features**
4. MindSpore Development Components
5. AI Application Development Process

Dynamic and Static Graphs

- Currently, there are two execution modes of a mainstream deep learning framework: a static graph mode (GRAPH_MODE) and a dynamic graph mode (PYNATIVE_MODE).
- In static graph mode, when the program is built and executed, the graph structure of the neural network is generated first, and then the computation operations involved in the graph are performed. Therefore, in the static graph mode, the compiler can achieve better execution performance by using technologies such as graph optimization, which facilitates large-scale deployment and cross-platform running.
- In the dynamic graph mode, the program is executed line by line according to the code writing sequence. In the forward execution process, the backward execution graph is dynamically generated according to the backward propagation principle. In this mode, the compiler delivers the operators in the neural network to the device one by one for computing, enabling users to build and debug the neural network model.

MindSpore Static Graph

- In MindSpore, the static graph mode is also called the Graph mode. You can set the static graph mode by calling the set_context API.
 - set_context (mode=GRAPH_MODE) is the default mode.
- In Graph mode, MindSpore converts the Python source code into an IR (MindIR), optimizes related graphs based on the IR, and executes the optimized graphs on the hardware device.
 - In Graph mode, you need to use the nn.Cell class and write execution code in the construct function, or use the @ms_function modifier.
 - The Graph mode is compiled and optimized based on MindIR.

- The static graph mode is applicable to scenarios where the network is fixed and high performance is required. In static graph mode, the compiler can perform global optimization on graphs based on technologies such as graph optimization and entire computational graph offloading. Therefore, good performance can be obtained when the compiler is executed in static graph mode. However, the execution graph is converted from the source code, therefore, not all Python syntax is supported in static graph mode.
- The nn.Cell class is the base class for building all networks and the basic unit of a network.
- ms_function is a tool provided by MindSpore to accelerate the execution efficiency of dynamic graphs.

MindSpore Static Graph Usage

```
import numpy as np
import mindspore.nn as nn
import mindspore.ops as ops
import mindspore as ms

ms.set_context(mode=ms.GRAPH_MODE, device_target="CPU")

class Net(nn.Cell):
    def __init__(self):
        super(Net, self).__init__()
        self.mul = ops.Mul()
    def construct(self, x, y):
        return self.mul(x, y)

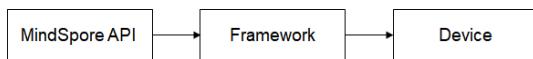
x = ms.Tensor(np.array([1.0, 2.0, 3.0]).astype(np.float32))
y = ms.Tensor(np.array([4.0, 5.0, 6.0]).astype(np.float32))

net = Net()
print(net(x, y))
```

[4. 10. 18.]

MindSpore Dynamic Graph

- In MindSpore, the dynamic graph mode is also called the PyNative mode. You can set the dynamic graph mode by calling the `set_context` API.
 - `set_context(mode=PYNATIVE_MODE)`
- In PyNative mode, you can use complete Python APIs. In addition, when APIs provided by MindSpore are used, the framework executes operator API operations on the selected hardware platform (Ascend/GPU/CPU) and returns the corresponding result.



- The front-end Python API is called to the framework layer and finally computed on the corresponding hardware device.
- The forward propagation process in PyNative mode is executed based on the Python syntax.
- Extension: In PyNative, backward propagation is implemented based on tensors. During the forward propagation process, all operations applied to tensors are recorded and computed backward, and all backward propagation processes are connected to form an overall backward propagation graph (backward graph). Finally, the backward graph is executed on the device and the gradient is computed.

MindSpore Dynamic Graph Usage

```
import numpy as np
import mindspore.nn as nn
import mindspore as ms
import mindspore.ops as ops

ms.set_context(mode=ms.PYNATIVE_MODE,
device_target="CPU")
x = ms.Tensor(np.ones([1, 3, 3, 4]).astype(np.float32)) # Create
a tensor.
y = ms.Tensor(np.ones([1, 3, 3, 4]).astype(np.float32))
output = ops.add (x, y) # Add
print(output.asnumpy())
```



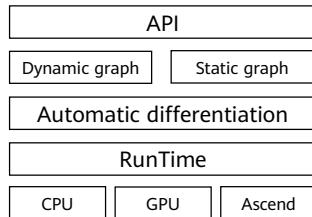
```
[[[[2. 2. 2. 2.]
[2. 2. 2. 2.]
[2. 2. 2. 2.]]]
```

```
[[2. 2. 2. 2.]
[2. 2. 2. 2.]
[2. 2. 2. 2.]]]
```

```
[[2. 2. 2. 2.]
[2. 2. 2. 2.]
[2. 2. 2. 2.]]]]
```

Unification of Dynamic and Static Graphs

- Currently, dynamic and static graphs are supported in the industry. Dynamic graphs are executed through explanation, with dynamic syntax affinity and flexible expression. Static graphs are executed through just in time (JIT) build, which focuses on static syntax and has many syntax constraints. The build process of the dynamic graph is different from that of the static graph. As a result, the syntax constraints are also different.
- For dynamic and static graph modes, MindSpore first unifies the API expression and uses the same APIs in the two modes. Then, it unifies the underlying differentiation mechanism of dynamic and static graphs.



- In MindSpore, you can control the input parameter mode to set the dynamic and static graph mode. However, the static graph has restrictions on the Python syntax. Therefore, you need to comply with the syntax restrictions of the static graph so that the static graph can be correctly used for execution.

MindIR

- An intermediate representation (IR) is a representation of a program between the source and target languages, which facilitates program analysis and optimization for the compiler.
- MindSpore IR (MindIR) is a function-style IR based on graph representation. Its core purpose is to serve automatic differential transformation, which uses semantics close to ANF functional IR.
 - MindIR can help you implement multiple deployments in one training session and implement device-cloud interconnection.

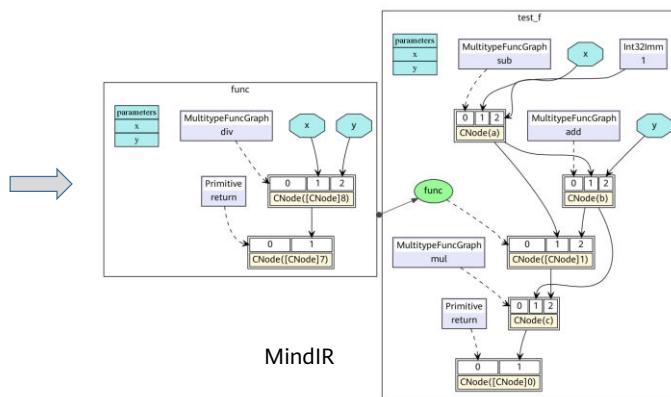
- MindIR is an intermediate representation provided by MindSpore. It can help you implement multiple deployments in one training session and implement device-cloud interconnection. Simply put, you can train and generate MindIR models on the Ascend, GPU, and CPU processors, and then quickly deploy the models on mobile phones and inference servers to experience the full-scenario capabilities of MindSpore.
- MindIR eliminates model differences between different backends through unified operator IR definitions. You can perform collaborative tasks on different platforms (device, edge, and cloud) based on the same model file.

Example for MindIR

```
def func(x, y):
    return x / y

@ms_function
def test_f(x, y):
    a = x - 1
    b = a + y
    c = b * func(a, b)
    return c
```

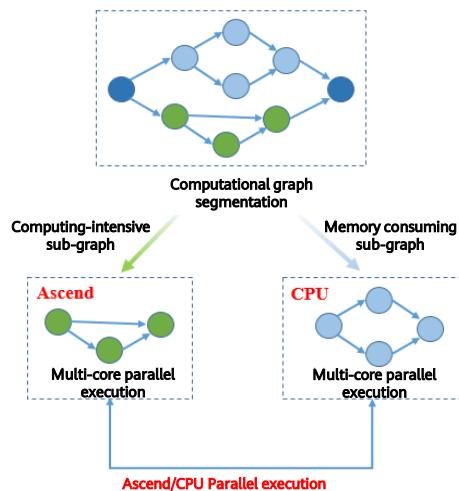
Python code



- Opens MindIR, which is a functional IR based on graph representation. It defines the extensible graph structure and the operator IRs, and stores the basic data structures of MindSpore, including computational graphs and parameter weights.

Heterogeneous Parallel Training

- Through the heterogeneous parallel training method, the operators that have high memory consumption or are suitable for CPU logical processing, are segmented to the CPU sub-graph. Operators that have low memory consumption are segmented to the hardware accelerator sub-graph. Different sub-graphs perform network training under the framework, and this method allows independent sub-graphs in different hardware to be executed in parallel.



37 Huawei Confidential



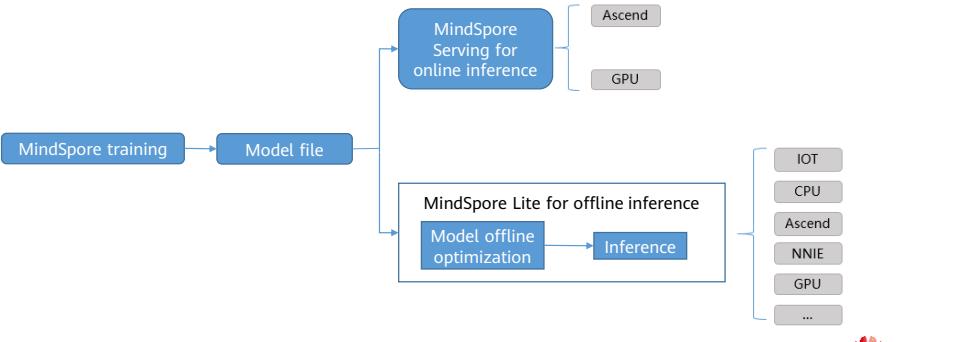
- Computing process:
 - User sets the backend device for the network execution.
 - User sets the backend device for specific operator.
 - The framework segments the computational graph according to the operator feature.
 - The framework schedules different backend to execute sub-graphs.
- The typical heterogeneous parallel computing scenarios are as follows: heterogeneous optimizer, heterogeneous graph embedding, and heterogeneous PS.

Contents

1. AI Framework Development
2. MindSpore
3. MindSpore Architecture and Features
- 4. MindSpore Development Components**
5. AI Application Development Process

MindSpore Inference Deployment Process

- The model file trained by MindSpore can be executed in cloud services through MindSpore Serving and executed on servers and devices through MindSpore Lite. Besides, MindSpore Lite supports offline model optimization by using the convert tool to build a lightweight inference framework and achieve high-performance model execution.



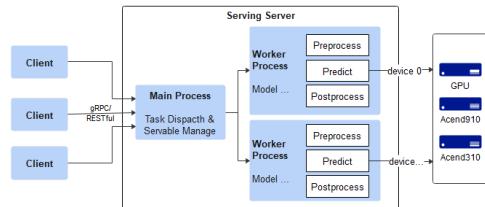
40 Huawei Confidential



- MindSpore abstracts the algorithm application process and provides a set of unified algorithm APIs. Therefore, the programming code of the network model can be consistent on different hardware. If the same model file is loaded at the same time, inference can be effectively executed on different hardware supported by MindSpore. Considering that a large number of developers use C++ or C to program codes, the C++ inference programming interface is provided. The type of C++ programming interface for inference is similar to that of the Python interface.

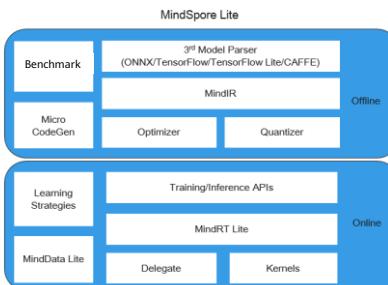
MindSpore Serving

- MindSpore Serving is a lightweight and high-performance service module that helps MindSpore developers efficiently deploy online inference services in the production environment.
- Three ways are available to access the MindSpore Serving services:
 - Call the gRPC API to access MindSpore Serving services.
 - Call the RESTful API to access MindSpore Serving services.
 - The Servable of MindSpore Serving provides the inference services:
 - From a single model.
 - From the combination of multiple models.



MindSpore Lite

- MindSpore Lite is an ultra-fast, intelligent, and simplified AI engine that enables intelligent applications in all scenarios, provides E2E solutions for users, and helps users enable AI capabilities.
- MindSpore Lite consists of two modules: online and offline.



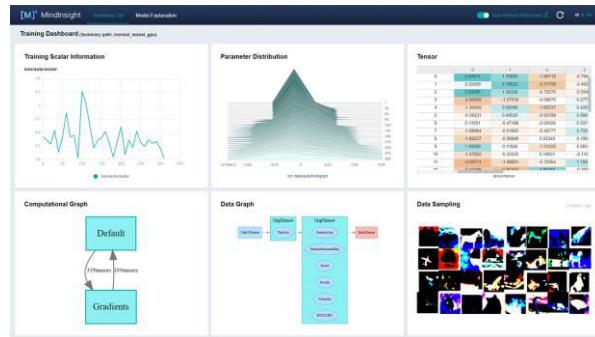
42 Huawei Confidential



- Offline Module:
 - 3rd Model Parser: converts third-party models into the unified MindIR model. Third-party models include TensorFlow, TensorFlow Lite, Caffe 1.0, and ONNX models.
 - MindIR: unified device-cloud IR of MindSpore.
 - Optimizer: optimizes graphs based on IR, such as operator fusion and constant folding.
 - Quantizer: post-training quantization module, which supports post-training quantization methods such as weight quantization and activation value quantization.
 - Benchmark: a toolset for testing performance and debugging precision.
 - Micro CodeGen: a tool used to directly compile models into executable files in IoT scenarios.
- Online module:
 - Training/Inference APIs: unified device-cloud C++/Java training and inference APIs.
 - MindRT Lite: supports asynchronous execution during lightweight online runtime.
 - MindData Lite: used for data processing on the device side.
 - Delegate: agent used to connect to professional AI hardware engines.
 - Kernels: built-in high-performance operator library, providing CPU, GPU, and NPU operators.
 - Learning Strategies: device-side learning strategy, such as the transfer learning.

MindInsight

- MindInsight is a visualized debugging and optimization tool of MindSpore. It can visualize the training process, optimize model performance, and improve debugging accuracy. Besides, MindInsight also provides a command line for users to easily search for hyperparameters and migrate models.



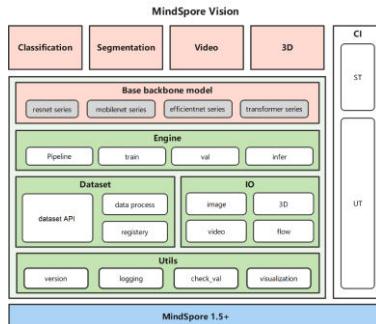
43 Huawei Confidential



- <https://www.mindspore.cn/mindinsight/docs/zh-CN/r1.7/index.html>

MindSpore Vision

- MindSpore Vision is an open source computer vision research tool library based on the MindSpore framework. The tasks executed through the tool library include classification, segmentation (under development), video (under development), and 3D (under development). MindSpore Vision aims to provide easy-to-use APIs to help users redevelop existing classic models and develop their own models.



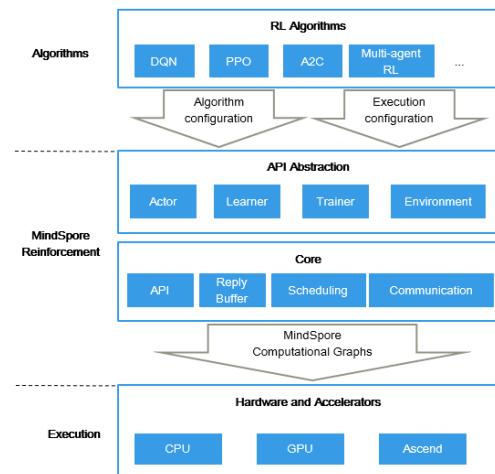
44 Huawei Confidential



- Advanced functions provided by MindSpore Vision are as follows:
 - Classification: deep neural network workflow built on the engine system.
 - Backbone network: basic backbone network of models such as ResNet and MobileNet.
 - Engine: callback function used for model training.
 - Dataset: provides various dataset APIs for various fields.
 - Tools: provides various visualization, input, and output APIs.

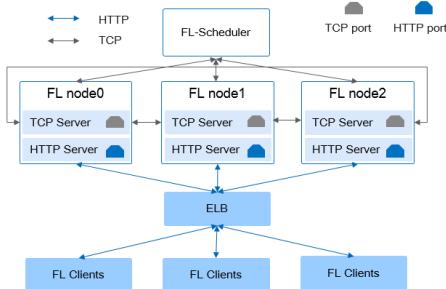
MindSpore Reinforcement

- MindSpore Reinforcement is an open-source reinforcement learning framework that supports distributed training of agents using reinforcement learning algorithms.
 - MindSpore Reinforcement provides simple and abstract APIs for writing reinforcement learning algorithms. It decouples algorithms from specific deployment and execution processes, including accelerator usage, parallelism degree, and cross-node computing scheduling.



MindSpore Federated

- Federated learning is an encrypted distributed machine learning technology that allows different participants to build AI models without sharing local data. MindSpore Federated is an open-source federated learning framework that supports the commercial deployment of tens of millions of stateless devices.



Other components

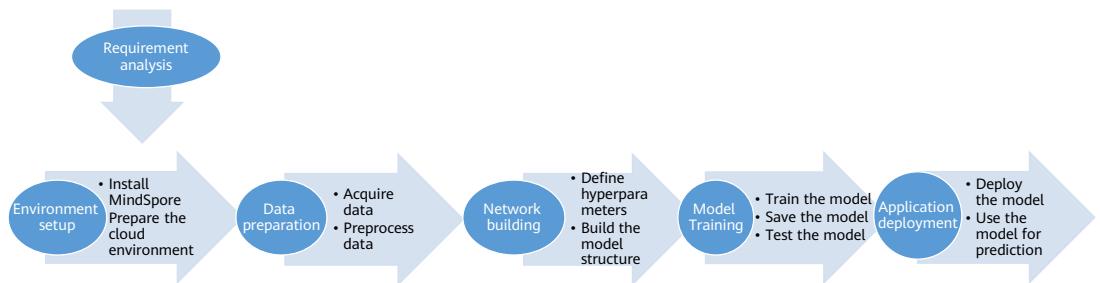
- MindQuantum: An open-source deep learning framework of MindSpore and a common quantum computing framework developed by HiQ. It supports training and inference of multiple quantum neural networks.
- MindSpore Probability: MindSpore probabilistic programming provides a framework that seamlessly integrates Bayesian learning and deep learning. It aims to provide users with a complete probability learning library for establishing probabilistic models and applying Bayesian inference.
- MindSpore Golden Stick: model compression algorithm set
-

- MindSpore Golden Stick not only provides users with various model compression algorithms, but also offers a unified and simple development environment for users, reducing the cost of applying model compression algorithms.

Contents

1. AI Framework Development
2. MindSpore
3. MindSpore Features
4. MindSpore Development Components
- 5. AI Application Development Process**

AI Application Development Process



ResNet50 Image Classification Application

- Image classification is the most basic computer vision application and belongs to the supervised learning category. For example, we can determine the category to which an image (such as an image of a cat, a dog, an airplane, or a car) belongs. This application describes how to use the ResNet50 network to classify flower datasets.

What kind of flower is this?



Environment Setup

- Select the specifications of your operating environment to obtain the installation commands.

- For example, install MindSpore 1.7.1 in Linux or Python 3.7 in pip mode:

Version 1.9.0 1.8.1 2.0.0 Nightly

Hardware Platform Ascend 910 Ascend 310 GPU CUDA 10.1 GPU CUDA 11.1 CPU

Operating System Linux-aarch64 Linux-x86_64 Windows-x64 MacOS-aarch64 MacOS-x86_64

Programming Language Python 3.7 Python 3.8 Python 3.9

Installation Mode Pip Conda Source Docker Binary

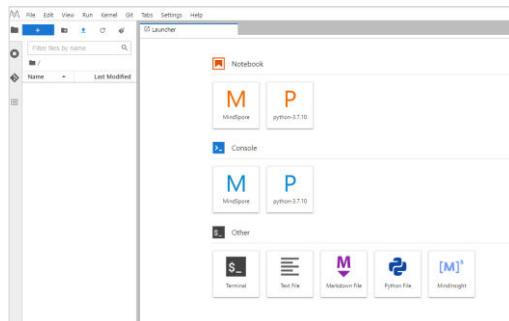
Commands

```
pip install https://ms-release.oss-cn-north-4.myhuaweicloud.com/1.9.0/MindSpore/ascend/aarch64/mindspore_ascend-1.9.0-cp37-cp37m-linux_aarch64.whl --trusted-host ms-release.oss-cn-north-4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple
# Refer to the following installation guide, ensure that the installation dependency and environment variables are correctly configured.
```

<https://www.mindspore.cn/install>

Cloud Environment (optional)

- If no environment is available, you can use Huawei Cloud ModelArts for development, model training, and deployment.
 - ModelArts helps you quickly create and deploy models and manage the AI development lifecycle.



Data Preparation

- Photos of five kinds of flowers (open-source data) is used for training: daisy (633 photos), dandelion (898 photos), roses (641 photos), sunflowers (699 photos), tulips (799 photos). These 3670 photos are respectively saved in 5 folders. The file structure of the photos contains two parts: flower_photos_train and flower_photos_test.

```
flower_photos_train:  
├── daisy  
├── dandelion  
├── roses  
├── sunflowers  
└── tulips  
└── LICENSE.txt  
  
flower_photos_test:  
├── daisy  
├── dandelion  
├── roses  
├── sunflowers  
└── tulips  
└── LICENSE.txt
```

File structure



Data Preprocessing

- To improve the model accuracy and ensure the model generalization capability, operations such as data enhancement and standardization are performed before the data is used to train the model.
- You can load data sets by defining the `read_dataset` function, which has the following functions:
 - Read the dataset.
 - Define parameters required for data augmentation and processing.
 - Generate corresponding data augmentation operations according to the parameters.
 - Use the `map` function to apply data operations to the dataset.
 - Process the generated dataset.
 - Display the processed data as an example.

- You do not need to process the datasets too much because the dataset is open-source data. However, if the data is business data, you need to clean and integrate data.

APIs For Dataset Processing

- `mindspore.dataset` is used to load and process various common datasets.
- `mindspore.dataset.vision` is used to enhance image data.

```
from mindspore.dataset.vision import c_transforms as vision
vision.Normalize(mean, std) # Normalize the input image with the mean and standard deviation.
# std/mean The list and tuple of image composed of the mean and standard deviation. The mean and standard deviation should range from 0.0 to 255.0.
vision.HWC2CHW() # Convert the shape of the input image from <H, W, C> to <C, H, W>. The channel of input image should be 3.
vision.CenterCrop(size) # Crop the central area of the input image. If the size of the input image is less than that of the output image, the edge of the input image is padded with 0 pixel before cropping.
```

- `mindspore.dataset.text` is used to enhance text data.
- `mindspore.dataset.audio` is used to enhance audio data.
- `mindspore.dataset.transforms` is the general data enhancement module.

- `mindspore.dataset.vision` is used to enhance image data and includes the `c_transforms` and `py_transforms` submodules. `c_transforms` is a high-performance module for image enhancement developed by using C++ OpenCv. `py_transforms` is an image enhancement module developed by using Python Pillow.
- `mindspore.dataset.text` is used to enhance text data, including `transforms` and `utils` sub-modules.
 - `transforms` is a high-performance module for common text data enhancement.
 - `utils` provides some text processing tools and methods.
- `mindspore.dataset.audio` is used to enhance audio data, including `transforms` and `utils` submodules. `transforms` is a high-performance module for common audio data enhancement. `utils` provides some audio processing tools and methods.

Code Implementation for Data Preprocessing

```
def read_data(path,config,usage="train"):
    dataset = ds.ImageFolderDataset(path, class_indexing={'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4})
    decode_op = vision.Decode() # Operator for image decoding
    normalize_op = vision.Normalize(mean=[cfg._R_MEAN, cfg._G_MEAN, cfg._B_MEAN], std=[cfg._R_STD, cfg._G_STD, cfg._B_STD]) #
    Operator for image normalization
    resize_op = vision.Resize(cgf._RESIZE_SIDE_MIN) # Operator for image resizing
    center_crop_op = vision.CenterCrop((cfg.HEIGHT, cfg.WIDTH)) # Operator for image cropping
    horizontal_flip_op = vision.RandomHorizontalFlip() # Operator for image random horizontal flipping
    channelswap_op = vision.HWC2CHW() # Operator for image channel quantity conversion
    # Operator for random image cropping, decoding, encoding, and resizing
    random_crop_decode_resize_op = vision.RandomCropDecodeResize((cfg.HEIGHT, cfg.WIDTH), (0.5, 1.0), (1.0, 1.0),
    max_attempts=100)
    ....
    ....
    dataset = dataset.repeat(1) # Data enhancement
    dataset.map_model = 4
    return dataset
```

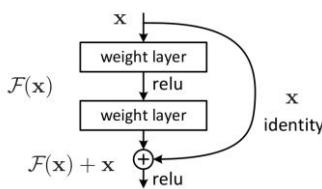
Hyperparameters Definition

- Models contain both parameters and hyperparameters. Hyperparameters enable the model to learn the optimal parameters.

```
cfg = edict({  
    '_R_MEAN': 123.68, # Average value of CIFAR10  
    '_G_MEAN': 116.78,  
    '_B_MEAN': 103.94,  
    '_R_STD': 1, # Customized standard deviation  
    '_G_STD': 1,  
    '_B_STD':1,  
    '_RESIZE_SIDE_MIN': 256, # Minimum resize value for image enhancement  
    '_RESIZE_SIDE_MAX': 512,  
  
    'batch_size': 32, # Batch size  
    'num_class': 5, # Number of classes  
    'epoch_size': 5, # Number of training times  
    'loss_scale_num':1024,  
})
```

Introduction to ResNet

- ResNet-50 was proposed by He Kaiming of Microsoft Research in 2015 and won the 2015 ILSVRC.
- The residual network is a main tag of ResNet with which the degradation problem can be effectively alleviated and a deeper network can be designed.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112					
				3×3 max pool, stride 2		
conv2.x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3.x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$
conv4.x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5.x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Introduction to Network Structure APIs (1)

- `mindspore.nn`: the cell of neural network which provides predefined building blocks or compute units in a neural network.
 - `mindspore.nn.Cell`: the Cell class of MindSpore is the base class for building all networks and the basic unit of a network.
 - `mindspore.nn.LossBase`: specifies the base class of the loss function.
- Neural network structure APIs:
 - `mindspore.nn.Dense(in_channels, out_channels, weight_init='normal', bias_init='zeros', has_bias=True, activation=None)`: specifies the fully connected layer.
 - `in_channels` (int): specifies the spatial dimension of the tensor input to the Dense layer.
 - `out_channels` (int): specifies the spatial dimension of the tensor output from the Dense layer.
 - `weight_init`: specifies the weight parameter initialization method.

- In MindSpore, user-defined network structures and loss functions can be quickly created by inheriting corresponding parent classes.
- For details about how to use functions and classes, see the MindSpore document at <https://www.mindspore.cn/docs/zh-CN/r1.7/index.html>.

Introduction to Network Structure APIs (2)

- `mindspore.nn.Conv2d`: (`in_channels`, `out_channels`, `kernel_size`, `stride=1`, `pad_mode= "same"`, `padding=0`, `dilation=1`, `group=1`, `has_bias=False`, `weight_init= "normal"`,
`bias_init="zeros"`, `data_format="NCHW"`): two-dimensional convolutional layer.
 - `kernel_size`: specifies the height and width of the two-dimensional convolution kernel.
 - `stride`: specifies the stride of the two-dimensional convolution kernel.
 - `pad_mode`: specifies the padding mode. The value can be **same**, **valid**, or **pad**. The default value is **same**.
- `mindspore.nn.MaxPool2d` (`kernel_size=1`, `stride=1`, `pad_mode= 'valid'`,
`data_format='NCHW'`): specifies the two-dimensional maximum pooling layer.

Introduction to Network Structure APIs (3)

- `mindspore.nn.RNN(*args, **kwargs)`: specifies the recurrent neural network (RNN) layer, whose activation function is tanh and relu.
 - `input_size` (int): specifies the feature vector dimension input to the input layer.
 - `hidden_size` (int): specifies the feature vector dimension output from the hidden layer.
 - `num_layers` (int): specifies the number of stacking RNN layers. The default value is 1.
- `mindspore.nn.Dropout(keep_prob=0.5, dtype=mstype.float32)`: automatically set some neurons output to 0 during training based on the dropout probability $1 - \text{keep_prob}$.
 - `keep_prob` (float): specifies the input neuron retention rate, ranging from 0 to 1.
- `mindspore.nn.ReLU`: specifies the activation function of the rectified linear unit.
- `mindspore.nn.Softmax (axis=-1)`: specifies the Softmax activation function.
 - `axis`: specifies the axis of the Softmax operation.

MindSpore Network Building

- When a neural network is required, you need to inherit the Cell class and overwrite the `__init__` and `construct` methods.

```
# For details about ResNet code implementation, see the experiment guide.  
class ResNet(nn.Cell):  
    def __init__(self, *args, **kwargs):  
        super(ResNet, self).__init__()  
        ....  
        Layers in the network  
        ....  
    def construct(self, x):  
        ....  
        Network structure  
        ....
```

- In MindSpore, the basic unit of the neural network is `nn.Cell`. All the models or neural network layers must inherit this base class. As a member function of this base class, `construct` defines the calculation logic to be executed and must be rewritten in all inherited classes.

Model Training

- Start model training after data preprocessing and network building.
- There are two ways to train the model:
 - Start training from scratch based on your own dataset. This method applies to scenarios where the data volume is large and the training resources are robust.
 - Start training based on the trained model and perform fine-tuning training on own dataset. This method applies to scenarios where the data volume is small (applied in the current experiment).
 - Pre-trained model: ResNet model file trained on the ImageNet dataset.
 - Modify the parameters at the last layer of the pre-trained model parameters. (The model is pre-trained on the ImageNet dataset to classify 1001 types. However, the current experiment is to classify the five types of flowers.)
 - Training is performed based on its self-owned data set.
- Parameter tuning: During training, you can tune the hyperparameter combinations.
- For details about the code, see the experiment guide.

Introduction to Training APIs

- `mindspore.Model(network, loss_fn=None, optimizer=None, metrics=None, eval_network=None, eval_indexes=None, amp_level="O0", boost_level="O0", **kwargs)`: The model encapsulates instances that can be trained or inferred based on the parameters input by users.
 - `network` (Cell): used for training and inference of the neural network.
 - `loss_fn` (Cell): specifies the loss function.
 - `optimizer` (Cell): an optimizer used to update network weights.
 - `metrics` (Union[dict, set]): a set of evaluation functions used for model evaluation.

Model Saving and Loading

- After the network training is complete, save the network model as a file. There are two types of APIs for saving models:

- One is to simply save the network model before and after training.

```
import mindspore as ms
Use the save_checkpoint provided by MindSpore to save the model, pass it to the network, and save the path.
# net indicates a defined network model, which is used before or after training.
ms.save_checkpoint(network, "./MyNet.ckpt")
```

- You can also save the interface during network model training. MindSpore automatically saves the number of epochs and number of steps set during training. That is, the intermediate weight parameters generated during the model training process are also saved to facilitate network fine-tuning and stop training.

```
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig
# Set the value of epoch_num.
epoch_num = 5
# Set model saving parameters.
config_ck = CheckpointConfig(save_checkpoint_steps=1875, keep_checkpoint_max=10)
# Apply the model saving policy.
ckpoint = ModelCheckpoint(prefix="lenet", directory="./lenet", config=config_ck) model.train(epoch_num, dataset_train, callbacks=[ckpoint])
```

- During network model training, we want to save the intermediate and final results for fine-tuning and subsequent model deployment and inference. If the model is too large, you need to save the model while training.
- In the second type of model saving, you need to initialize a CheckpointConfig class object to set the saving policy.
 - save_checkpoint_steps indicates the interval (in steps) for saving the checkpoint file.
 - keep_checkpoint_max indicates the maximum number of checkpoint files that can be retained.
 - prefix indicates the prefix of the generated checkpoint file.
 - directory indicates the directory for storing files.
- Create a ModelCheckpoint object and pass it to the model.train method. Then the checkpoint function can be used during training.

Model Loading and Prediction (1)

- Use the trained model weights to call `model.predict()` to test the test data.
 - After the training is complete, call `model.predict` for prediction.
 - Perform prediction after the weight file is loaded.
 - To load the model weight, you need to create an instance of the same model and then use the `load_checkpoint` and `load_param_into_net` methods to load parameters.

Model Loading and Prediction (2)

- The load_checkpoint method loads the network parameters in the parameter file to the param_dict dictionary.
- The load_param_into_net method loads the parameters in the param_dict dictionary to the network or optimizer. After the loading, parameters in the network are stored by the checkpoint.

```
from mindspore import load_checkpoint, load_param_into_net
# Save the model parameters to the parameter dictionary. The model parameters saved during
the training are loaded.
param_dict = load_checkpoint("./ResNet_1875ckpt")
# Redefine a ResNet neural network.
net = ResNet(num_classes=5, pretrained=False)
# Load parameters to the network. load_param_into_net(net, param_dict)
# Redefine the optimizer function.
net_opt = nn.Momentum(net.trainable_params(), learning_rate=0.01, momentum=0.9)
model = Model(net, loss_fn=net_loss, optimizer=net_opt, metrics={"accuracy"})
```

Model Deployment

- Deployment on device (using mobile phones as an example):
 - Convert the CKPT file to the MindIR file format, and then convert MindIR file format to the MindSpore Lite recognizable file on the Android phone (MS model file).
 - Deploy the app APK on the mobile phone, that is, download a MindSpore Vision suite Android APK.
 - Import the MS model file to the mobile phone.
- Deployment on cloud:
 - Deploy the model in cloud services through MindSpore Serving.
 - Quickly deploy model on the cloud using ModelArts.
- Deployment on edge:
 - Deploy the model based on the AscendCL and Atlas computing platforms.

- Due to insufficient computing power on devices, the model size and calculation workload need to be considered.

Summary

- This course describes the knowledge related to the AI development framework, including the basic knowledge of the common AI framework, MindSpore framework basics, and AI application development process.

Recommendations

- Official MindSpore website
 - <https://www.mindspore.cn/>

Thank you.

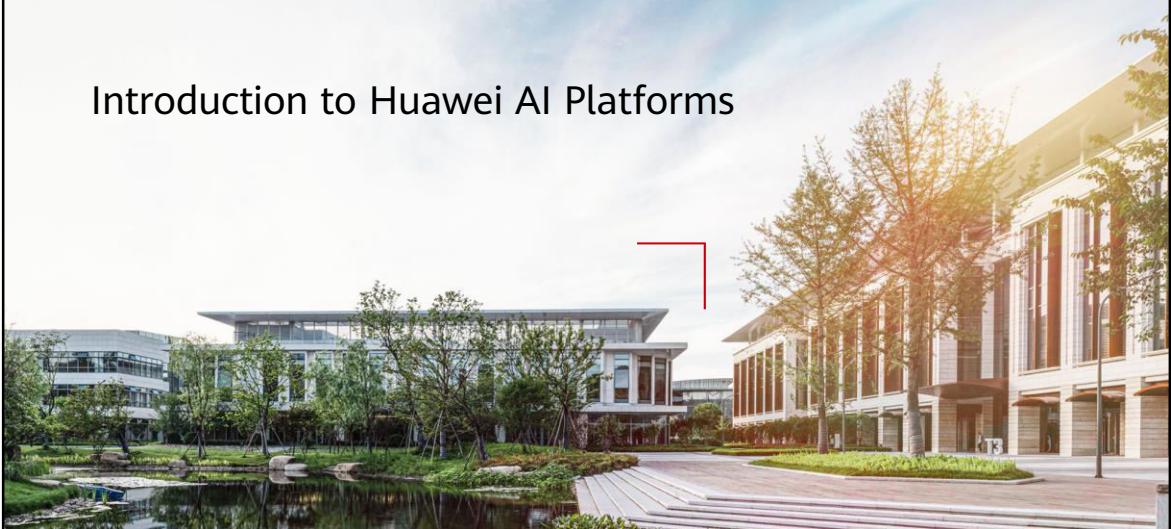
Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright © 2023 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.



Introduction to Huawei AI Platforms



Foreword

- This chapter describes Huawei's development platforms in the AI field, including the Ascend computing platform based on Da Vinci architecture, Huawei Cloud EI platform based on cloud services, and device AI platform that provides AI capabilities for devices.

Objectives

Upon completion of this course, you will understand:

- Software and hardware architectures of the Ascend processor.
- Features and application scenarios of the Atlas AI computing platform.
- AI services and application development process of Huawei Cloud EI.
- Features of HarmonyOS, HMS Core, ML Kit, HiAI, and MindSpore Lite.

Contents

1. Huawei Ascend Computing Platform
 - AI Processor Overview
 - Hardware Architecture of the Ascend Processor
 - Software Architecture of the Ascend Processor
 - Huawei Atlas AI Computing Platform
 - Atlas Application in the Industry
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms

Overview and Objectives

- This section describes the concept of AI processor, software and hardware architectures of the Ascend processor, and Atlas series overview and application scenarios.

AI Processor Definition

- More than 99% of operations in AI tasks are matrix operations.
- **AI processor:**
 - AI processor, in a broad sense, is a module dedicated to processing a large number of computing tasks in AI applications. In this sense, all processors oriented to the AI field are AI processors.
 - AI processor, in a narrow sense, is a type of processor specially designed for AI algorithm acceleration, which is also called an AI accelerator.
 - "An AI accelerator is a class of specialized hardware accelerator or computer system designed to accelerate artificial intelligence and machine learning applications, including artificial neural networks and machine vision. Typical applications include algorithms for robotics, internet of things, and other data-intensive or sensor-driven tasks. They are often manycore designs and generally focus on low-precision arithmetic, novel dataflow architectures or in-memory computing capability."

-- Wikipedia

- Current AI models contain a large number of matrix operations. AI processors are optimized for matrix multiplication and equipped with various processing units to efficiently perform matrix operations.

AI Processor Types

- AI processors are classified into training processors and inference processors according to their application scenarios:
 - During training, a simple deep neural network (DNN) model is trained by inputting a large amount of data or using an unsupervised learning method like reinforcement learning. The training process involves massive training data, complex DNN structures, and large computing amounts, posing very high performance requirements on computing capability, accuracy, and scalability of a processor. AI processors used for training can be seen in Huawei Atlas 900 clusters and NVIDIA GPU clusters.
 - Inference is to use a trained model and new data to infer various conclusions. Facial authentication is an example of inference, where the device uses a DNN model to determine whether the face belongs to the device owner. Although the calculation workload of inference is much less than that of training, a large number of matrix operations are still required. CPUs, NPUs, GPUs, and FPGAs can be used during the inference process.

- Training processors are more complex due to more design considerations. The precision is usually FP32 or FP16.
- Inference processors are designed with fewer factors in mind. The precision is usually INT8, lower than that of training processors.

Application Fields of AI Processors



Cloud-based training

- **Processor features:** high power consumption, high throughput, high accuracy, distributed deployment, scalability, large memory, and high bandwidth
- **Application:** cloud, HPC, and data centers



Cloud-based inference

- **Processor features:** low power consumption, high throughput, high accuracy, distributed deployment, scalability, and low latency
- **Application:** cloud, HPC, and data centers

Edge computing



- **Processor features:** low power consumption, low latency, independent deployment or co-deployment with other devices, virtualization of multiple end users, and small rack space
- **Application:** smart manufacturing, smart home, smart transportation, and smart finance

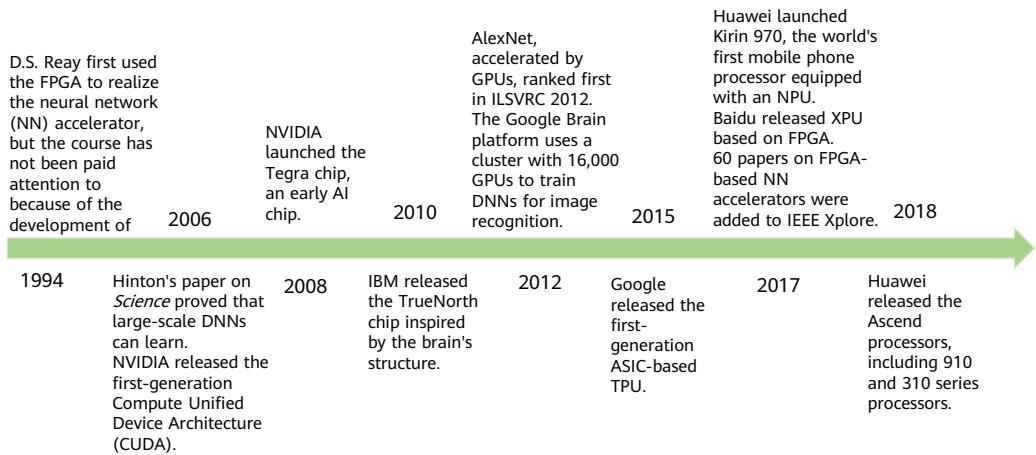
End devices



- **Processor features:** ultra-low power consumption, high energy efficiency, inference orientation, low throughput, low latency, and cost-sensitivity
- **Application:** consumer electronics in diversified product forms, and IoT

- Cloud processors are deployed in equipment rooms of adequate environments. Edge computing processors are usually deployed outdoors and need to adapt to high- and low-temperature environments. For end devices, power consumption and costs are the main consideration.

AI Processor Development Process



9 Huawei Confidential



Currently, the mainstream processor types are GPU, FPGA, and ASIC. The latter two have derived a series of processors such as XPU and TPU.

AI Processor Technology Directions

Technical Architecture	Customization Level	Programmability	Computing Power	Price	Advantages	Disadvantages	Application Scenarios
GPU	General-purpose	Not programmable	Medium	High	GPUs are universal, sophistically designed and manufactured, and suitable for large-scale parallel computing.	The parallel computing capability cannot be fully utilized at the inference end.	Advanced complex algorithms and universal AI platforms
FPGA	Semi-customized	High	High	Medium	FPGAs can be flexibly configured to adapt to algorithm iterations. The average performance is high, the power consumption is low, and the development period is short.	The unit price for mass production is high, the peak computing capability is low, and hardware programming is difficult.	Various specific industries
ASIC	Fully customized	Low	High	Low	Algorithms of ASICs are fixed to achieve optimal performance and energy efficiency. The average performance is high, the power consumption is low, the size is small. The cost for mass production is low.	High initial investment, long R&D time, and high technical risks.	Special scenarios with dedicated intelligent algorithm software
Brain-inspired processor	Human brain simulation	Not programmable	High	-	Low power consumption, high communication efficiency, and strong cognitive ability.	It is still in the exploration phase.	Various specific industries

- All these technical directions are being actively developed.

General Computing and AI Computing Build Diversified Computing Together

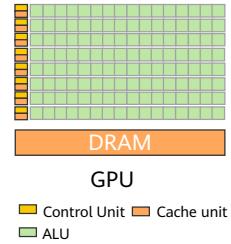
		Hardware Structure	Computing Features	Application Scenarios
General computing powered by CPUs	Kunpeng	<p>Control Unit</p> <p>Cache unit</p> <p>ALU</p> <p>Legend: Control unit (yellow), Cache unit (orange), ALU (green)</p>	<p>Suitable for complex logical operations, for example, most general-purpose software.</p> <p>More than 70% transistors are used to build caches and control units. The number of computing cores ranges from several to dozens.</p>	<p>General applications</p> <p>Office, database, and numerical calculation (weather forecast, fluid simulation, and electromagnetic simulation).</p>
AI computing powered by GPUs	NVIDIA AMD	<p>Tensor Core</p> <p>Control unit</p> <p>Cache unit</p> <p>ALU</p> <p>Legend: Control unit (yellow), Cache unit (orange), ALU (green)</p>	<p>Suitable for compute-intensive and high-concurrency tasks with simple logic</p> <p>More than 70% transistors are used to construct computing units, forming thousands or tens of thousands of computing cores.</p>	<p>Specific applications</p> <p>Image recognition: license plate recognition, object recognition, and object detection.</p> <p>Natural language processing (NLP): machine translation and text generation.</p> <p>Language processing: speech recognition and text to speech.</p> <p>Search recommendation, driving assistant, and trend prediction.</p>
AI computing powered by NPUs	Ascend	<p>Da Vinci core</p> <p>Da Vinci core</p> <p>Control unit</p> <p>Cache unit</p> <p>ALU</p> <p>Legend: Control unit (yellow), Cache unit (orange), ALU (green)</p>		HUAWEI

11 Huawei Confidential

- CPUs focus on general computing. GPUs focus on mathematical computing. NPUs focus on matrix operations required by AI computing.

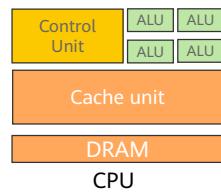
Comparison of CPU and GPU Designs (1)

- GPUs are mainly used for pure computing environments that have unified, large-scale independent data and are not interrupted.
 - Adopt a large-scale parallel computing architecture that consists of thousands of small and efficient cores designed for processing multiple tasks at the same time;
 - Designed for large throughput:
 - A GPU is configured with a large number of ALUs and few caches (serving the threads instead of CPU cores). DRAM accesses are coalesced in the caches, causing latency.
 - Control units coalesce the accesses.
 - The latency problem is masked by a large number of ALUs processing threads in parallel.
 - Good at programs that are compute-intensive and easy to run in parallel.



Comparison of CPU and GPU Designs (2)

- CPUs are used to process different data types in a highly universal manner, and perform logic judgment. In addition, CPUs need to handle a large number of branch directs and interrupts.
 - Consist of several cores optimized for serial processing;
 - Designed for low latency:
 - Powerful ALUs can complete the computation in few clock cycles.
 - A large number of caches reduce the delay.
 - High clock rate.
 - Complex ALUs can reduce the latency of multi-branch programs through branch prediction.
 - For some instructions that depend on previous instruction results, ALUs determine the positions of instructions in the pipeline to implement fast data forwarding.
 - Good at logic control and serial operations.



Evolution of CPU and GPU Designs

- CPU
 - General performance is improved by increasing the number and frequency of cores.
 - AI performance is improved by adding instructions (modifying the architecture):
 - Intel adds the AVX-512 FMA instruction sets to the CISC architecture.
 - ARM adds the Cortex A instruction set to the RISC architecture, which is planned to be continuously upgraded.
 - Intel plans to add vision processing units (VPUs) to CPUs.
- GPU
 - Dedicated AI computing units (Tensor Cores and Matrix Cores) are added.

- Dedicated AI computing modules can be added to both CPUs and GPUs to improve AI computing power.

Comparison of FPGA and ASIC

	FPGA	ASIC
Computing speed	Low due to inevitable redundancy caused by universality of the architecture and latency between different structures.	High because there is no special requirement on the architecture and specific modules can be placed close to each other to reduce latency.
Processor size	Large (when the functions are the same)	Small (when the functions are the same)
Power consumption	High (under the same process conditions)	Low (under the same process conditions)
Cost	The R&D risk and cost is low. The cost is mainly attributed to production.	The tool development and tape-out processes may incur significant cost because the hardware must be determined before production.
Running process	It takes time to load the configuration to the storage device.	Programs can run immediately.
Product positioning	Suitable for products that require rapid market occupation or flexible feature design.	Suitable for products with large-scale design or mature technologies, such as consumer electronics.
Development direction	Large capacity, low voltage, low power consumption, and SoC.	Larger scale, intellectual property reuse, and SoC.

- The two apply to different areas and are not competing with each other.

FPGA and ASIC Processor Evolution

- FPGA
 - Xilinx adds the hardware/software programmable engine with many AI cores to the processor.
 - Intel upgrades the DSP module in the traditional FPGA.
- ASIC
 - Companies are deploying ASICs, including TPUs, NPUs, and VPUs.

Da Vinci Architecture: Born for Ultimate Efficiency of AI Computing

The proportion of Tensor Cores is low because the GPU needs to support both image rendering and AI computing.

First-generation Tensor Core

V100 – Volta architecture (2017)

4x4x4

64 operations in a clock cycle

Third-generation Tensor Core

A100 – Ampere architecture (2020)

8x8x4

256 operations in a clock cycle

Fourth-generation Tensor Core

H100 – Hopper architecture (2022)

16x8x4

512 operations in a clock cycle

Ascend NPU is specially designed for AI computing with a high proportion of Cube units.

Ascend NPU – Da Vinci architecture (2018)

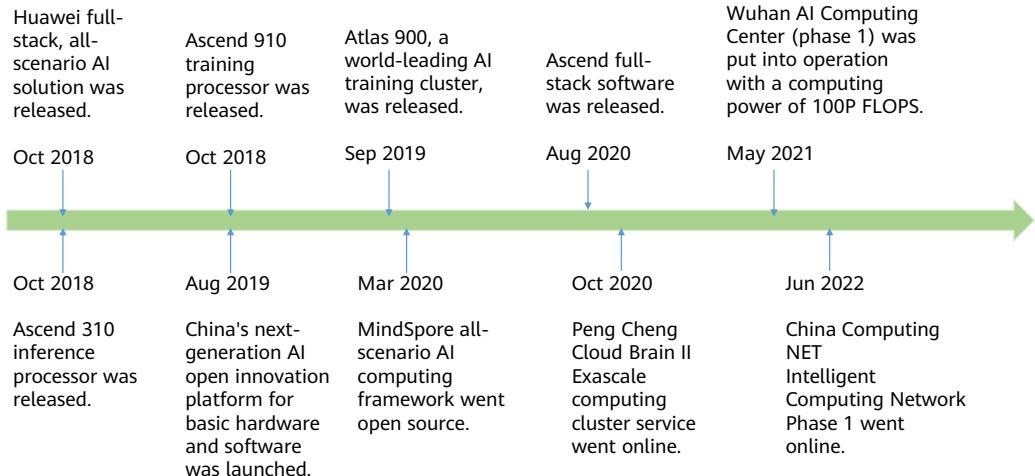
The proportion of Cube computing units is about 99.2%.

Cube=16x16xN
N=1/2/4/8/16

A maximum of 4096 operations in a clock cycle

- The core of the Da Vinci architecture is the Cube computing unit.

Huawei's Investment in the AI Industry



18 Huawei Confidential



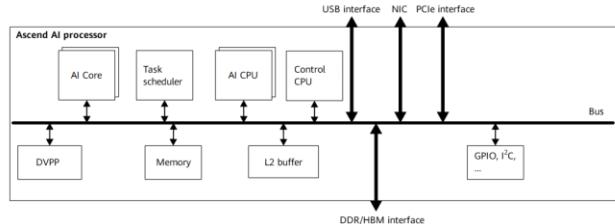
- At the World Artificial Intelligence Conference 2019, the Ministry of Science and Technology of China announced that Huawei was tasked with building China's next-generation AI open innovation platform for basic hardware and software. Huawei provides full-process and inclusive basic platform services oriented to research and development of AI applications for various industries, startups, universities, and institutes.

Contents

1. Huawei Ascend Computing Platform
 - AI Processor Overview
 - Hardware Architecture of the Ascend Processor
 - Software Architecture of the Ascend Processor
 - Huawei Atlas AI Computing Platform
 - Atlas Application in the Industry
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms

Logical Architecture of the Ascend AI Processor

- The Ascend AI processor consists of:
 - Processor system control CPU (control CPU)
 - AI computing engine (AI Cores and AI CPUs)
 - Multi-layer system-on-chip (SoC) cache or buffer
 - Digital vision pre-processing (DVPP) module



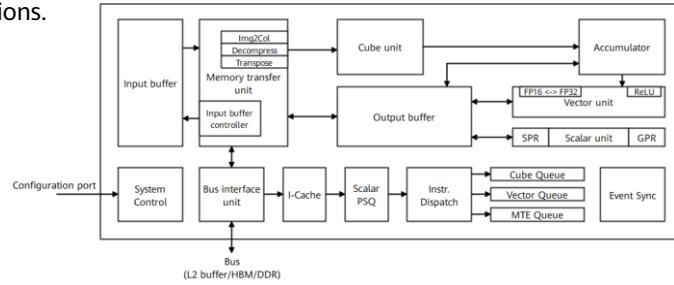
Ascend AI Computing Engine - Da Vinci Architecture

- One of the four major structures of the Ascend AI processor is the AI computing engine, which consists of AI Cores (Da Vinci architecture) and AI CPUs. Da Vinci architecture, an architecture dedicated to improving AI computing power, is the core of the Ascend AI computing engine and AI processor.

- The AI Core is responsible for efficient matrix computing, and the AI CPU is responsible for general computing.

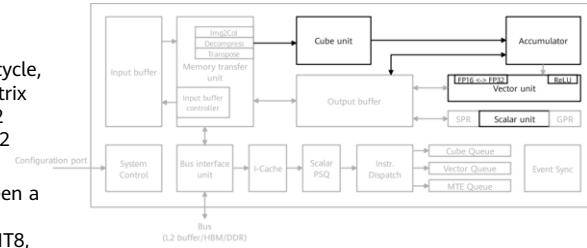
Da Vinci Architecture (AI Core)

- Main components of the Da Vinci architecture:
 - Compute units: Cube unit, Vector unit, and Scalar unit
 - Storage system: on-chip storage units of AI Core and the data paths
 - Control units: the brain of AI Core, responsible for AI Core runtime control with instructions.



Da Vinci Architecture (AI Core) — Computing Unit

- The three basic compute resources, Cube, Vector, and Scalar Units, perform computations related to matrices, vectors, and scalars, respectively.
 - The Cube unit works with the accumulator to perform matrix-related operations. Per clock cycle, it completes a 16×16 matrix and 16×16 matrix multiplication (4096 ops) at FP16 or a 16×32 matrix and 32×16 matrix multiplication (8192 ops) at INT8.
 - The Vector unit implements computing between a vector and a scalar or between two vectors, supporting precisions of FP16, FP32, INT32, INT8, and more customized types.
 - The Scalar unit controls the program flow as a mini CPU via iteration control, selection judgment, address and parameter computations of Cube/Vector instructions, and basic arithmetic operations.

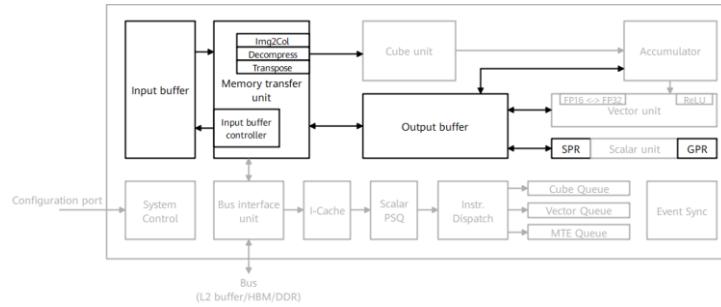


Da Vinci Architecture (AI Core) — Storage System (1)

- Storage units and corresponding data paths form the storage system of AI Core.
- Storage units consist of the storage control unit, buffers, and registers:
 - The storage control unit accesses lower level caches outside AI Core through the bus, and directly accesses DDR or HBM memory. A storage conversion unit is introduced as a transfer controller for the internal data paths in AI Core to implement read/write management of internal data of AI Core between different buffers, and for format conversion, such as padding, Img2Col, transpose, and decompress.
 - The input buffer temporarily stores frequently-used data to reduce round-trip to memory outside AI Core, which decreases data accesses over the bus and avoids bus congestion, achieving improved performance with lower power consumption.
 - The output buffer stores the intermediate results at each layer in NNs to facilitate data transfer to the next layer. Unlike the bus offering low bandwidth and severe latency, the output buffer greatly speeds up computation.
 - Registers in AI Core are mainly used by the Scalar unit.

Da Vinci Architecture (AI Core) — Storage System (2)

- Data paths: paths for data movement in AI Core during computations
 - Multiple-input single-output (parallel input) characterizes data paths in the Da Vinci architecture, in view of the diversity and large quantity of input data in NN computation. On the contrary, usually, only a feature matrix is output. A single output in data paths is enough, which frees up certain processor hardware resources.

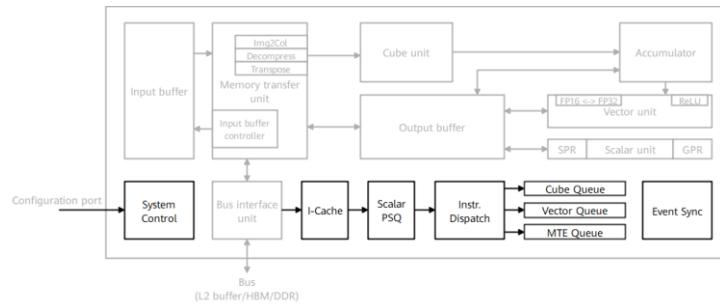


Da Vinci Architecture (AI Core) - Control Unit (1)

- Control units include System Control, I-Cache, Scalar PSQ, Instr. Dispatch, Cube Queue, Vector Queue, MTE Queue, and Event Sync.
 - System Control controls the execution process of the task block (the minimum computing task granularity of AI Core), and reports execution status through interrupts. If an execution error occurs, it reports the error to the task scheduler.
 - I-Cache is an instruction cache. During instruction execution, it prefetches and reads subsequent instructions at a time, accelerating instruction execution.
 - Scalar PSQ organizes decoded instructions, including those for matrix computation, vector computation, and storage conversion.
 - Instr. Dispatch reads the configured instruction addresses and decoded parameters in the Scalar PSQ, and sends instructions to corresponding execution queues by type. Scalar instructions reside in the Scalar PSQ.

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

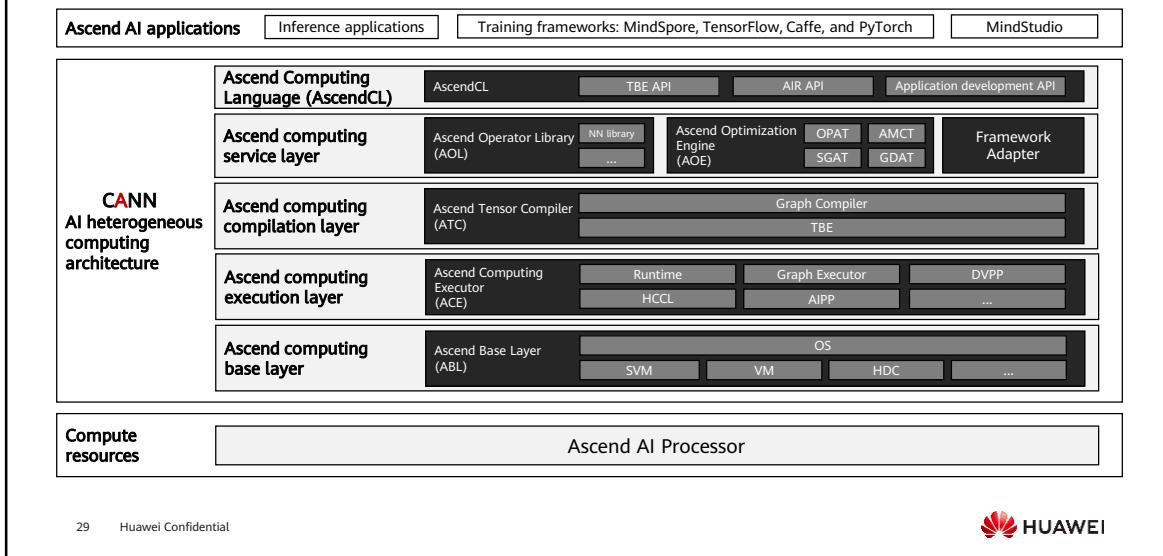
- Instruction execution queues include Cube Queue, Vector Queue, and MTE Queue. Different instructions go to corresponding queues and are executed in sequence.
- Event Sync controls the execution of each instruction pipeline in real time and analyzes the dependency between pipelines in consideration of data dependency and synchronization between instruction pipelines.



Contents

1. Huawei Ascend Computing Platform
 - AI Processor Overview
 - Hardware Architecture of the Ascend Processor
 - **Software Architecture of the Ascend Processor**
 - Huawei Atlas AI Computing Platform
 - Atlas Application in the Industry
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms

CANN AI Heterogeneous Computing Architecture (1)



- <https://www.hiascend.com/en/software/cann>

CANN AI Heterogeneous Computing Architecture (2)

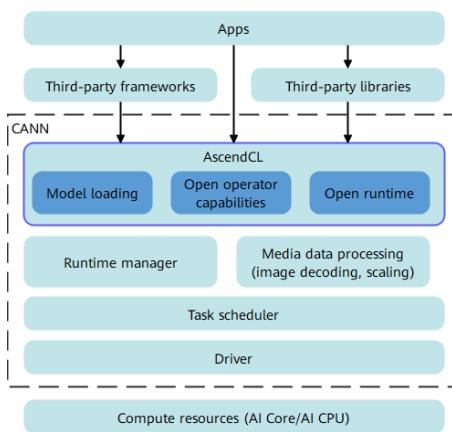
- Ascend AI Applications
 - Ascend AI applications include inference applications, framework applications (for training), and MindStudio IDE.
- Ascend Computing Language
 - The Ascend computing language (AscendCL) API is an open programming framework for Ascend computing. It shields the differences between underlying processors and provides TBE operator development API, AIR standard graph development API, and application development API, allowing users to quickly build Ascend-based AI applications and services.
- Ascend Computing Service Layer
 - The Ascend computing service layer provides AOL and accelerates computing by using high-performance operators of the NN library and Basic Linear Algebra Subprograms (BLAS) library. The service layer also provides AOE to improve end-to-end model running speed through OPAT operator tuning, SGAT subgraph tuning, GDAT gradient tuning, and AMCT model compression. In addition, framework adapters are provided to adapt mainstream AI frameworks such as TensorFlow and PyTorch.

- Inference applications refer to AscendCL-based applications. Framework applications are underlying training frameworks that use Ascend processors to train models, for example, MindSpore.

CANN AI Heterogeneous Computing Architecture (3)

- Ascend computing compilation layer
 - Ascend computing compilation layer uses Graph Compiler to compile the computational graph of the intermediate representation (IR) input by the user into a model executable by Ascend hardware. In addition, the automatic scheduling mechanism of tensor boost engine (TBE) is used to efficiently compile operators.
- Ascend computing execution layer
 - Ascend computing execution layer executes models and operators. The functional modules include the Runtime library, Graph Executor, DVPP, AI pre-processing (AIPP), and Huawei Collective Communication Library (HCCL).
- Ascend computing base layer
 - Ascend computing base layer provides base services for upper layers through shared virtual memory (SVM), virtual machines (VMs), and host-device communication (HDC).
- Compute resources
 - Compute resources, as the hardware computing power basis of the Ascend AI processor, execute specific computing tasks.

NN Software Flow of the Ascend AI Processor



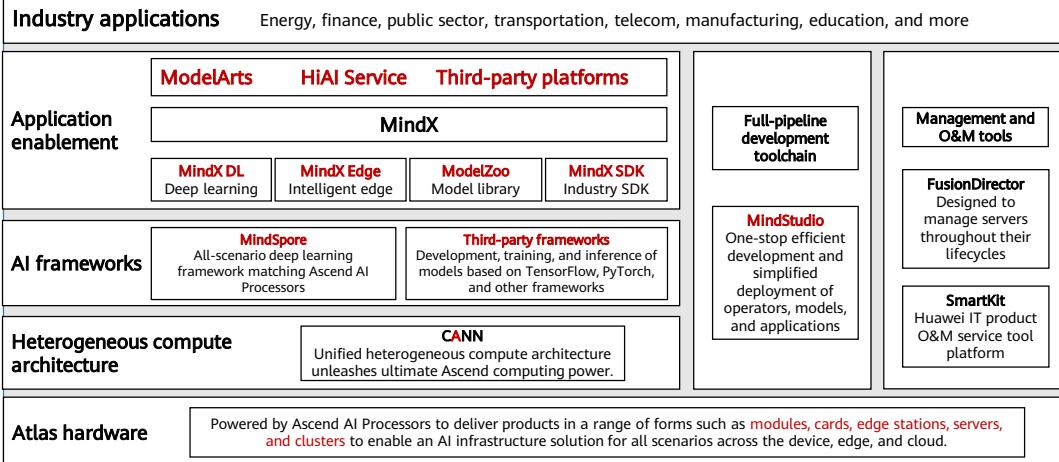
- The NN software flow of the Ascend AI processor is a bridge between the deep learning framework and the Ascend AI processor. It implements and executes NN applications and integrates the following functional modules:
- DVPP: performs data processing and cleaning before input to meet format requirements for computing.
- Open operator capabilities (TBE): continuously provides powerful computing operators for NN models.
- Open runtime (runtime manager): provides various resource management paths for task delivery and allocation of the NNs.
- Task scheduler: provides specific target tasks for the Ascend AI processor as a task driver for hardware execution. The runtime manager and task scheduler work together to form a dam system for NN task flow to hardware resources, and monitor and distribute different types of tasks for execution in real time.

- The figure shows the NN software flow of an inference application.

Contents

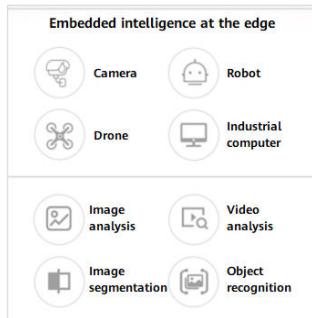
1. Huawei Ascend Computing Platform
 - AI Processor Overview
 - Hardware Architecture of the Ascend Processor
 - Software Architecture of the Ascend Processor
 - **Huawei Atlas AI Computing Platform**
 - Atlas Application in the Industry
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms

Ascend Full-Stack AI Software and Hardware Platform: Cornerstone of an Intelligent World



Atlas 200 AI Accelerator Module

- The **Atlas 200 AI Accelerator Module 3000** is widely used in device-side AI scenarios such as smart cameras, robots, and drones to implement object recognition and image classification.



Optimal Performance

- 22 TOPS INT8 in the size of half a credit card, supporting real-time analysis of 20 HD video channels of 1080p at 25 fps
- Multi-level computing power of 22, 16, and 8 TOPS configurations

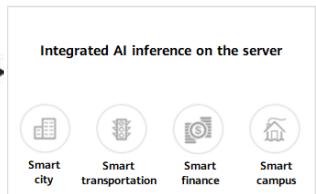
Ultra-low power consumption

- Milliwatt-level hibernation power consumption and millisecond-level wakeup enable edge AI applications with typical power consumption of 6.5 W.

The power consumption of the accelerator module can be adjusted. The computing power varies depending on the power consumption.

Atlas 300V Pro Video Analysis Card

- The **Atlas 300V Pro video analysis card** integrates the general-purpose processor, AI Core, and codec to provide powerful AI inference and video and image encoding and decoding functions. With advantages such as numerous video analysis channels, high-performance feature retrieval, and secure boot, it supports real-time analysis of 128-channel HD videos and can be used in a wide array of AI application scenarios, such as smart city, smart transportation, and smart campus.



Analysis of numerous video channels

- Real-time analysis of 128 channels of HD videos
- JPEG and video hardware encoding/decoding, boosting performance of image and video applications

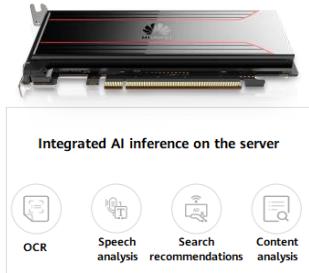
Secure boot

- Complete device boot chain and determined initial boot status of the device, preventing backdoor implantation



Atlas 300I Pro Inference Card

- The **Atlas 300I Pro inference card** integrates the general-purpose processor, AI Core, and codec, to provide powerful AI inference and object search functions. With advantages such as powerful computing power, ultra-high energy efficiency, high-performance feature retrieval, and secure boot, it can be widely used in a wide array of AI application scenarios, such as OCR, speech analysis, search recommendation, and content moderation.



Superior computing power

- Up to 140 TOPS INT8 per card to better support data center inference

• 8-core x 1.9 GHz CPU computing power

Ultra-high energy efficiency

- 2 TOPS/W, 2.1 times the industry average

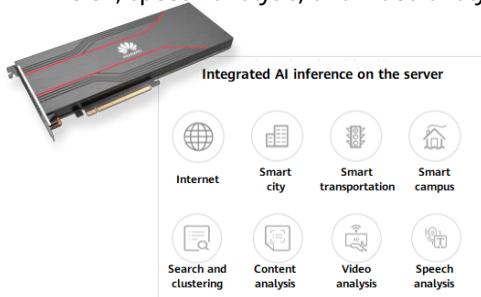
Secure boot

- Complete device boot chain and determined initial boot status of the device, preventing backdoor implantation



Atlas 300I Duo Inference Card

- The **Atlas 300I Duo inference card** integrates the general-purpose processor, AI Core, and codec to provide powerful AI inference and video analysis functions. With advantages such as superior computing power, ultra-high energy efficiency, and high-performance video analysis, it is suitable for a wide array of scenarios such as Internet, smart city, and smart transportation, and supports multiple applications such as search clustering, content analysis, OCR, speech analysis, and video analysis.



Superior computing power

- Up to 280 TOPS INT8 computing power per card to better support data center inference

16-core x 1.9 GHz CPU computing power

Ultra-high energy efficiency

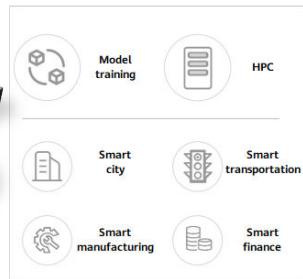
- Industry-leading energy efficiency ratio of 1.86 TOPS/W

High-performance video analysis

- Real-time analysis of 256 channels of HD videos
- JPEG and video hardware encoding/decoding, boosting performance of image and video applications

Atlas 300T Pro Training Card

- The **Atlas 300T training card** is an AI accelerator card that works with servers to provide powerful computing power for data centers. A single card provides up to 280 TFLOPS FP16 computing power, accelerating deep learning training. The Atlas 300T Pro features superior computing power, high integration, and high bandwidth to meet the requirements for AI training of Internet, carrier, and finance industries and computing power of high-performance computing.



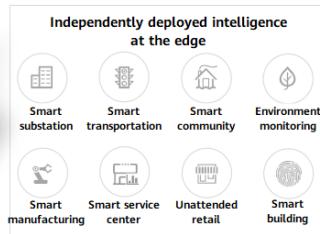
Superior computing power

- 30 built-in Da Vinci AI Cores
 - Industry-leading computing power of 280 TFLOPS FP16
- ### High integration
- Three-in-one (AI computing power, general-purpose computing power, and I/O capabilities)
 - Integration of 32 Huawei Da Vinci AI Cores, 16 TaiShan cores, and one 100GE RoCE v2 NIC
- ### High bandwidth
- Supports PCIe 4.0 and 1 x 100G RoCE high-speed ports, achieving the total egress bandwidth of 56.5 Gbit/s
 - Improves the training data and gradient synchronization efficiency by 10% to 70% without the need of external NICs

- Different from the inference cards, the training card is more complex. The 300T Pro provides both HBM2 and DDR4 memory for the AI core and AI CPU, and provides RoCE interfaces for inter-card communication and distributed training.

Atlas 500 AI Edge Station

- Oriented to edge applications, the **Atlas 500 AI edge station** (model 3000) features powerful computing performance, compact size, strong environment adaptability, easy maintenance, and support for cloud-edge collaboration. It can be widely deployed in edge scenarios to meet application requirements in complex environments such as security protection, transportation, communities, campuses, shopping malls, and supermarkets.



Intelligent edge

- Industry-leading edge product that integrates AI processing capabilities

- Operates from -40° C to 70° C outdoors with the fan-free design

Compact and powerful

- 22 TOPS INT8 in the size of half a set-top box

- Processing of 20 channels of HD videos of 1080p at 25 fps

Edge-cloud synergy

- LTE wireless transmission

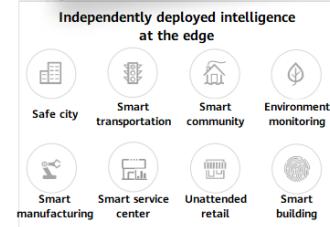
- Cloud-edge synergy for real-time model update

- Unified device management and firmware upgrade on the cloud



Atlas 500 Pro AI Edge Server

- The **Atlas 500 Pro AI edge server** (model 3000) is designed for edge applications. It features powerful computing performance, high environment adaptability, easy deployment and maintenance, and support for cloud-edge collaboration. It can be widely deployed in edge scenarios to meet application requirements in complex environments such as security protection, transportation, communities, shopping malls, and supermarkets.



Superior computing power

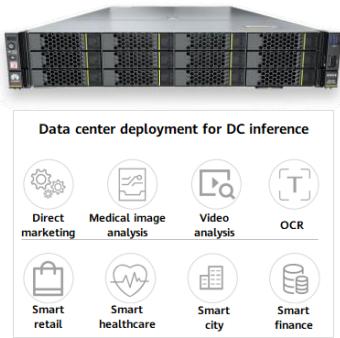
- Supports up to three Atlas 300 V/VI inference cards to meet inference requirements in multiple scenarios
- Supports real-time analysis of 384 channels of HD videos of 1080p at 30 fps
- Uses Kunpeng 920 processors to accelerate applications efficiently

Ultra-high energy efficiency

- Benefits from the multiple cores and low power consumption of the Kunpeng architecture to build an efficient and power-saving AI computing platform for inference scenarios
- Ultra-low power consumption of the Atlas 300I Pro (72 W per card), providing a better energy efficiency ratio while accelerating computing power of AI servers

Atlas 800 Inference Server (Model 3000)

- The **Atlas 800 inference server** (model 3000) supports up to 8 Atlas 300I/V Pro cards to provide powerful real-time inference and video analysis. It is perfectly suited for AI inference in data centers.



Superior computing power

- Supports up to eight Atlas 300I/V Pro inference cards to meet inference requirements in multiple scenarios
 - Supports real-time analysis of 1024 channels of HD videos of 1080p at 30 fps
 - Uses Kunpeng 920 processors to accelerate applications efficiently
- ### Ultra-high energy efficiency
- Benefits from the multiple cores and low power consumption of the Kunpeng architecture to build an efficient and power-saving AI computing platform for inference scenarios
 - Ultra-low power consumption of the Atlas 300I/V Pro (72 W per card), providing a better energy efficiency ratio while accelerating computing power of AI servers

Atlas 800 Training Server (Model 9000)

- The **Atlas 800 training server** (model 9000) features high computing density, ultra-high energy efficiency, and high network bandwidth. With these competitive advantages, the server is perfect for deep learning model development and training scenarios. It is an ideal option for compute-intensive industries, such as smart city, smart healthcare, astronomical exploration, and oil exploration.



High computing density

- 2.24 PFLOPS FP16 in a 4U space

Ultra-high energy efficiency

- 2.24 PFLOPS/5.6 kW

High network bandwidth

- 8 x 100GE RoCE v2 high-speed ports

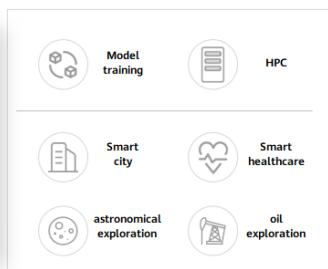
- Cross-server processor interconnect latency slashed by 10-70%



- High-speed interfaces are provided for distributed training.
- The energy efficiency ratio is continuously optimized and updated.

Atlas 900 PoD

- The **Atlas 900 PoD** (model 9000) is a basic AI training cluster unit that features powerful AI computing, optimal AI energy efficiency, and optimal AI expansion. With these competitive advantages, the basic unit is perfect for deep learning model development and training scenarios. It is an ideal option for AI compute-intensive fields, such as smart city, smart healthcare, astronomical exploration, and oil exploration.



Supreme AI computing

- Provides a maximum of 20.4 PFLOPS FP16 computing power with 47U

High AI energy efficiency

- Provides ultra-high energy efficiency of 20.4 PFLOPS/46 kW

Optimal AI expansion

- Supports cabinet unit expansion to a maximum of 4096 Ascend 910 processors, delivering the total computing capability of 1 EFLOPS FP16

- Different from the Atlas 800 training server, the Atlas 900 PoD is designed as a cluster unit for large-scale training.

Contents

1. Huawei Ascend Computing Platform
 - AI Processor Overview
 - Hardware Architecture of the Ascend Processor
 - Software Architecture of the Ascend Processor
 - Huawei Atlas AI Computing Platform
 - **Atlas Application in the Industry**
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms

Atlas Facilitates Huawei's All Service Scenarios

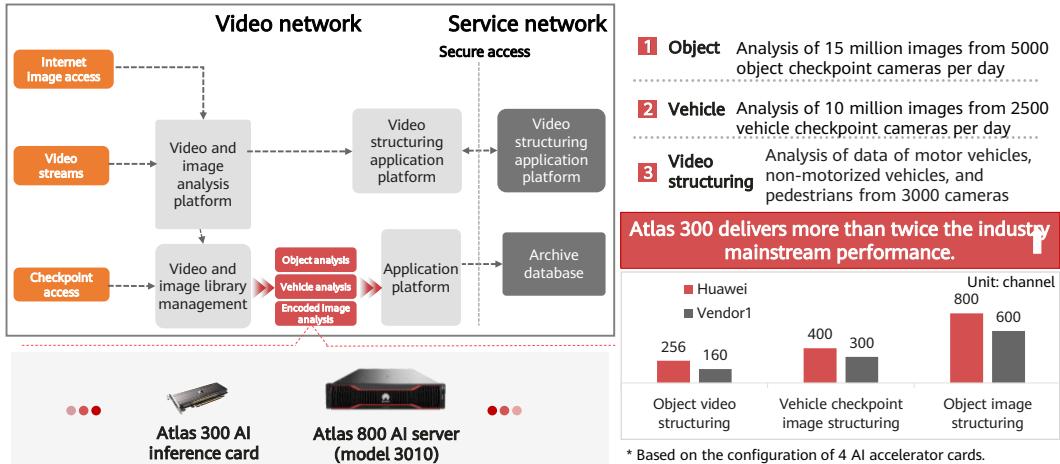


46 Huawei Confidential



- Huawei has deployed more than 100,000 Kunpeng and Ascend devices, covering full-service scenarios from R&D, production, office, delivery, to sales.

Atlas-Based Video and Image Application Platform Improves Structuring Efficiency by 60%



47 Huawei Confidential



- Actual project requirements:
 - Object analysis
 - Analysis of 15 million images from 5000 object checkpoint cameras per day
 - 15 cards, each processes 1 million images per day
 - Two 8-card servers
 - Vehicle analysis
 - Analysis of 10 million images from 2500 vehicle checkpoint cameras per day
 - 10 cards, each processes 1 million images per day
 - Two 8-card servers
 - Video structuring
 - Analysis of data of motor vehicles, non-motorized vehicles, and pedestrians from 3000 cameras
 - Each card processes 20 video channels.
 - An 8-card server processes 160 video channels.

Atlas Helps Peng Cheng Cloud Brain II Build a National AI Platform



Atlas 900

Building AI supercomputing
based on Kunpeng + Ascend
256~1024 PFLOPS FP16

6195 x86 cabinets = 208 GPU cabinets = 16 Atlas cabinets
40,268 kW 1,352 kW 736 kW

AI Industry Applications Facilitate the Development of the Greater Bay Area

	Smart transportation	Smart healthcare	Smart finance
National Strategies	<ul style="list-style-type: none">International AI supercomputing platformNational open source platform for AI technologiesOpen source AI and innovative ecosystem		
Serving Shenzhen	<ul style="list-style-type: none">Supports major AI application requirements such as the intelligent computing system and robot system in Guangdong, Hong Kong, and Macao.Improves the basic position and innovation of AI research on open source platforms and intelligent applications in the Guangdong-Hong Kong-Macao Greater Bay Area.Attracts national AI resources, technologies, and talent.		

48 Huawei Confidential

Peng Cheng Cloud Brain II won three world No. 1s in AIPerf, IO500 full-node, and 10-node rankings. It participated in the industry-recognized MLPerf training v1.0 benchmark test and ranked second in the image classification track (with 1024 cards of the same scale).

Benchmarking Competitors: TCO Reduced by 9.3% with the Same Computing Power

1. **Computing power improved by two times**
1024+ Interconnected Ascend 910 processors, each delivering computing power twice the industry average

2. **Network latency reduced by 70%**
Integrates three high-speed interfaces: HCCS, PCIe 4.0, and 100G RoCE, reducing latency by 70%.

3. **Cost on power cut by over 60%, space reduced by 80%**
50 kW hybrid liquid cooling system for a single cabinet, PUE < 1.1
 Ultra-high-density prefabricated modular equipment room, low power consumption, fast deployment, and exabyte-level cloud brain cluster rollout within six months



- The world's fastest training cluster, Atlas 900, consists of thousands of Ascend 910 AI processors. Leveraging the Huawei cluster communication library and job scheduling platform, it also integrates HCCS, PCIe 4.0, and 100G RoCE interfaces to fully unlock the computing power of Ascend 910. It delivers 256 to 1024 PFLOPS @FP16, a performance equivalent to 500,000 PCs, allowing users to easily train algorithms and datasets for various needs.
- Huawei has deployed an Atlas 900 cluster on HUAWEI CLOUD. With the Atlas 900 cluster, training of a typical network can be completed within 59.8 seconds, 15% faster than the nearest competitors without compromising the accuracy (75.9%), based on the ResNet-50 v1.5 model and ImageNet-1k dataset with 1.28 million images. In addition to basic hardware, Huawei also provides HUAWEI CLOUD EI cluster services based on Atlas 900. The full-stack optimization from underlying hardware to development architecture provides superb computing power.

Huawei Helps Build a 'City-Wide Traffic Brain' for Predictive Travel

- With video cloud, big data, and AI being the core technologies, this 'Traffic Brain' presents a unified, open, and intelligent traffic control system. It focuses on the following five significant areas:

1	Ultra-broadband traffic network	Technologies such as a high-speed Optical Transport Network (OTN) are used to support transmission at 400G bandwidth, data storage of over 20 PB, and 10 billion-level data processing. The data bearing capability is approximately 40 times that of traditional public security networks.
2	City-wide comprehensive traffic awareness	A road monitoring system detects traffic conditions through license plate identification, video surveillance, and more, with a detection accuracy rate of up to 95%. The system collects about 700 million pieces of vehicle data every month, and integrates nearly 40 TB of data from 78 system databases, both internal and external. All these contribute to useful Big Data-enabled traffic congestion analysis and optimization.
3	AI-assisted law enforcement	The use of AI technology improves the efficiency of identifying traffic-violation images by 10 folds, ensuring the closed-loop processing of those images.
4	Improved crime fighting efficiency	The big data platform and a traffic analysis modeling engine are used to create multiple big data analytics models, including disqualified driving, drug driving, and multi-violation. Intelligence can now be generated and precisely pushed within 30 minutes, helping to pinpoint then clampdown on violations.
5	Improved travel experience	Scientific setting of intersection channelization, plus traffic organization innovation through big data management and control, is helping to improve road capacity by approximately 8%.

49 Huawei Confidential



- In the future, the city will develop smart police terminals based on 5G technologies; analyze historical data by means of traffic simulation to predict the traffic volume in each corner of the city; invest nearly CNY 3 billion to upgrade all checkpoint systems, including the HD traffic camera system, to cover more than 70% of the city roads. In this way, a complete traffic collection network can be built. Relying on the most powerful urban traffic brain and related measures, the city will build a robust traffic network. Also, the open big data platform can provide powerful support for the city in multiple aspects such as in-depth AI application and law enforcement.

Contents

1. Huawei Ascend Computing Platform

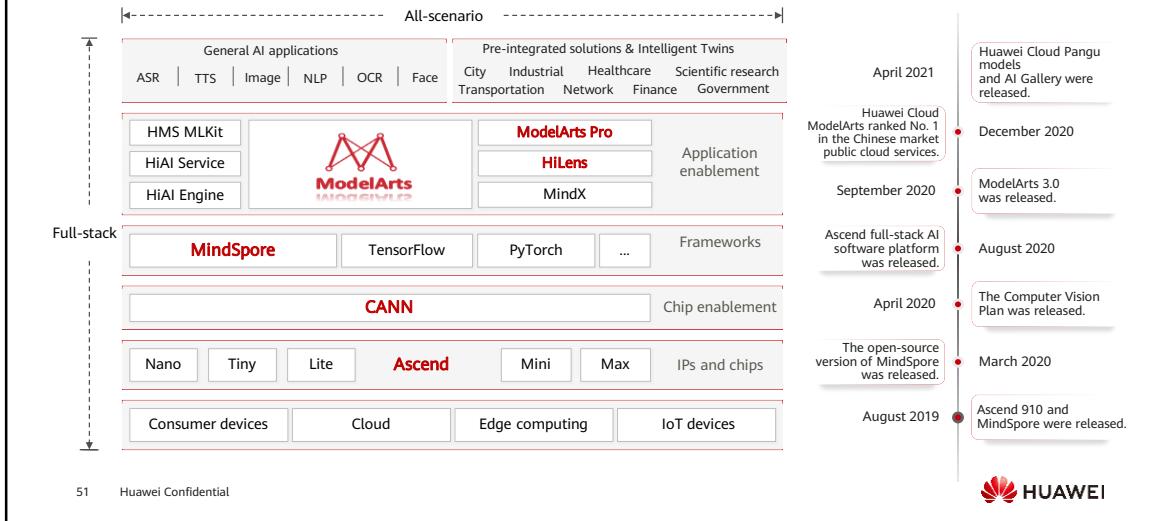
2. Huawei Cloud EI Platform

- **Huawei Cloud EI Open Capability Panorama**

- **Huawei Cloud AI Development Platform ModelArts**
- **Huawei Cloud General AI Capabilities**
- **Intelligent Twins**

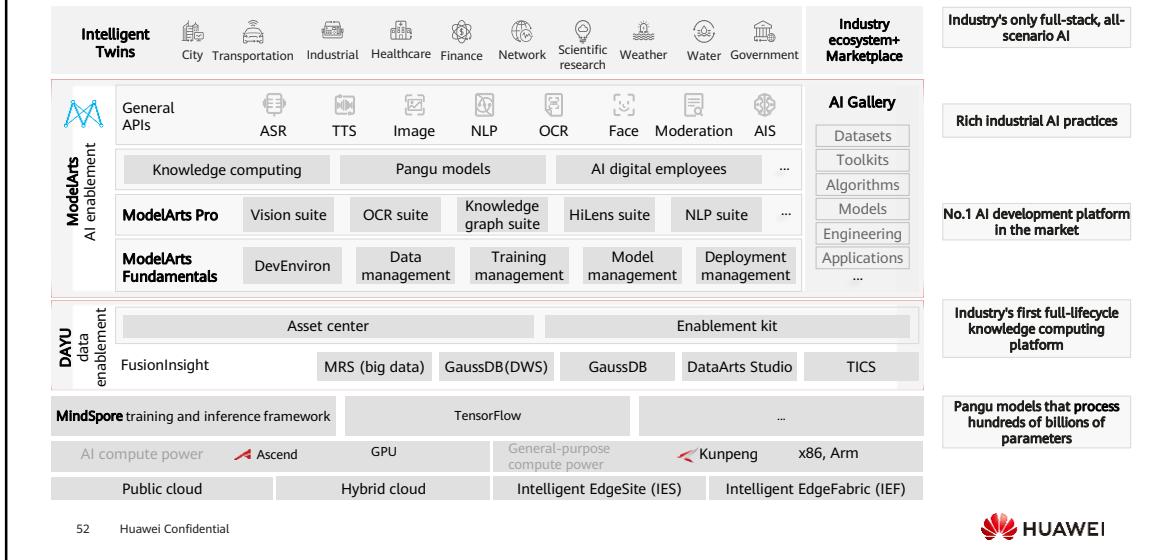
3. Huawei Device AI Platform

Huawei's Full-Stack, All-Scenario AI Promotes the Intelligent Upgrade of Thousands of Industries



- To address enterprises' challenges and difficulties, Huawei proposes a full-stack, all-scenario AI solution to promote the intelligent upgrade of thousands of industries. From a full-stack perspective, or you could say a technical perspective, the foundational bottom layer of this scenario contains devices, IoT, public cloud, and private cloud. The chip layer contains a series of Huawei-developed Ascend AI chips. The CANN layer offers a chip operator library and automated operator development tools. It provides optimal development efficiency and operator performance. The framework layer provides the Huawei-developed AI framework, MindSpore. The AI application layer contains Huawei device services HMS and HiAI, one-stop AI development platform ModelArts, and AI developer community AI Gallery. From an all-scenario perspective, Huawei provides various types of general API services, such as image and text recognition, facial recognition, speech recognition, and speech-to-text conversion. These functions can be integrated into devices or cloud products through APIs, which provide AI capabilities. In addition, Huawei provides mature Intelligent Twins and solutions for various industries based on industry knowledge.
- In 2019, Huawei launched Ascend 910 and MindSpore, and that was when Huawei started to deploy its AI strategy. Huawei has updated several MindSpore and ModelArts versions, released the Ascend full-stack AI software platform, and released Pangu models and AI Gallery this year. For now, Huawei has established a complete set of full-stack all-scenario AI solutions to help enterprises better use AI.

Huawei Cloud EI Open Capability Panorama



- Industry's only end-to-end full-stack AI solution. It supports AI deployment in cloud services, edge computing, terminal applications, and more.
- 10+ industry applications and 800+ projects. AI is already being used in 30% of enterprise business systems and has increased profit by an average of 18%.
- Huawei Cloud ModelArts ranks No. 1 in China's public cloud service market for machine learning.
- Industry's first knowledge graph pipeline that integrates industrial mechanism models, expert experience, and AI.
- Huawei Cloud Pangu models provide strong generalization capabilities and model performance, helping you quickly develop AI for industry scenarios.

Contents

1. Huawei Ascend Computing Platform
2. **Huawei Cloud EI Platform**
 - Huawei Cloud EI Open Capability Panorama
 - **Huawei Cloud AI Development Platform ModelArts**
 - Huawei Cloud General AI Capabilities
 - Intelligent Twins
3. Huawei Device AI Platform

From AI+ to +AI: AI Empowers Best Practices

AI+

Explore AI capabilities



EfficientNet model accuracy: 98.7% for the top 5 labels, higher than manual accuracy (96%)



RNN-T model accuracy: 96.8%, higher than manual accuracy (94.17%)



BERT model accuracy: 90.9%, higher than manual accuracy (82%)

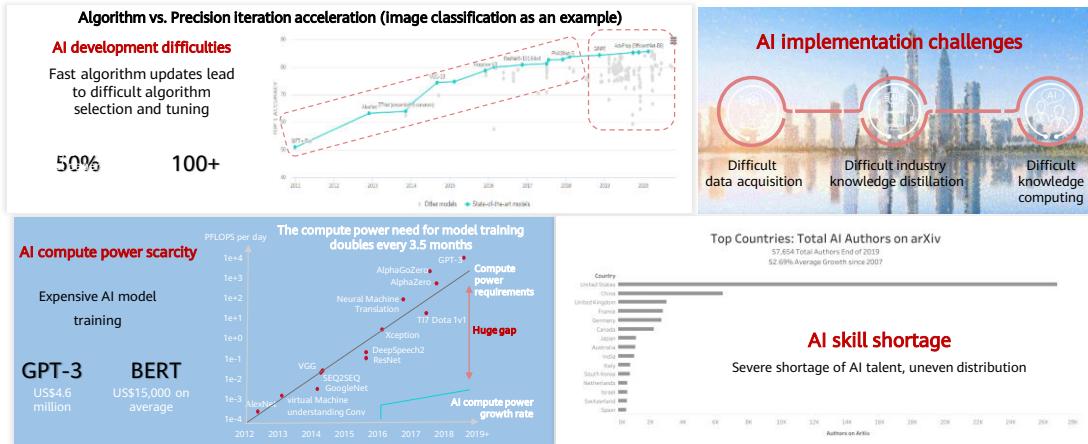
+AI

AI-empowered core production systems



...

Enormous Challenges in Deploying AI applications



55 Huawei Confidential



Rich Industry Practices Accelerate the Application of AI in Enterprise Production Systems

30%+
AI-powered core application systems

18%
Higher profitability

More than 10 years of experience and practices of 800+ Huawei projects accelerate the implementation of AI

Healthcare	Transportation	Industrial	Water management	Weather forecasts	Airport
Hours → Minutes AI-powered genome analysis	17.7% Reduced vehicle delay percentage	CNY30 million Costs saved per million tons of coal with intelligent coal blending	81% Water vision accuracy rate vs.	2 hours Short-term forecasts	10% O&M efficiency improvement
Months → Hours AI-assisted drug screening	4.2% Average vehicle speed	80% More precise steel quality inspection	10% Industry average	10 minutes Thunderstorm forecasting	4 million person-times/year Not taking shuttle buses

Huawei Has Been Developing a New AI Development Model Together With Developers

**50,000+ jobs/month
840,000+ hours/month**

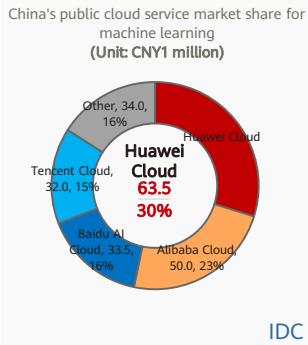
**50,000+ monthly active users
700,000+ AI developers
2500+ AI partners**

**30+ AI contests
70,000+ contestants
70+ universities**

Statistics as of October 2021

ModelArts Continuously Sees Greater Influence in the Industry

Huawei Cloud ranks No. 1 in the Chinese machine learning market according to IDC's latest data on China's public cloud market.



The Forrester Wave: Huawei Cloud was named as a leader in PAML.



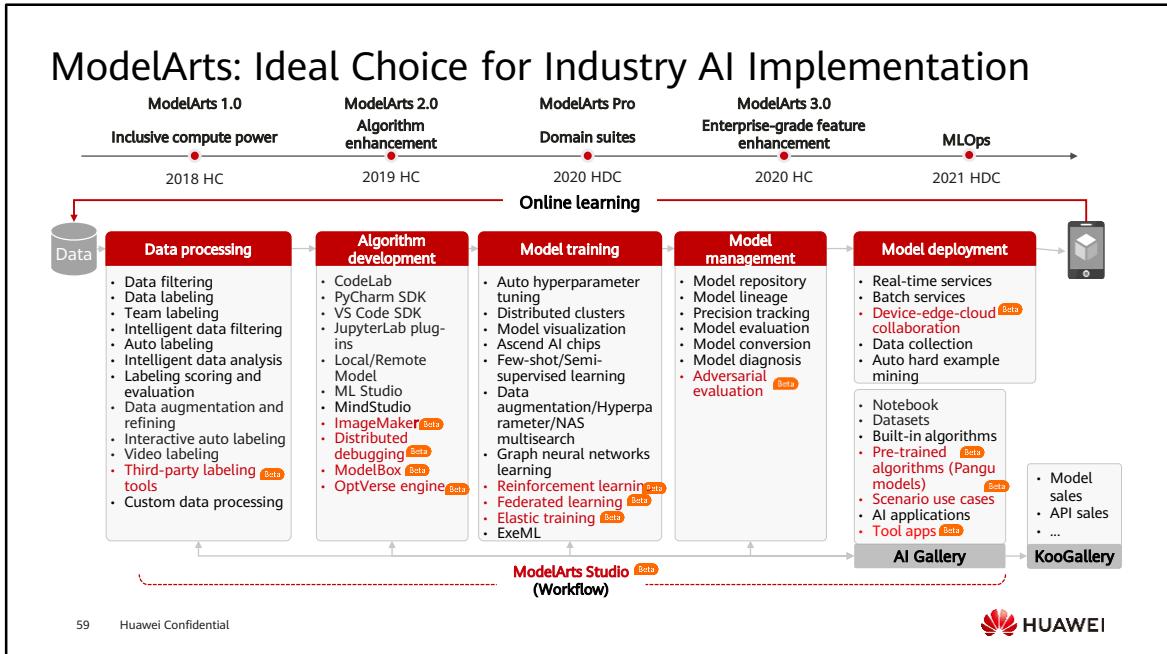
Huawei Cloud ModelArts has gained national recognition and its influence in the industry is continuously increasing.



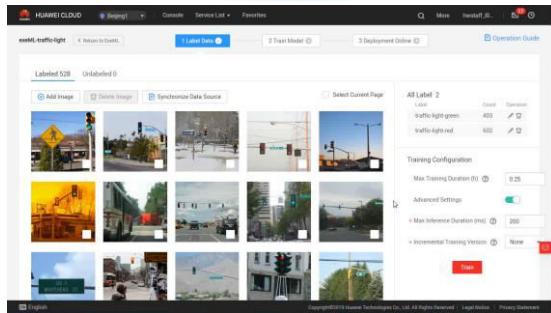
At the World Artificial Intelligence Conference 2019, the Ministry of Science and Technology of China announced that Huawei was tasked with building China's next-generation AI open innovation platform for basic hardware and software. The platform combines cloud services and product software and hardware to provide full-process and inclusive basic platform services oriented to research and development of AI applications for various industries, startups, universities, and research institutions.



ModelArts: Ideal Choice for Industry AI Implementation



ExeML Engine Helps Beginners Easily Handle Common Application Scenarios



0
Code

0
AI experience

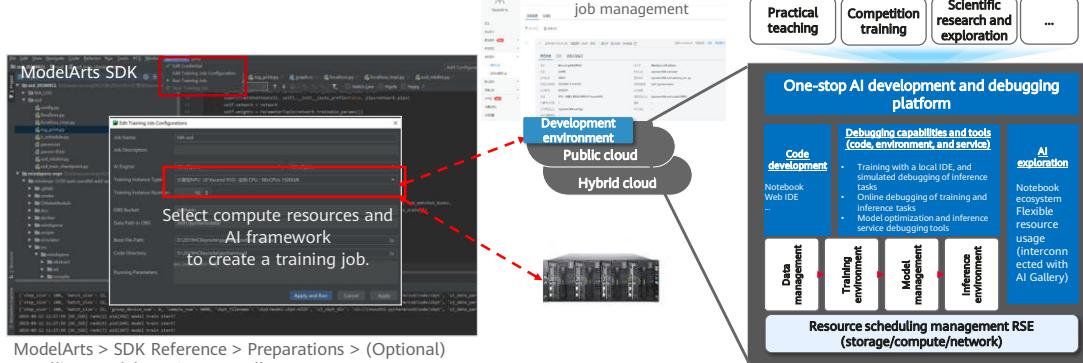
Step 1:
Upload data and
label it.

Step 2:
Train a model.

Step 3:
Check and publish
the model.



Development Environment 2.0, Providing the Ultimate Experience on Cloud and On-Premises



ModelArts > SDK Reference > Preparations > (Optional)
Installing ModelArts SDKs Locally

https://support.huaweicloud.com/intl/en-us/sdkreference-modelarts/modelarts_04_0004.html

Experience: consistent experience on cloud and on-premises, out-of-the-box, and provisioning in seconds

Flexibility: flexible resource switching between dedicated and shared, 5 to 10 higher resource utilization

Experience: 300 cards for 1000 concurrent users (GPUs used within 25% of a 1-hour course)

Ecosystem: Seamless integration of AI Gallery (IPython Notebook) zone and GitHub

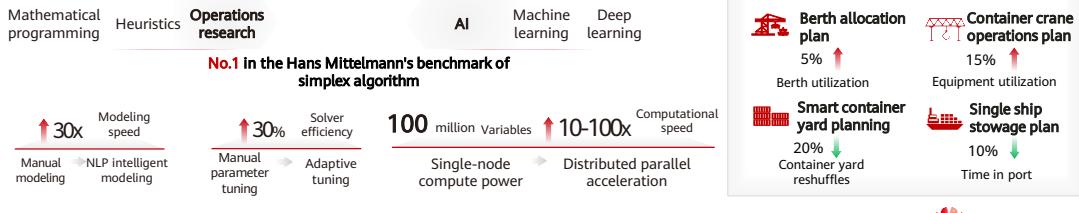


- AI supercomputing requires practical training, competition, and scientific research and exploration. These customers mainly use DevEnviron, and user experience of DevEnviron is especially important.

Huawei Cloud OptVerse AI Solver Helps Customers Make Informed Decisions and Optimize Services



OptVerse AI Solver



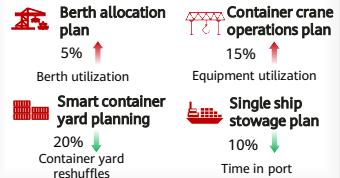
62 Huawei Confidential

Empower Tianjin Port's intelligent port planning platform



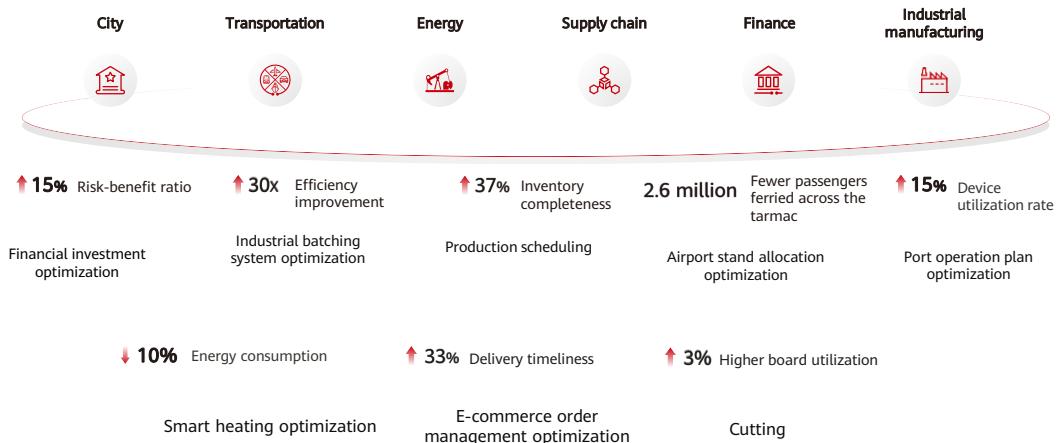
Tianjin Port: operation plan optimization

Time to develop a plan **24 hours** → **10 minutes**

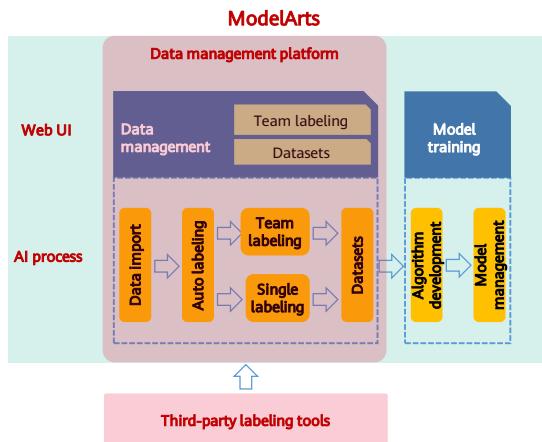


HUAWEI

Tailoring OptVerse AI Solver to the Specific Needs of Industry Scenarios for Easier Application

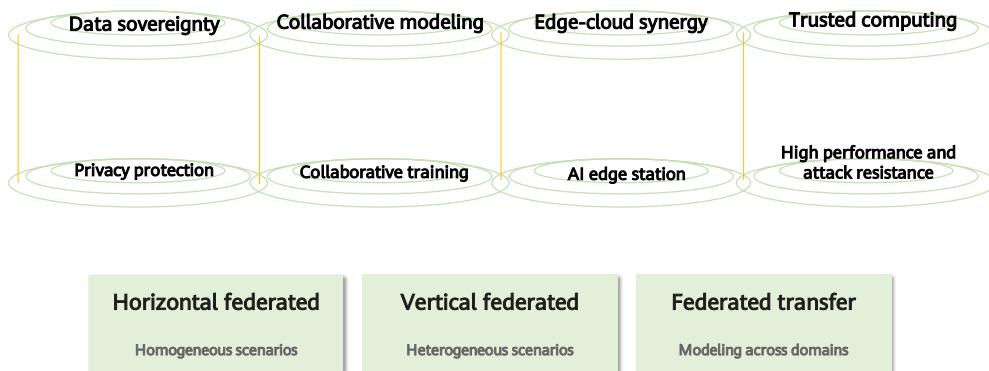


Data Management Supports a Wide Range of Data Formats and Iterative Auto Labeling



- **A wide range of data formats**
 - Five types of data
 - Image, audio, video, text, and table
 - Custom data formats
- **Team labeling**
 - Ideal for ultra-large-scale labeling
- **Iterative intelligent labeling framework**
 - Adaptive to data and algorithm changes
 - Intelligent data filtering and auto pre-labeling

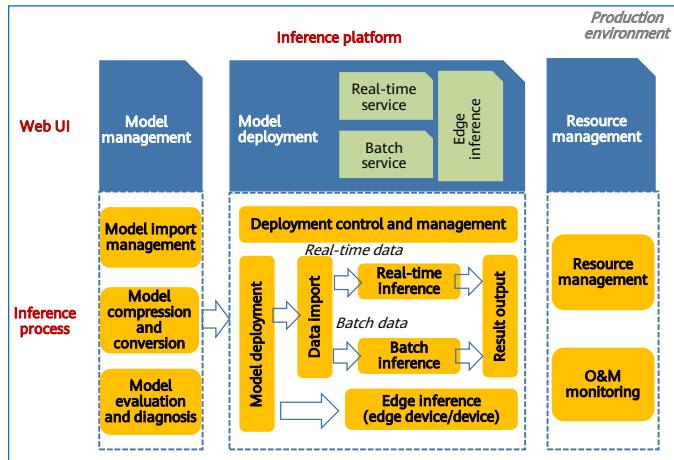
Federated Learning: Eliminates Data Silos and Promotes Collaborative Modeling Across Industries



- Data is the basis of AI applications. Diverse data is essential for intelligent AI awareness. However, in industrial AI implementation, data is scattered among different data controllers. This causes data silos of industry applications and limits the training effect of AI algorithms.
- To solve this problem, Huawei Cloud ModelArts provides federated learning to implement cross-domain modeling. Users use local data for training. They do not exchange data but only exchange updated model parameters in encryption mode to implement collaborative training.
- Huawei Cloud EI cooperates with Academician Jiang Hualiang of Shanghai Institute of Materia Medica (SIMM), Chinese Academy of Sciences to apply Huawei-developed FedAMP and AutoGenome algorithms to AI-powered drug R&D. These algorithms can accurately predict drug solubility, cardiotoxicity, and kinase activity. The accuracy is much higher than that of traditional federated learning and deep learning algorithms.
- Huawei Cloud also provides cloud services with cloud-edge synergy to support the federated training of data of different customers in different places. The data is uploaded to the server in encryption mode, the global model updates, and is then delivered to edge devices. This enables the horizontal federated learning of the same business and vertical federated learning of different businesses. Federated training customers can participate in federated training using the cloud or Huawei Cloud edge devices (such as AI edge stations) to implement

joint modeling in the industry.

Inference Platform: Lifecycle Management of AI Application Running



66 Huawei Confidential

- **Unified management**

- Unified management of models with different frameworks and functions from different vendors

- High-concurrency model deployment, low-latency access, auto scaling, grayscale release, and rolling upgrade

- **Flexible deployment**

- A rich array of deployment scenarios, including real-time inference services and batch inference jobs on the cloud, edge devices, and devices
- Real-time inference of large models and model update in seconds, adapting to fast service iteration

- **Higher performance**

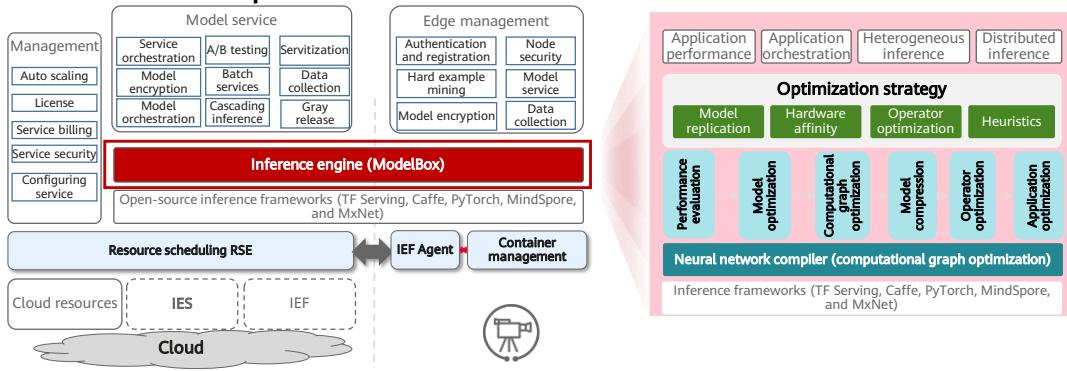
- Huawei-developed inference frameworks hide underlying hardware and software differences from the upper layer software to improve performance.

- **Iterative model update**

- Data collection and automatic hard example mining to quickly adapt to data changes

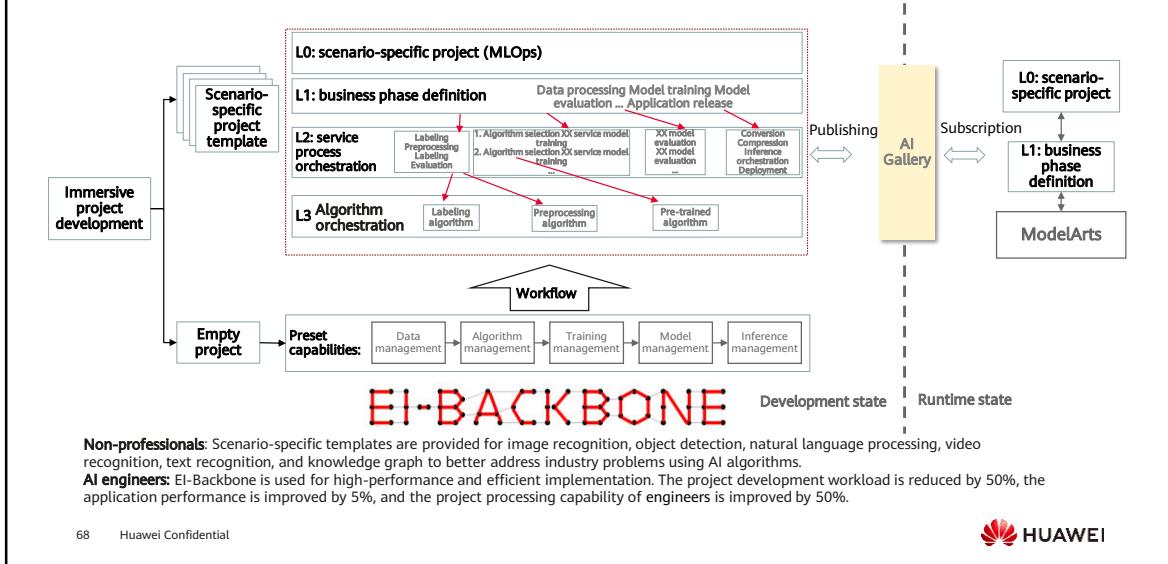


ModelBox: Device-Edge-Cloud Joint Development and Real-Time Model Update



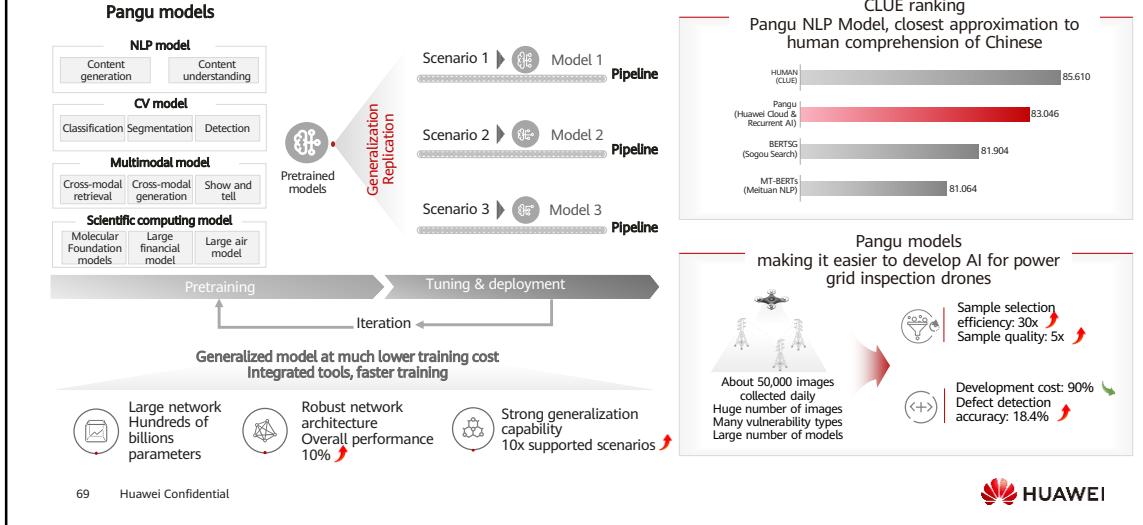
- Mindx-Edge provides edge Kubernetes container management and common open-source inference frameworks.
- AIFlow Engine is unique to ModelArts.

MLOps Immersive Development with Standardized Process

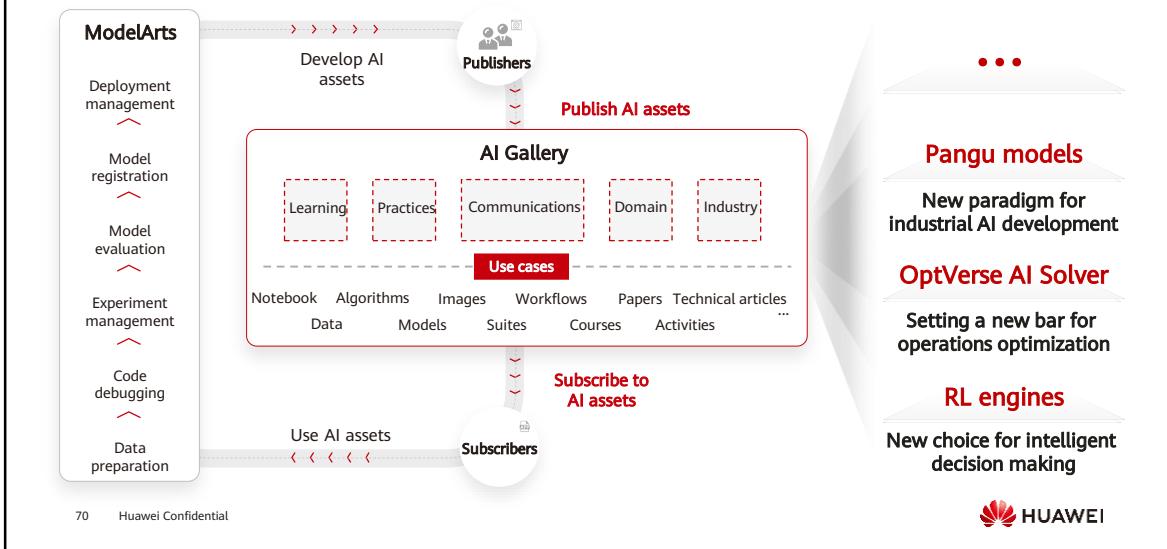


- The immersive development was released for public cloud OBT on November 30, and will be released for hybrid cloud in H1 next year.
- EI-Backbone is a backbone tool chain of Huawei Cloud. It integrates industry-leading models, data, computing, and knowledge to improve industrial AI implementation. EI-Backbone has been successfully applied in more than 10 industries and won more than 10 titles in competitions. More than 100 top-level conference papers about EI-Backbone have been published. EI-Backbone provides a new pattern for AI development. Let's look at **lung image segmentation**. In the past, hundreds of labeling data records were required for training. Now, with EI-Backbone, only dozens of labeling data records are required, **reducing labeling costs by more than 90%**.

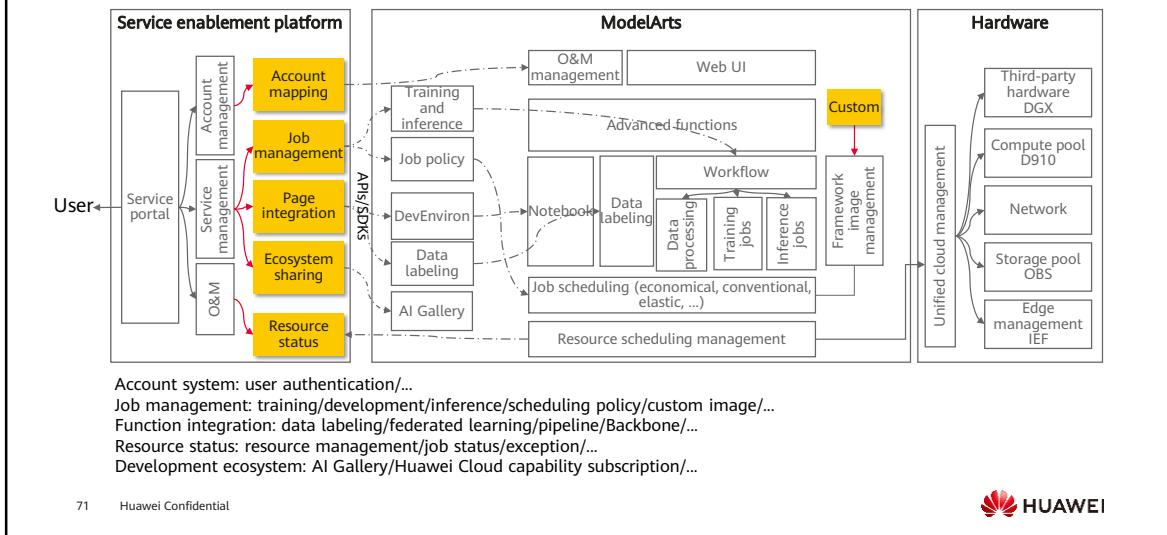
Huawei Cloud Pangu Models: Pioneering A New Paradigm for Industrial Scale AI Development



AI Gallery: Bridging Supply and Demand in the AI Ecosystem



Third-Party Service Platform Integration Solution



- ModelArts provides cloud services integrating third-party application platforms, such as Peng Cheng Cloud Brain and Leinao of Chinese Academy of Sciences.
- The yellow part indicates the workloads interconnected with third party platforms. There are sample codes for this part and can be provided to customers.

Contents

1. Huawei Ascend Computing Platform
2. **Huawei Cloud EI Platform**
 - Huawei Cloud EI Open Capability Panorama
 - Huawei Cloud AI Development Platform ModelArts
 - **Huawei Cloud General AI Capabilities**
 - Intelligent Twins
3. Huawei Device AI Platform

Vision Services: Empowering Service Systems with Awareness Capabilities

OCR	Recognition of general tables, ID cards, driving licenses, vehicle licenses, express waybills, contact cards, 1D code, QR code, bank cards, invoices used for reimbursement, network screenshots, and slogans, and quick customization...
	
Image recognition	Image tagging, image classification, content moderation, offering search, scenario identification, image recognition, image comparison, image interpretation, license plate recognition, and customization...
	
Facial recognition	Portrait recognition, facial recognition, facial attributes, face detection, human figure recognition, human body recognition, action recognition, human body analysis, face retrieval, face verification...
	

73 Huawei Confidential

100+ tables

Value recognition rate > 98%, supporting recognition of letters and digits in sealed, twisted, tilted, or interlaced tables

23 thousand+ labels

Semantic label, copyright search, reverse image search...

Recognizing a target from 100,000 records in seconds

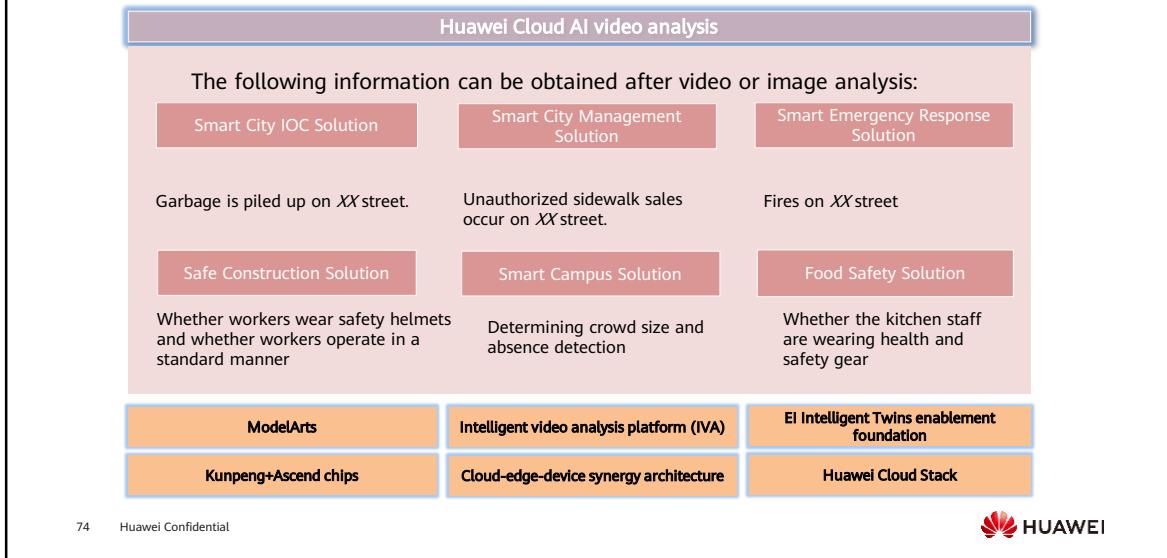
95.50%+ search accuracy for a face library with 100,000 face records



Smart vision services empower service systems with awareness capabilities.

- OCR includes general text recognition for certificates, receipts, and industries. In addition, OCR can be quickly customized.
- Image recognition service includes image recognition, image search, and content moderation, which are widely used in industrial quality inspection, government affairs, and Internet services.
- Facial recognition service includes facial recognition, action recognition, and face search.

VAS Makes Video AI Available to All Industries



- VAS uses AI technologies to intelligently analyze videos. It provides functions such as video detection, video content moderation, video search, and content analysis, and provides efficient video analysis in industries such as smart city, smart campus, and smart security.

Natural Language Processing (NLP)

(Named Entity Recognition (NER))	(Word segmentation)	(Sentiment analysis)	(Text summarization)		
Named Entity Recognition (NER) Extracts entities, such as person names, organization names, and place names, in text. <ul style="list-style-type: none"> ✓ 10+ industry-leading NER models ✓ 30+ supported categories (3 to 4 categories supported by other industry peers) ✓ 10% higher accuracy than that of major competitors 	Word segmentation Segments a text into sequences in unit of independent words and attaches a part-of-speech (POS) tag to each word. <ul style="list-style-type: none"> ✓ Supports multiple mainstream standards, such as PKU, CTB, and MSR ✓ Supports multi-granularity word segmentation. ✓ 6% higher accuracy than that of major competitors 	Sentiment analysis Analyzes the sentiments involved in a text to determine whether the text is positive or negative. <ul style="list-style-type: none"> ✓ Supports sentiment analysis in multiple domains. ✓ Supports attribute-based sentiment analysis. ✓ 10% higher accuracy than that of major competitors 	Text summarization Automatically summarizes the main content of a text to form a summary. <ul style="list-style-type: none"> ✓ Automatic extraction and customizable content length ✓ 12% higher accuracy than that of major competitors 		
 Full-stack NLP service <ul style="list-style-type: none"> • Basic algorithm/Language understanding/Language generation/Machine translation • Available for all mainstream service scenarios 	 Cost-effectiveness <ul style="list-style-type: none"> • Industry-leading performance • 5%+ ahead of the competition in accuracy 	 Stability and reliability <ul style="list-style-type: none"> • Multi-active instance deployment and high service stability • Fault recovery within seconds and multi-dimension fault isolation 	 Domain customization <ul style="list-style-type: none"> • Named entity recognition supports multiple domains, such as business and entertainment. • Sentiment analysis supports e-commerce and automobile. • NLP supports customization in specific scenarios. 		
<p>Won numerous international awards World-leading precision, performance, and effect</p>					
	Big Data Expo 2019 Leading Technology Achievement Award	2019 CCKS Technical innovation award	2019 DiggScience Champion	2019 CCF BDCI Champion	2020 WSDM Champion

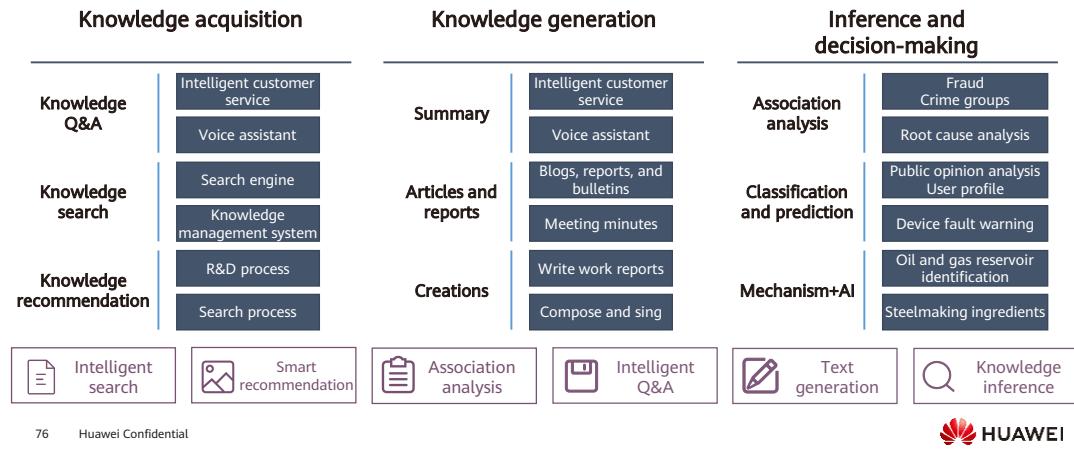
75 Huawei Confidential



- NLP covers mainstream scenarios, including discrimination tasks (named entity recognition and word segmentation) and generative tasks (sentiment analysis and text summarization). It is cost-effective, stable, and reliable, and supports customization in specific scenarios.

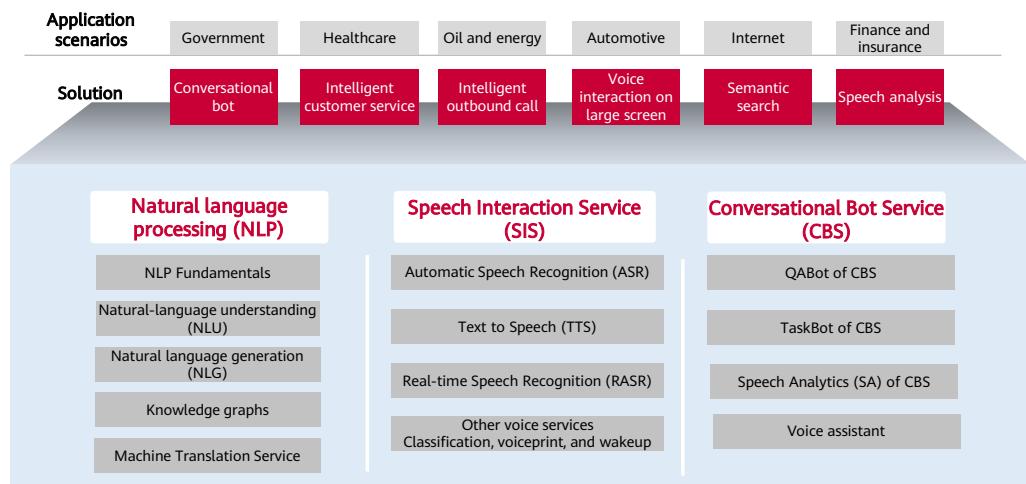
Knowledge Graph Service: Full-Stack, Full-Lifecycle, and First-Class in the Industry

A knowledge graph is a structured semantic knowledge base, which is used to quickly depict concepts and relationships between concepts in the physical world. The Knowledge Graph (KG) service provides one-stop knowledge graph lifecycle management, including graphical ontology designing, automatic graph pipeline building, and graph applications such as intelligent Q&A, intelligent search, and knowledge inference.



- The Knowledge Graph (KG) service provides full-stack knowledge graph lifecycle management, including knowledge acquisition, knowledge generation, and inference and decision-making. It is widely used in scenarios such as intelligent search, intelligent recommendation, association analysis, intelligent Q&A, text generation, and knowledge inference.

Rich Basic Speech and Semantics APIs+Conversational Bot



77 Huawei Confidential



NLP and voice interaction services such as speech recognition and speech synthesis are flexibly combined to build conversational robots, intelligent customer service, and intelligent outbound call services, which are used in government, healthcare, energy, and Internet scenarios.

Speech and semantic APIs provide NLP, knowledge graph, speech and language understanding, language generation, and machine translation. They can be flexibly combined to implement conversational bots, knowledge computing, text summarizations, and semantic searches, for scenarios such as government, healthcare, oil and energy, and automobile.

- First, let's look at speech and semantic products, which can be classified into four categories: voice access, voice processing, text processing, and knowledge management.

Voice access: Cognitive Engagement Center (CEC), a call center platform, which allows the access of incoming voice calls and enables outbound voice calls.

Voice processing: Automatic Speech Recognition (ASR), Real-time Speech Recognition (RASR), and Text to Speech (TTS) are provided for conversion between speech and text, such as sentence transcription, RASR, audio file transcription, TTS, and TTS customization.

Customer Service+AI: Huawei Cloud Conversational Bot Service (CBS) Is Used to Quickly Build an Intelligent Q&A System for Customer Service

The screenshot shows the CBS interface. At the top, there's a navigation bar with links for Home, Online Help, Call 12345, Service Request, and Service Records. Below the navigation is a sidebar with a tree view of service categories: Account/Privilege, New Employee, ePost/Email, Attendance/Leave, New PC/Replacement, Vehicle/Property, Meeting Book, Desktop Cloud, Certificate/Design, Performance/CQ, Reception & Card, IT Services/SE, Business Travel, and Logistics. The main area features a search bar with placeholder text "One number, One dialog, ...". Below the search bar are several buttons: All, IT, Travel, HR, Logistics, Admin, AP, Procurement, and a magnifying glass icon. A message says "IT Replenishing, Please input your question here." At the bottom of the main area, there are three links: "FAQs about maintaining inventory", "How should I clean the transparent...", and "Upgrade Your Desktop Cloud To W...". Below this is a section titled "Self-service" with a horizontal menu: IT, HR, Admin, Travel, Finance, Procurement, and SC. Underneath the menu are icons for IT on-site service, Mail Configuration, System Configuration, Cleaning Disk C, Print driver install..., Proxy Configuration, and Office 19 Activation.

79 Huawei Confidential

Service scenario

- Customer needs are increased, and thus more customer service personnel are required, increasing labor costs.
- Customer requirements are diverse, and some customer service personnel do not have comprehensive knowledge about the service. As a result, customer satisfaction decreases.
- It is difficult for manual customer service personnel to collect statistics and analyze data upon Q&A, and thus the recorded feedback is not enough to improve product quality.
- A large number of historical service tickets, logs, and cases are not used to extract service experience.

Solution

- Huawei Cloud Conversational Bot Service (CBS) is used to build an intelligent Q&A system for customer service, greatly improving customer service efficiency.
- Seamless transfer to manual processing provides better user experience.
- Closed-loop knowledge management quickly iterates and enriches the knowledge base. Active model learning makes robots smarter as long as it is used.

Benefits

- Over 350,000 Huawei staff are served.
- The time used for building the knowledge base is shortened by 48 times.
- At the early stage of commercial use, the problem hit rate exceeds 85%, and the manual substitution rate reaches 65%.
- CBS handles workloads of 179 human customer service personnel annually.



Contents

1. Huawei Ascend Computing Platform
2. **Huawei Cloud EI Platform**
 - Huawei Cloud EI Open Capability Panorama
 - Huawei Cloud AI Development Platform ModelArts
 - Huawei Cloud General AI Capabilities
 - Intelligent Twins
3. Huawei Device AI Platform

Huawei Cloud AI Explores Industry Best Practices with Customers and Partners

Intelligent Twins

City Transportation Industrial Healthcare Finance Network Scientific research Weather Water Government

800+ Huawei projects



30%+
AI-powered core
application systems

18%
Higher profitability

81 Huawei Confidential

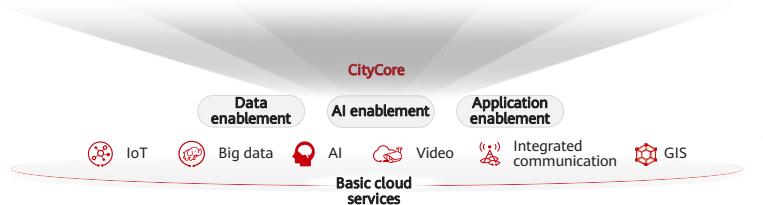


For now, Huawei has provided AI services to 800+ enterprise projects. AI has been used in 30% enterprise business systems and has increased profit by an average of 18%.

- Huawei Cloud AI will continue to explore industry best practices with customers and partners.

Success Stories of City Intelligent Twins

Smart urban management "Clean City" management in Longgang District <ul style="list-style-type: none">AI-based investigation and handling of more than 40 littering incidents every dayA showcase district for the "Clean City" campaign	Smart community Smart community in Guangming District <ul style="list-style-type: none">AI community governance, with dynamic event monitoring and rectification24-hour e-government station, providing self-service handling of six types of services	City IOC Voice assistant in Guangming District <ul style="list-style-type: none">Precise response to voice commandsPrecise audio response to inquiries	Smart Q&A Smart enterprise Q&A in Guangming District <ul style="list-style-type: none">Policy consulting to help clarify doubts about frequent policy changesEnterprise-related affairs in the service hall (500+ items)Requirement issues (various types of requirements)
--	--	--	---



Success Stories of TrafficGo

City-level traffic dashboards

- Infrastructure overview
- Real-time traffic
- Top congested roads
- Congestion alarms
- Incident detection

...

District-level traffic dashboards

- Best optimized roads
- Optimization result evaluation
- Intersection rotation display

...

Intersection traffic dashboards

- Intersection pedestrian density
- Intersection vehicle queue length
- Intersection video

...

Diagnostics dashboard

- Old solution vs New solution
- Entire diagnosis process displayed

...



Based on multi-source, converged data, build multi-dimensional (**macro-mld-micro**) traffic evaluation/metrics systems that are accurate and real-time.



Automatic evaluation of the results of each measure taken. **Intelligent, iterative optimization** in a "sense-understand-diagnose-optimize-evaluate" loop.

- Shenzhen traffic police

Success Stories of Industrial Intelligent Twins

Coking coal blending

Expert experience	Auto-learning of multidimensional parameters	E2E benefits prioritized
Quality prediction precision Cost saved per million tons on production line	> 95%	over CNY20 million

Steelmaking ingredients

Constituent prediction accuracy
85%-90% → 95%

Alloying constituent costs saved
CNY20 million/year

Cutting

Higher board utilization
84%-86% → 86%-88%

Plate cost reduction
40 million/year

Success Stories of EIHealth

Scenarios

Hospital	Pharmaceutical	Research institutes
Target gene prediction		
	Drug repurposing	
		Target gene discovery
		New application creation
Biomarker discovery		
	Drug design	
Biomarker research		
		AI assisted drug design

Real-time inference - AI prediction

DevEnviron - Notebook analysis

85 Huawei Confidential

HUAWEI

Some companies provide drug R&D platforms based on ModelArts to hospitals, pharmaceutical companies, and research institutes for medical research, such as target gene prediction, biomarker research, new drug application, and drug design.

The figures on the right show two instances for AI prediction and notebook analysis on ModelArts.

Success Stories of Financial Intelligent Twins



Automatic identification of insurance policies

Automatic identification and recording of the text in ID cards, bank cards, and medical documents. Handling problems such as misaligned lines, overlapped words, and seal interference in medical records, and applying to various complex scenarios.

Claim information processing time is reduced from **1x minutes** to **seconds**.



Overall recognition accuracy > 95%; average recognition time < 1s



Over 10x efficiency in service processing, faster flow of letter of credit (LC) fund

Over 10-time increase



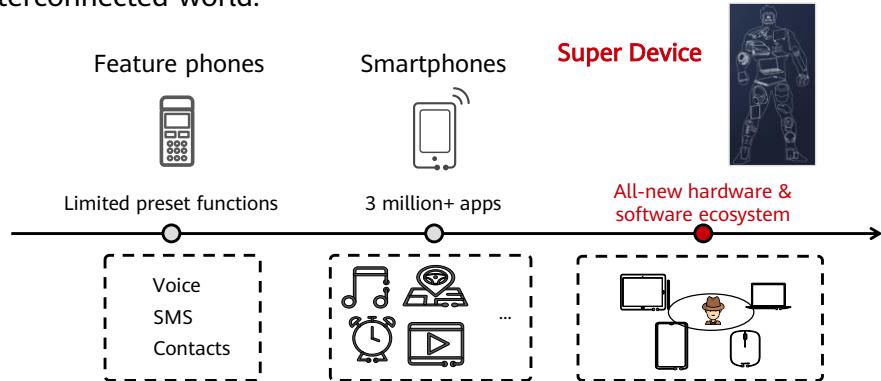
High-precision recognition of images with poor quality, such as rotation and wrinkles

Contents

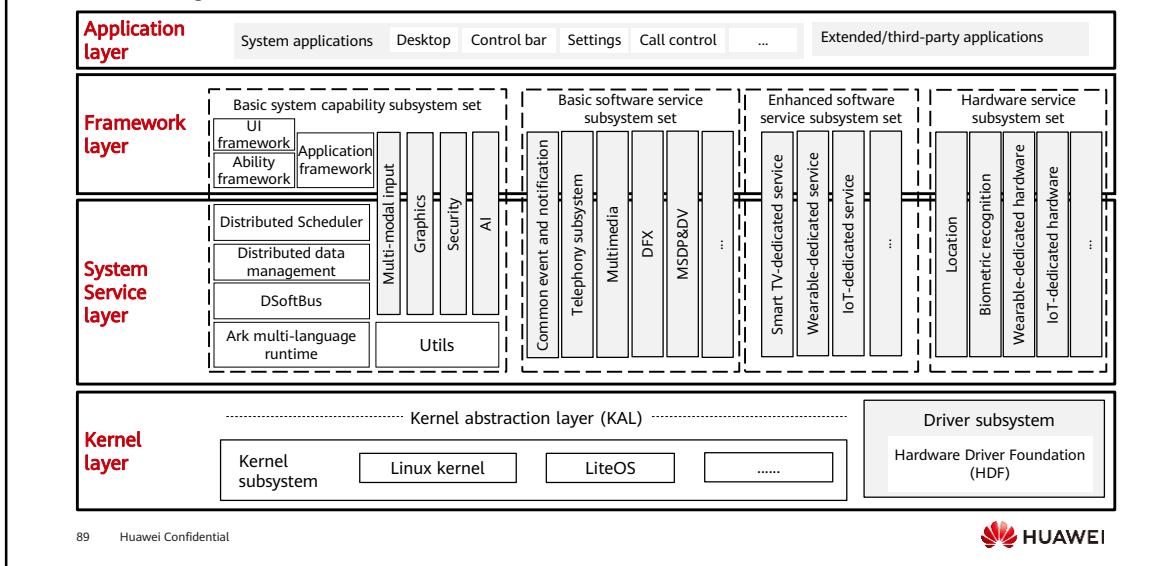
1. Huawei Ascend Computing Platform
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms
 - HarmonyOS
 - HMS Core
 - ML Kit
 - HiAI
 - MindSpore Lite

Introduction to HarmonyOS

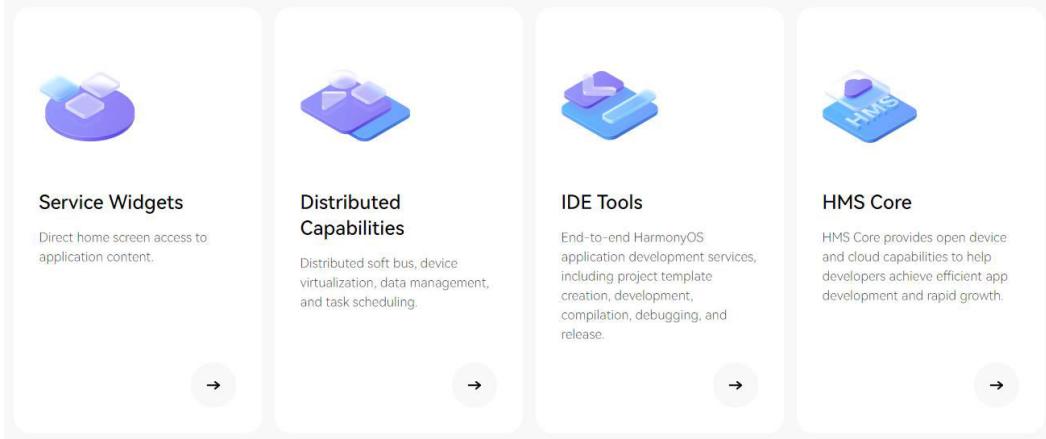
- HarmonyOS is an innovative and distributed operating system designed for an interconnected world.



HarmonyOS Architecture



Key Capabilities of HarmonyOS



90 Huawei Confidential



- <https://developer.harmonyos.com/>

Contents

1. Huawei Ascend Computing Platform
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms
 - HarmonyOS
 - HMS Core
 - ML KIT
 - HiAI
 - MindSpore Lite

HMS Core Introduction

Apps



HMS Core 6

OSs



Devices

Smartphones, computers, smart TVs, smart vehicles, tablets, cameras, smart watches, headsets, home appliances, and smart lock

- The HMS Core can be used not only on Huawei smartphones but also on other OSs.

Capabilities in Seven Domains Lay the Foundation for a Fully Connected Intelligent App Ecosystem

Efficiency	Reliability	Value	Intelligence				
Quick application development for efficient connections	Secure applications for reliable connections	Continuously growing application value for business connections	Intelligent applications for smart connections				
App Service	Graphics	AI	Media	System	Security	Smart Device	
Account Kit Ads Kit Analytics Kit Location Kit Awareness Kit Game Service App Linking Identity Kit In-App Purchases Business Touch Kit DCI Kit Dynamic Tag Manager SignPal Kit	Map Kit Navi Kit Push Kit Quick apps Scan Kit Wallet Kit Search Kit Location Kit Weather Kit Health Kit UI Engine Drive Kit Membership Kit	Accelerate Kit AR Engine VR Engine Computer Graphics Kit Scene Kit GameTurbo Engine 3D Modeling Kit 3D Engine	ML Kit HiAI Foundation HiAI Engine HiAI Service	Audio Kit Audio Engine Image Kit Video Kit Video Engine WisePlay DRM Camera Engine Panorama Kit Audio Editor Kit AV Pipeline Kit Video Editor Kit	hQUIC Kit LinkTurbo Engine Nearby Service Network Kit MDM Engine HEM Kit Haptics Engine 5G Modem Kit	FIDO Safety Detect Keyring LocalAuthentication Engine DataSecurity Engine iTrustee TEE	CaaS Engine Cast Engine DeviceVirtualization Engine OneHop Engine Share Engine Wear Engine HUAWEI HiCar Pencil Engine
IDE/Tools	Reality Studio	Theme Studio	Graphic Profiler	HMS Toolkit	Quick App IDE		

Contents

1. Huawei Ascend Computing Platform
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms
 - HarmonyOS
 - HMS Core
 - ML KIT
 - HiAI
 - MindSpore Lite

ML Kit Facilitates AI App Development With Brand New Experience

Text-related	Language/Voice-related	Image-related	Face/body-related	NLP	Custom models
Text recognition Document recognition Card recognition Form recognition	Translation Language detection Sound detection Text to speech (TTS) Automatic speech recognition (ASR) Audio file transcription Simultaneous interpretation (New)	Image classification Object detection Landmark recognition Image segmentation Product visual search Document skew correction Text-image super-resolution (TISR)	Face detection/verification Hand gesture recognition Skeleton detection Interactive/static biometric verification	Real-time text extraction Text embedding	Custom image classification Custom text classification Custom object detection

Core advantages

Wide device model coverage

All Android, iOS, and HarmonyOS devices are supported.

Global coverage

ML Kit is available around the globe.

Data security

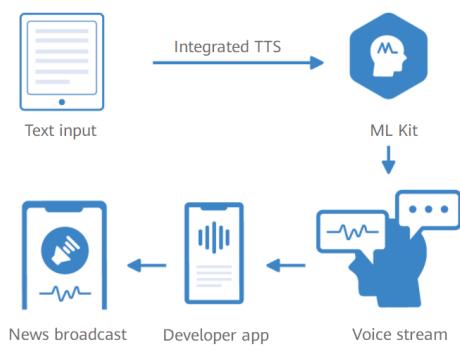
On-device data is not uploaded to the cloud. On-cloud data is stored and operated independently.

Customizable models

You can train and generate custom AI models to meet actual application requirements.

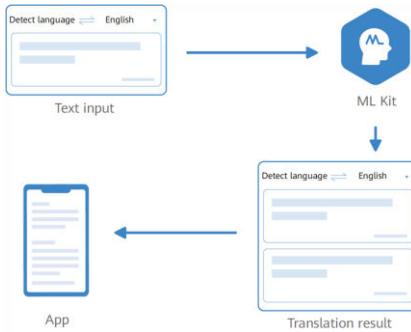


Text to Speech



- Supports Chinese, English, French, German, Italian, and Spanish
- Supports text containing both Chinese and English
- Free of charge for Huawei devices

Translation

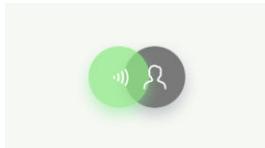


- Quickly detects 52 languages
- Online and offline translation
- Accurate and fast AI enablement
- Template-based integration, which is easy to use

Contents

1. Huawei Ascend Computing Platform
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms
 - HarmonyOS
 - HMS Core
 - ML KIT
 - HiAI
 - MindSpore Lite

HUAWEI HiAI - A Fully Open Intelligent Ecosystem to Drive Developer Business Success

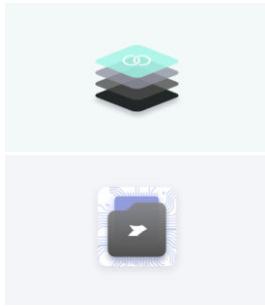


Cloud

HUAWEI HiAI Service

Provides intelligent digital services for an expanding range of application scenarios.

Open service capabilities better connect services and users.



Device

HUAWEI HiAI Engine

Creates an optimal user experience as a portal to the digital world.

Open application capabilities enable more intelligent and powerful apps.



Chip

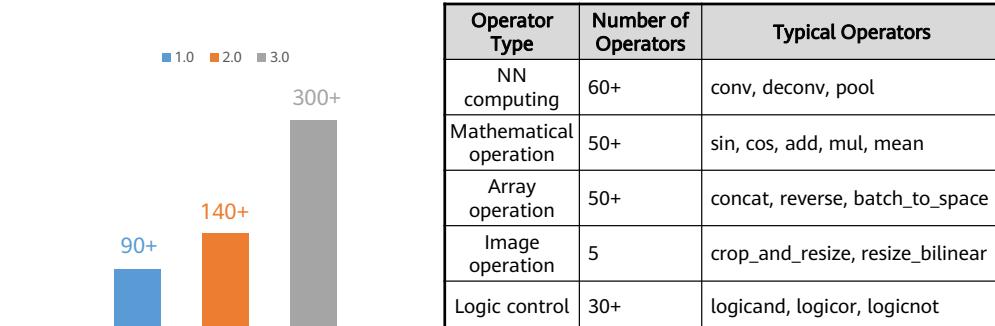
HUAWEI HiAI Foundation

Achieves performance breakthroughs and better power efficiency.

Open chip capabilities improve performance with NPU acceleration.

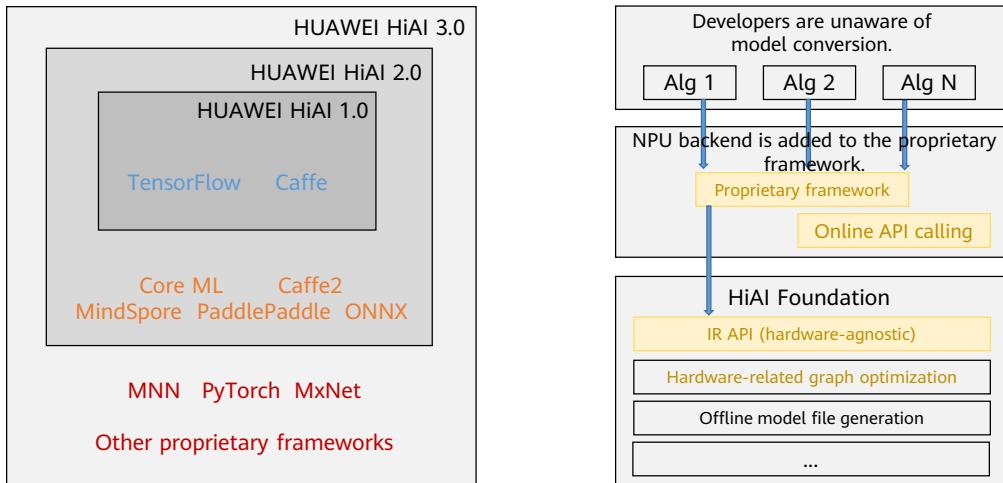
HUAWEI HiAI Foundation - Open Chip Capabilities Enable Powerful On-Device Computing

- Open chip capabilities are providing more and more operators for developers.



- Developers need both powerful computing capabilities and rich operators to build NN algorithms. Operators are like Lego bricks—the more types of bricks and the smaller the granularity, the more complex and realistic the model is. HUAWEI HiAI Foundation is leading the industry with support for more than 240 operators.
 1. NN computing: conv (convolution) /deconv (deconvolution) /pool (pooling)
 2. Mathematical operation: sin (sine) /cos (cosine) /add (tensor addition) /mul (tensor multiplication) /mean
 3. Array operation: concat (concatenation) /reverse/batch_to_space (tensor transformation)
 4. Image operation: crop_and_resize (cropping and resizing) /resize_bilinear (bilinear interpolation)
 5. Logic control: logicand (logical AND) /logicor (logical OR) /logicnot (logical NOT)

Excellent Framework Compatibility Through IR APIs



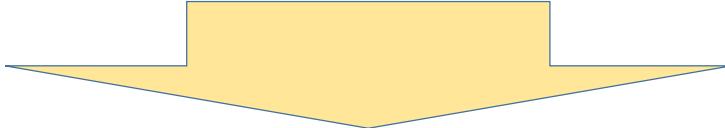
101 Huawei Confidential



- We always listen to developers to resolve their confusions and build more efficient and powerful solutions. As some developers reported that their algorithms are based on proprietary frameworks, which were not supported by HiAI, we introduced IR APIs to connect HiAI to proprietary frameworks, so that developers can leverage the acceleration capability of HiAI without changing their development habits.
- Alg means algorithm.

AI Capabilities for Various Devices

Cars, computers, large screens, tablets, smartphones, earphones, VR glasses, speakers, and smart watches



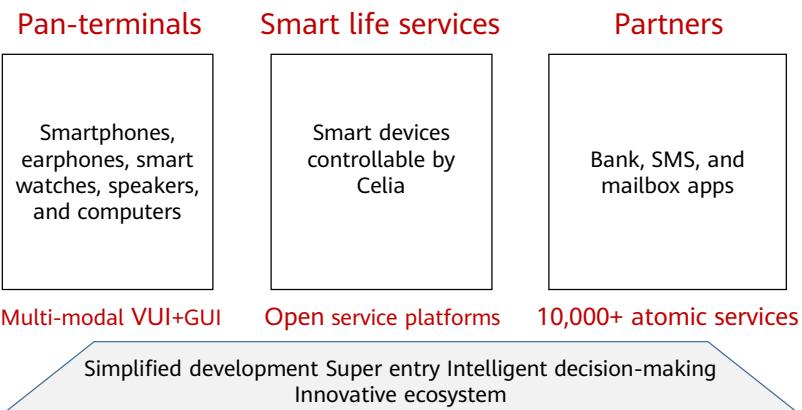
Honghu processors, Kirin processors, and AI camera processors

- HUAWEI Vision products use Honghu processors.
- AI camera processors, such as Hi3559, are processors used by cameras.

HUAWEI HiAI Engine – Rich AI Capabilities and Ultimate Experience Out of the Box

CV				NLU	ASR		
Text recognition	Image recognition	Facial recognition	Body recognition	Video technology	Code recognition	NLP	Speech recognition
General text recognition Form recognition Passport recognition ID card recognition Driving license recognition Vehicle license recognition Document converter Bank card recognition	Aesthetics score labeling Image category labeling Image super-resolution Scene detection Document skew correction Text-image super-resolution (TISR) Portrait segmentation Semantic segmentation	Facial comparison Face detection Face parsing Face attribute recognition Face orientation recognition Facial feature detection	Key skeletal feature recognition Video portrait segmentation	Video summarization Video thumbnail	Code recognition	Word segmentation Part-of-speech tagging Assistant-specific intention recognition IM-specific intention recognition Keyword extraction Entity recognition	Speech recognition

HUAWEI HiAI Service: One-Time Integration, Multi-Modal and Multi-Device Deployment



104 Huawei Confidential



- HiAI Service provides two platforms: The first is HUAWEI Ability Gallery (HAG), which aggregates developer content and services to provide Huawei users with direct access to third-party services. More than 10,000 atomic services are now available to users. The second is the Celia dialog development platform, which provides developers with end-to-end dialog service development, test, and deployment. It is a one-time integration and multi-modal multi-device deployment solution.

Contents

1. Huawei Ascend Computing Platform
2. Huawei Cloud EI Platform
3. Huawei Device AI Platforms
 - HarmonyOS
 - HMS Core
 - ML KIT
 - HiAI
 - MindSpore Lite

Introduction to MindSpore Lite

- MindSpore Lite is an ultra-fast, intelligent, and simplified AI engine that enables intelligent applications in all scenarios, provides end-to-end solutions for users, and helps users enable AI capabilities.

MindSpore Lite Users

HMS ML Kit

Use the machine learning kit provided by Huawei to quickly develop on-device machine learning applications.

[View More](#)

HUAWEI HiAI

An open AI capability platform for smart devices, thus accelerating your development cycle and making apps smarter.

[View More](#)

HUAWEI SiteAI

SiteAI builds a leading lightweight, efficient, safe and easy-to-use, three-layer collaborative embedded AI platform to enable intelligent network elements and autonomous driving networks.

- MindSpore Lite focuses on lightweight inference but can also perform training.
- <https://mindspore.cn/lite/en>

MindSpore Lite Features

Ultimate performance

- Provides efficient kernel algorithms and assembly-level optimization, and supports CPU, GPU, and NPU.
- Provides heterogeneous scheduling, maximizes hardware computing power, and minimizes inference latency and power consumption.

Lightweight

- Provides an ultra-lightweight solution to support model quantization and compression.
- Provides small AI models that run fast and can be quickly deployed in extreme environments.

All-scenario support

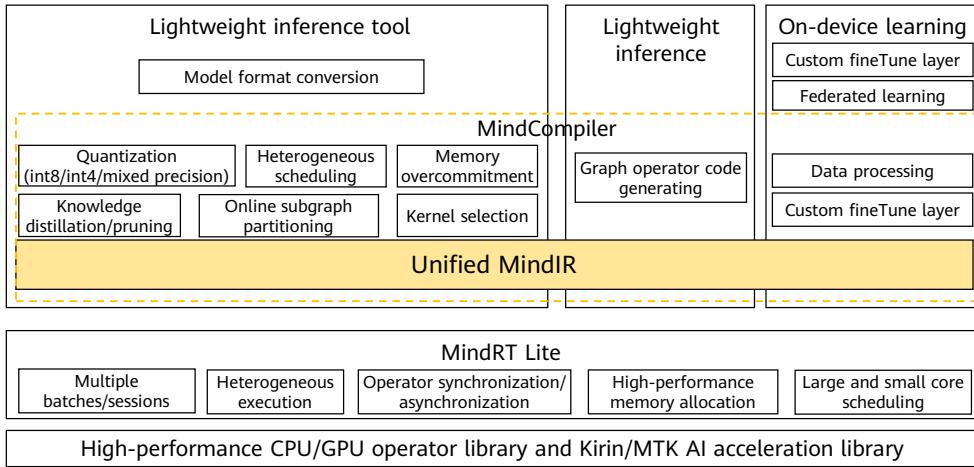
- Supports mobile phone operating systems such as iOS and Android, LiteOS embedded operating system, and AI applications on various intelligent devices such as mobile phones, large screens, tablets, and IoT devices.

Efficient deployment

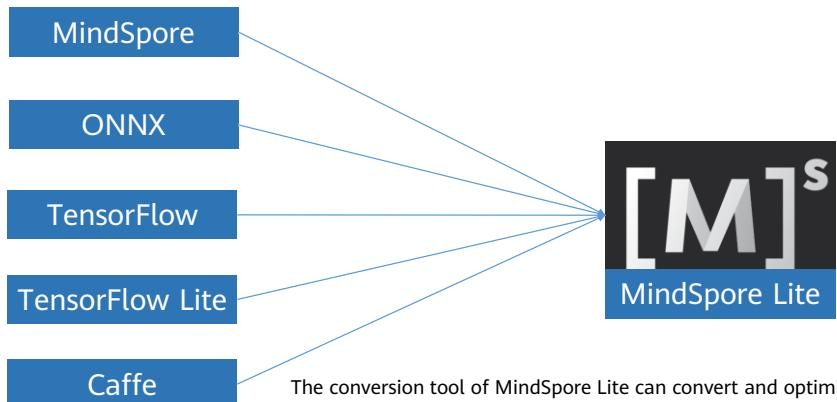
- Supports models such as MindSpore, TensorFlow Lite, Caffe, and ONNX, provides capabilities such as model compression and data processing, and supports unified training and inference IR, facilitating quick deployment.

- MindSpore Lite provides model compression and conversion tools to meet the performance restrictions of devices and support multiple OSs and frameworks. This ensures high accuracy and improved inference speed while reducing hardware requirements.

MindSpore Lite Architecture



Compatibility With Mainstream AI Frameworks



The conversion tool of MindSpore Lite can convert and optimize models of mainstream AI frameworks to seamlessly support on-device learning and high-performance inference of models trained by MindSpore.

Quiz

1. What is the architecture used by the Ascend processor?
2. What is the functional unit responsible for image preprocessing in CANN?
3. Does using Huawei Cloud EI require programming basics?
4. How many AI capabilities does HMS provide?

- 1. Da Vinci
- 2. DVPP
- 3. No. You do not need to write code when using ExeML and AI Gallery algorithms.
- 4. Four, including ML KIT and three capabilities of HIAi.

Summary

- This chapter introduces Huawei's full-stack AI solution, including hardware products (Ascend processors and Atlas series devices), software (CANN), public cloud service (Huawei Cloud EI), and on-device products (ML Kit and HiAI).

Recommendations

- Ascend Developer Community
 - <https://www.hiascend.com/>
- Official MindSpore website
 - <https://www.mindspore.cn/en>
- HarmonyOS application development official website
 - <https://developer.harmonyos.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。

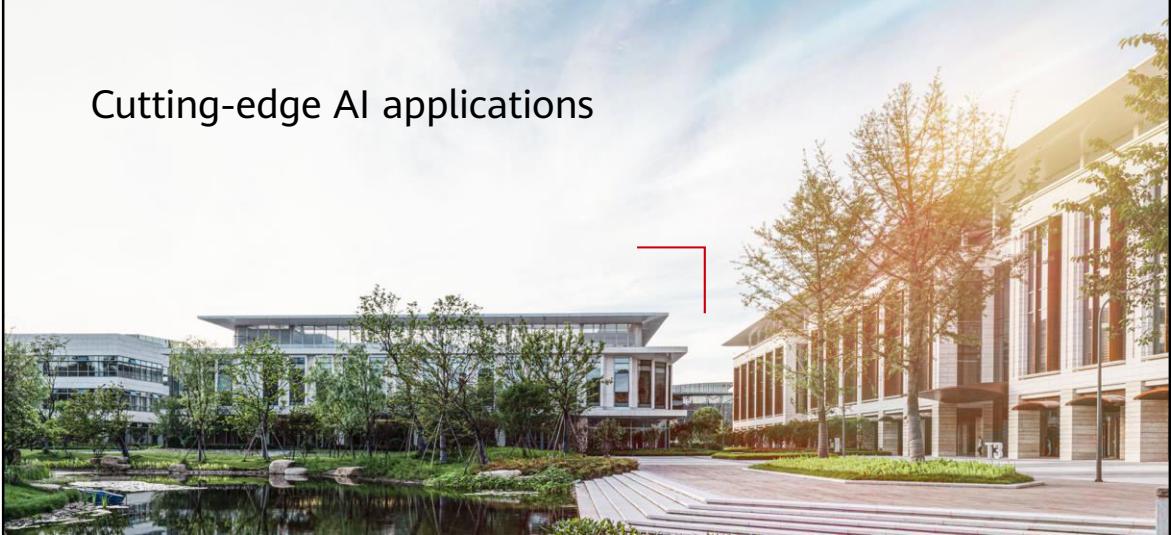
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2023 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.



Cutting-edge AI applications



Objectives

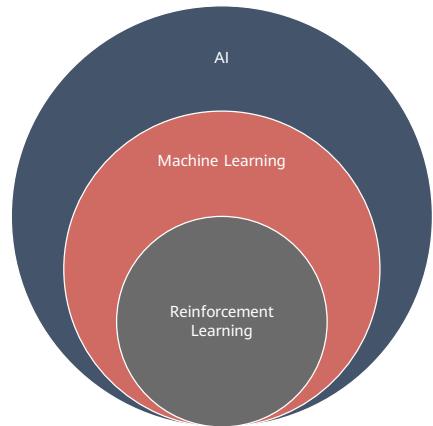
- Upon completion of this course, you will be able to:
 - Understand the concepts and applications of reinforcement learning;
 - Understand the concepts and applications related to the GAN;
 - Understand the concepts and applications related to knowledge graph;
 - Understand the concepts and applications of intelligent driving.

Contents

- 1. Reinforcement Learning**
2. GAN
3. Knowledge Graph
4. Intelligent Driving

Reinforcement Learning

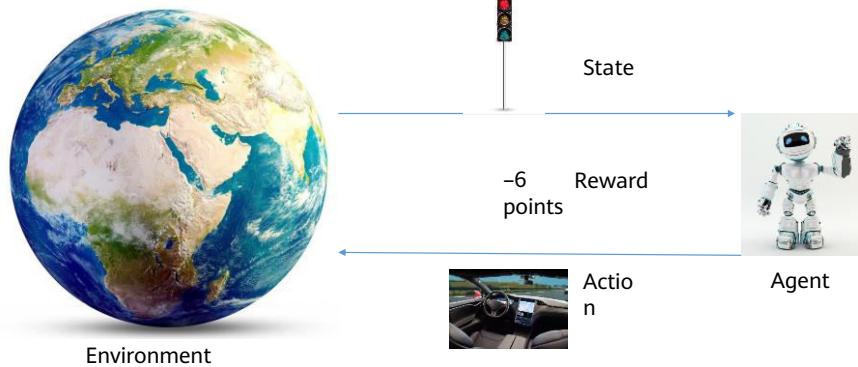
- Reinforcement learning (RL) is a branch of machine learning that emphasizes how to act based on the environment in order to maximize the expected benefits.



- The idea of reinforcement learning stems from the study of behavioral psychology. In 1911, Thorndike proposed the Law of Effect: The behavior that makes animals feel comfortable in a certain situation is closely connected to the situation. When the situation reappears, the behavior of animals is more likely to reappear. On the contrary, the behavior that makes an animal feel uncomfortable weakens the connection with the situation, which will be difficult to reproduce when the situation reappears. In other words, behavior will be remembered depending on the utility of that behavior. For example, when the owner throws out the Frisbee, the dog takes the Frisbee back to the owner and obtains a flesh bone, which will reinforce the connection between the behavior of "taking back the thrown Frisbee" and the situation of "the owner throwing out the Frisbee". The utility of "getting flesh bones" will make the dog remember the behavior of "taking back the thrown Frisbee".
- In a given situation, the rewarded behavior will be "reinforced" and the punished behavior will be "weakened". Such a bio-intelligent model allows animals to learn from rewards or penalties for different behavior attempts and choose the behavior that the trainer expects most in that situation. This is the core mechanism of reinforcement learning: trial-and-error, which is used to learn to choose the most appropriate behavior in a given situation. Sutton defines reinforcement learning as how to best match states and actions through trial-and-error learning and obtain the maximum rewards.

- Reinforcement learning directly imitates the intelligent mode of biological learning. Unlike most other machine learning methods where agents need to be told which action to choose, reinforcement learning enables agents to try different actions, autonomously discover and choose actions that generate the maximum rewards. As Tesauro describes, reinforcement learning enables agents to learn autonomously based on their own experience, without any preparatory knowledge or the help of any external intelligent "teacher".

Environment Sensing

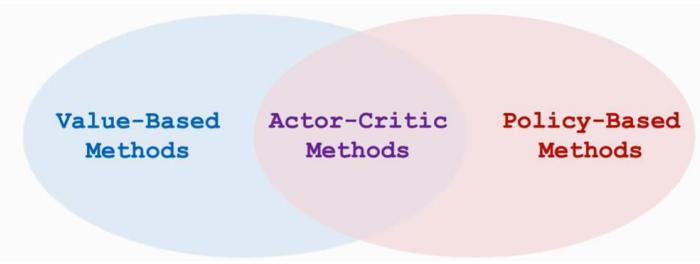


Basic Concepts in Reinforcement Learning

- Two objects:
 - Agent
 - Environment
- Main elements:
 - Action: performed at the current moment based on the status.
 - Policy: makes decisions based on the status and controls the agent to perform actions.
 - Reward: provided by the environment based on the current action.
 - Return: sum of rewards at all moments.

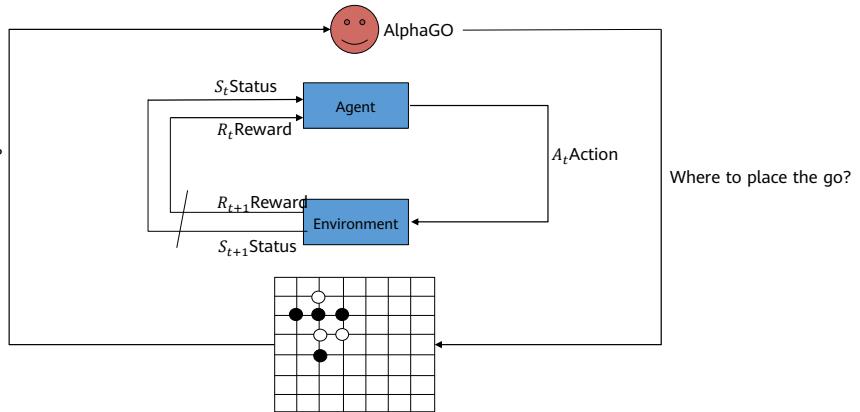
Classification of Reinforcement Learning Algorithms

- The reinforcement learning algorithms can be divided into two directions based on the process of finding the optimal policy:
 - Direct solution: The optimal policy function is optimized during the interaction with the environment.
 - Indirect solution: This type of algorithms are the most common algorithms. They indirectly calculate other indicators.



Reinforcement Learning Model - Go

What is the situation?
Who is winning?



Where to place the go?

Contents

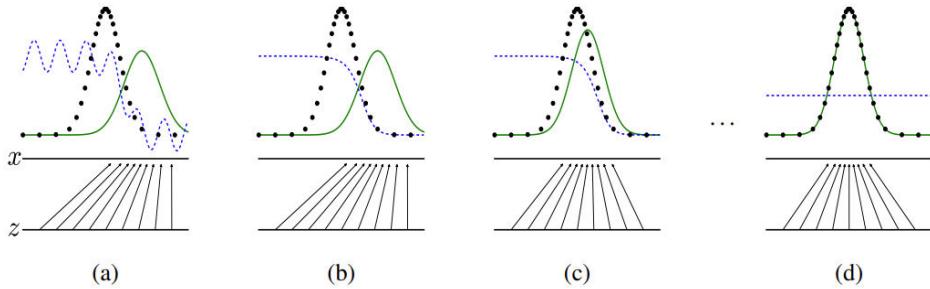
1. Reinforcement Learning
- 2. GAN**
3. Knowledge Graph
4. Intelligent Driving

Generative Adversarial Network (GAN)

- A GAN is a class of frameworks that trains generator G and discriminator D to compete in a game. The game between the two makes the discriminator D unable to distinguish whether the sample is a fake sample or a real sample.
 - The generator G generates "fake" images that look like the images for training.
 - The discriminator D determines whether the images output by the generator are real images or fake images.
- GANs are used in scenarios such as image generation, text generation, speech enhancement, and image super-resolution.

- Generator G: Input "noise" z (z complies with a selected prior probability distribution, such as uniform distribution and Gaussian distribution). Multi-layer perceptrons with the maximum likelihood prediction (MLP) parameters are used to represent the derivative mapping $G(z)$ and map the input space to the sample space.
- Discriminator D: Input real sample x and fake sample $G(z)$ with labels "real" and "fake", respectively. The discriminator network uses the multi-layer perceptions with the MLP parameters to output the probability $D(G(z))$ for determining whether a sample is a real sample.

GAN Training Process



- In the preceding figure, the blue dotted line indicates the discriminator, the black dotted line indicates the real data distribution, the green solid line indicates the false data distribution generated by the generator, z indicates the implicit vector, and x indicates the generated fake image $G(z)$.
 - At the beginning of the training, the quality of the generator and discriminator is poor. The generator randomly generates a data distribution.
 - The discriminator optimizes the network by calculating the gradient and loss function. The data close to the real data distribution is determined as 1, and the data close to the data distribution generated by the generator is determined as 0.
 - The generator generates data that is closer to the actual data distribution through optimization.
 - The data generated by the generator reaches the same distribution as the real data. In this case, the output of the discriminator is 1/2.

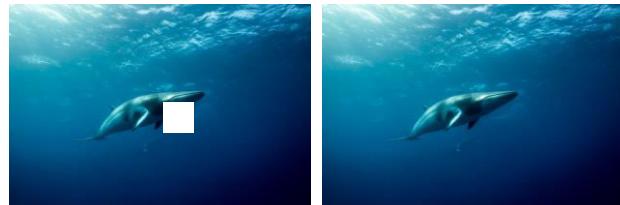
GAN Application

- Image dataset generation
- Image-to-image conversion
- Resolution enhancement, making photos clearer
- Text-to-image conversion

- AI training requires a large number of datasets, but manual data collection and labeling is expensive. GAN can automatically generate some datasets to provide low-cost training data.
- Converting one image into another one is like adding a filter: converting a photo into an oil painting.

GAN - Photo Repair

- If an area in a photo is faulty (for example, colored or erased), GAN can repair the area and restore it to its original state.



Contents

1. Reinforcement Learning
2. GAN
- 3. Knowledge Graph**
4. Intelligent Driving

Basic Concepts

- Knowledge graph is essentially a knowledge base of the semantic network. It describes various concepts, entities, and relationships in the real world in a structured manner to form a huge semantic network. Nodes in the network are entities, and each edge represents a property of entities or the relationship between entities.
- Entity: an object that exists in the real world and can be distinguished from other objects.
- Property: The nature and relationship of a specific object are called the properties of the object.
- Concept: a set of entities with the same features.
- Ontology: a set of abstract concepts used to describe the common features of all the things in a domain and the relationships between them.



 HUAWEI

- Semantic network, proposed in the 1960s, is a structured way to express knowledge with graphs. A knowledge graph consists of interconnected nodes and edges. A node indicates a concept or object, and an edge indicates a relationship between them.
- The concept of knowledge base comes from two different fields, one is artificial intelligence and its branch—knowledge engineering, and the other is the traditional database field. It is an organic combination of artificial intelligence and database technologies. The knowledge base is a knowledge-based intelligent system.
- <https://www.huaweicloud.com/product/nlpkg.html>
- An example is as follows:
- Concept: Country
- Entity: China, Russia...
- Property: Capital – Connected entity (the capital of China is): Beijing – (Entity) Property of Beijing: latitude, longitude... -- Data: longitude: E 116° 20'; latitude: N 38° 56'
- The basic unit of knowledge graph is a triplet of entity-relationship-entity.
- Ontology is a set of concepts.

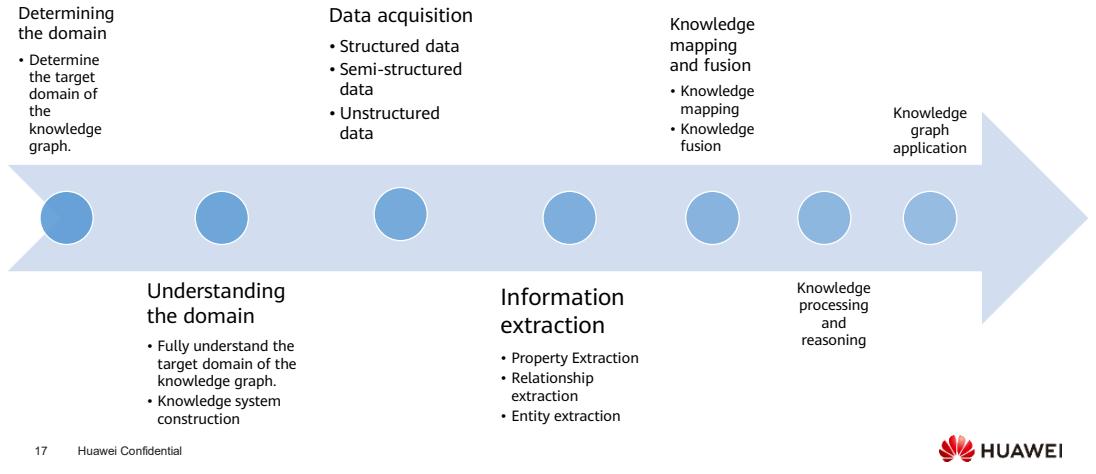
History of Knowledge Graph

- Knowledge graph is a relatively mature application "fruit" of knowledge engineering in the current stage of AI development.
- Five phases of knowledge engineering:
 - Pre-knowledge engineering period (1950s–1970s)
 - Expert system period (1970s–1990s)
 - World Wide Web 1.0 period (1990s–early 21st century)
 - Swarm intelligence period (2000–2006)
 - Knowledge graph period (2006–present)
- In the knowledge graph period, the vigorous development of large-scale structured encyclopedia websites and the continuous progress of text information technology provide conditions for obtaining large-scale knowledge.
- Google took the lead in applying knowledge graphs to search engines in 2012, successfully improving users' search quality and experience.

- For details about the development history, see pages 3–5 in *Artificial Intelligence - Knowledge Graph*.

Process for Constructing a Knowledge Graph

- The general knowledge graph construction process is as follows:



- Determine the domain: A good knowledge graph should be domain-specific.
- Knowledge mapping: Knowledge mapping is to establish the mapping between the structured information extracted from basic data and the knowledge graph ontology, for example, the mapping between the extracted structured information about movies and the graph ontology. In other words, two data models are established, and the elements of models are linked to form a mesh structure by using related languages or technologies. For example, you can introduce graph database knowledge by mapping from a relational database to a graph (or a non-relational database, such as a graph database).
- Knowledge fusion: Data processing and entity disambiguation are performed when data overlaps or is inconsistent.
- For more information, please visit https://support.huaweicloud.com/productdesc-kg/kg_02_0004.html#kg_02_0004_fig895816278185.

Knowledge Graph Application Scenario (1)

The common application scenarios are as follows:

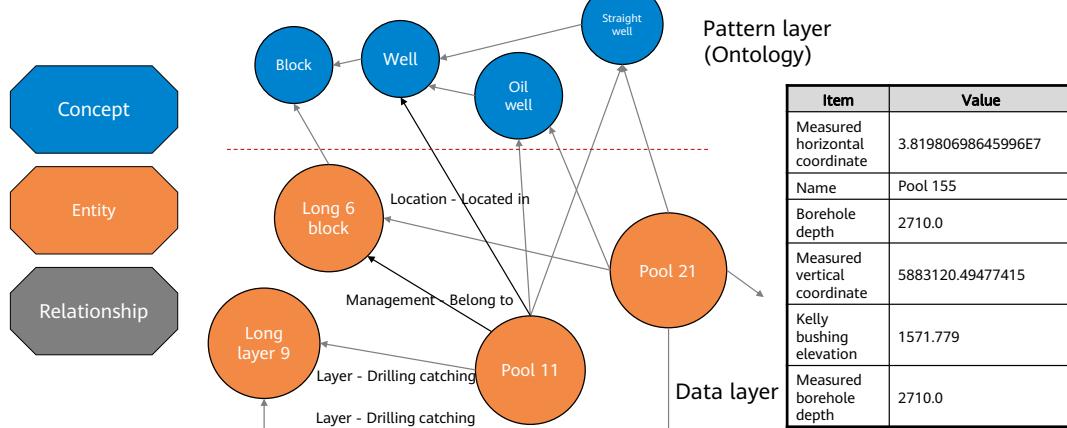
The diagram illustrates four common application scenarios for knowledge graphs:

- Precise recommendation:** Shows a mobile interface where a central video player is surrounded by a grid of smaller video thumbnails. A blue box labeled "Precise recommendation" points to this interface. Below it, the text reads: "Music apps accurately recommend songs that you like to listen to."
- Semantic search:** Shows a screenshot of the Petal search engine interface. A blue box labeled "Semantic search" points to this interface. Below it, the text reads: "Exact keyword matching by the Petal search engine".
- Intelligent Q&A:** Shows a list of major vendors: Huawei Xiaoyi, Apple Siri, Microsoft Cortana, and Baidu Xiaodu. To the right, three boxes represent underlying technologies: "Retrieval technology", "Knowledge graph", and "Deep learning". A dashed line connects the Intelligent Q&A section to the technology boxes.
- Music recommendation:** Shows a mobile interface with a central video player and surrounding thumbnails. A blue box labeled "recommendation" points to this interface. Below it, the text reads: "Music apps accurately recommend songs that you like to listen to."

- <https://consumer.huawei.com/cn/mobileservices/music/>
- Precise recommendation: The recommendation system works with the knowledge graph to generate user profiles for precise recommendation.
- Semantic search: Huawei Petal and other search engines can accurately match the knowledge you search for.
- Intelligent Q&A: Siri, Xiaodu, Cortana...

Knowledge Graph Application Scenario (2)

Oil and Gas Knowledge Computing



19 Huawei Confidential



- An example graph in the oil field shows entity types such as oil wells, reservoirs, and blocks. Oil wells have their own property information.
- By constructing such a unified knowledge graph, you can provide a unified knowledge base entry, providing functions such as semantic search and Q&A, and use it in models of other scenarios based on graph representation.

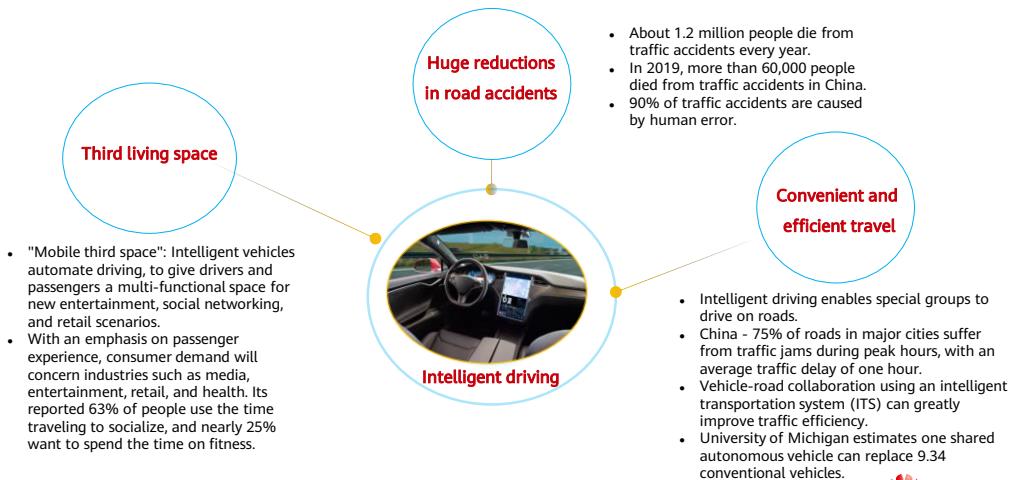
Contents

1. Reinforcement Learning
2. GAN
3. Knowledge Graph
- 4. Intelligent Driving**

History of Intelligent Driving

- 1995 – Navlab, Carnegie Mellon University
 - The Rapidly Adapting Lateral Position Handler (RALPH) vision system
 - Autonomous driving of 4,496 km (98.1% of the whole trip) during the long-distance driving experiment
 - Test environments included morning, night, and extreme weather conditions such as rainstorm.
- 1995 - Mercedes-Benz VITA developed by Ernst Dickmanns
 - Pioneered the 4D spatiotemporal dynamic vision model
 - Could drive at 130 km/h on highways
- 1996- ARGO autonomous, University of Parma
 - Built on a general-purpose chip and low-cost cameras
 - Autonomous driving for 94% of the 2,000 km trip, at speeds up to 112 km/h
- 1996-2000 - ATB-2 co-developed by NJUST, BIT, THU, ZJU, and NUDT
 - Developed on Mercedes-Benz Sprinter 414
 - Oriented to structured roads and off-road environments
 - Maximum speed of 74 km/h on structured roads
- 2004 - The first DARPA Grand Challenge
 - Autonomous driving of 240 km route across the Mojave Desert, USA.
 - Carnegie Mellon University's Sandstorm traveled the farthest distance, completing 11.78 km of the course.
 - Some vehicles were able to avoid obstacles but required large, expensive sensing systems.
- 2005 - The second DARPA Grand Challenge
 - The Mojave Desert 212 km off-road route was used again, but had rougher conditions.
 - Competitors were much more successful than 2004, with Stanford University's Stanley traveling the farthest in the shortest time. This challenge observed the first prototypes of intelligent driving vehicles.
- 2007 - The third DARPA Grand Challenge
 - The urban challenge of 96 km, with vehicles ranked on obeying all traffic regulations and avoiding collisions.
 - Carnegie Mellon University's car Ross ranked first.
- DARPA races foster the development of intelligent vehicles.
- 2009 - The first Intelligent Vehicle Future Challenge of China
 - The race involved obstacles, traffic lights, and hairpin turns.
 - Teams from HNU, BIT, SJTU, XITU, and University of Parma competed in the challenge.

Intelligent Driving Will Lift Society to New Heights

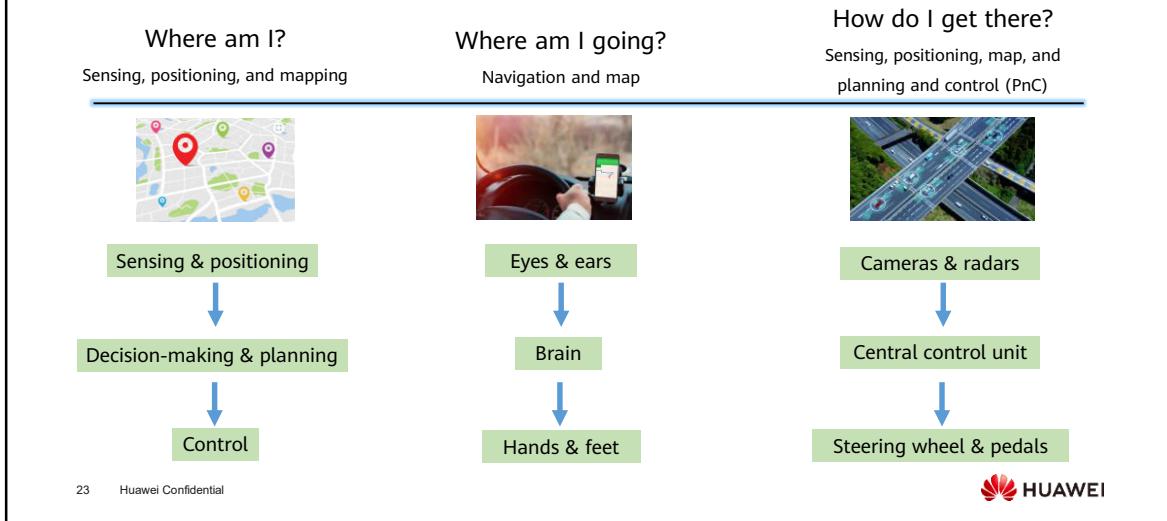


22 Huawei Confidential



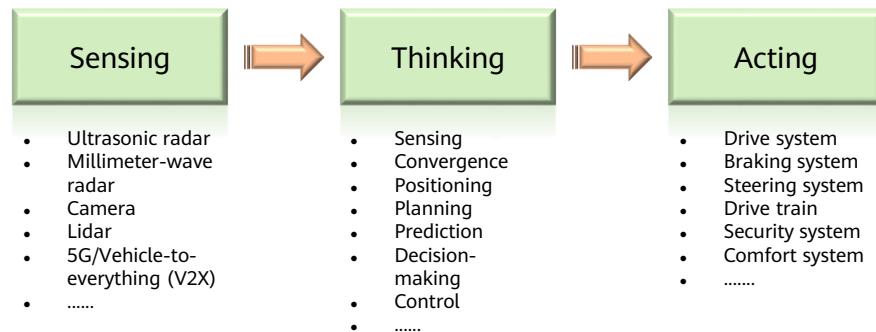
- Continuous iteration for zero accident and zero casualty: More than 90% of accidents are caused by human errors and drunk driving. Convenient travel: Intelligent driving improves traffic efficiency, reduces driving and parking time, and enables the elderly, disabled, and children to travel conveniently. The third living space: Passengers are completely free during travel and can rest, work, and entertain in vehicles.
- Image source:
http://3ms.huawei.com/multimedia/docMaintain/mmMaintain.do?method=showMMDetail&f_id=img202008150653

Manual Driving vs. Intelligent Driving

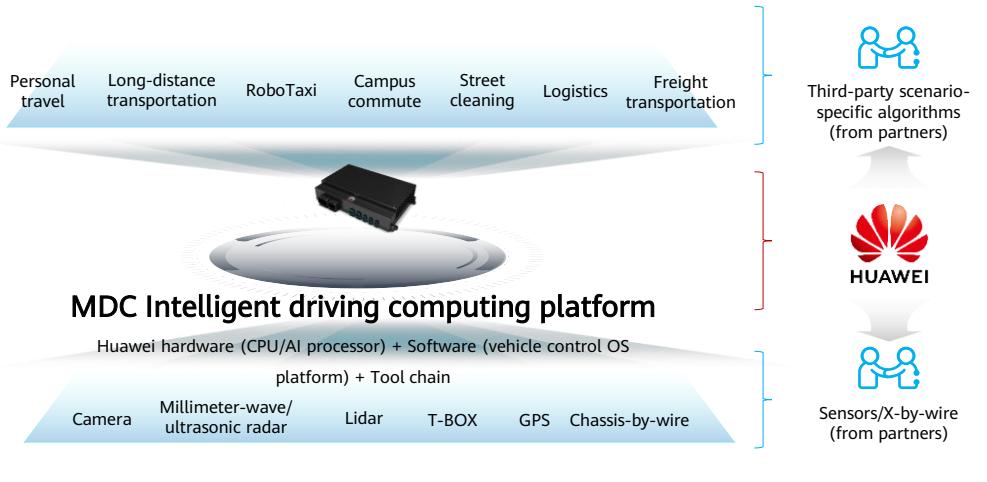


- Left: <http://3ms.huawei.com/km/static/image/detail.html?fid=63381>
- Middle: smartvi
- Right: <http://3ms.huawei.com/km/static/image/detail.html?fid=61786>

Three Subsystems of Intelligent Driving



Huawei MDC: Leading Intelligent Driving Computing Platform



25 Huawei Confidential

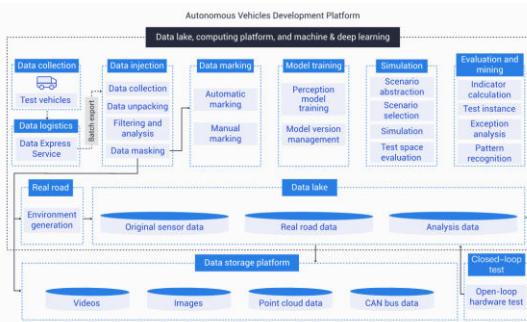
Huawei MDC intelligent driving computing platform accelerates commercial launch of intelligent driving vehicles through a "platform + ecosystem" approach.

Huawei MDC consists of the hardware platform, software platform, and tool chain. The software platform is compatible with AUTOSAR.

The hardware platform can be interconnected with products of multiple sensor and X-by-wire partners; the software platform supports application algorithms of different partners, such as sensing, convergence, positioning, decision-making, planning, and control. These algorithms enable various intelligent driving scenarios, such as long-distance transportation and campus commute.

Huawei MDC leverages the power of the platform and ecosystem to reduce costs and quickly implement scenario-based intelligent driving.

Autonomous Driving Cloud Service: One-Stop Development Platform for Autonomous Vehicles



- **Data services:**

- Data storage management, processing pipeline, overview, and playback, as well as an annotation platform

- **Training services:**

- Training tasks, algorithm and model management, algorithm iteration, and model evaluation

- **Simulation services:**

- Simulated scenarios, online simulations, task management, and simulation algorithms

- **Competitive advantages:**

- Annotation platform
- Model training
- Parallel simulation



- **Data service:** Data services of Huawei Octopus provide automatic processing of raw data collected by autonomous vehicles, facilitating data management, KPI statistics collection, and problem point playback and locating. In addition, Octopus can create new training datasets through the annotation platform.
 - **Data storage:** PB-level massive data storage supports data import from OBS and physical drives.
 - **Data processing pipeline:** The pipeline provides automatic RosBag data parsing, playback transcoding, and statistics analysis.
 - **Data overview:** The takeover dashboard and data overview dashboard allow you to quickly view fleet data and track overview.
 - **Data playback:** Playbacks contain detailed image information about sensor data during drive tests.
 - **Annotation platform:** Online annotation includes manual labeling and automatic labeling, improving labeling efficiency.
 - **Data management:** Major data sets in the industry, such as Pascal VOC, can be generated, imported, and managed.

Summary

- This chapter briefly introduces concepts of several cutting-edge AI technologies: obtaining the optimal policy through reinforcement learning, the game between the generator and discriminator (GAN), the construction process and applications of knowledge graphs, and brief introduction to intelligent driving.

Quiz

1. (Multiple-answer) Which of the following scenarios can Knowledge Graph be used? ()
 - A. Search engine
 - B. Product keyword search
 - C. Q&A bot
2. (True or false) In intelligent driving, lidar is the only sensing device. ()
 - A. True
 - B. False

- Answers: 1. ABC 2. False

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©2023 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.



Quantum Computing and Machine Learning



Contents

1. Introduction to Quantum Computing

- What Is Quantum?
 - Basic Concepts of Quantum Mechanics
 - Development of Quantum Computing
 - Application Scenarios and Advantages of Quantum Computing

2. Basic Concepts of Quantum Computing

3. Quantum Machine Learning

4. Quantum Computing Software

What Is Quantum?

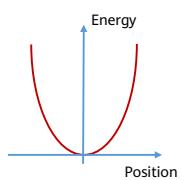
Quantum is the minimum amount of any physical entity in a microphysical system, such as energy, angular momentum and electric charge.

	Classical physics	Quantum physics
Energy	The physical quantity is considered continuous and the system energy can be any value. For example, a ball is released at any position at the beginning, and the system energy is a smooth and continuous parabola.	The physical quantity is quantized and the system energy can only be a discrete value. For example, the energy of a quantum harmonic oscillator can only be $\hbar\omega/2$ or $3\hbar\omega/2$, and the difference between the energy values must be an integer multiple of $\hbar\omega$.
State of motion	The motion state in the system can be completely determined and predicted. For a classical harmonic oscillator, its kinetic energy (velocity) and potential energy (position) can be 0 at the same time. For example, the position of the ball to the spring is balanced, and the system energy is 0.	The uncertainty principle of quantum mechanics shows that we cannot determine the precise position and momentum of a system at the same time. For quantum harmonic oscillators, the kinetic energy (velocity) and potential energy (position) cannot be 0 at the same time. Therefore, the system energy has the minimum value $\hbar\omega/2$.
Property	It is believed that microparticles can be composed of particles.	Quantum mechanics shows that particles have both the wave nature (wavefunction superposition) and particle nature (wavefunction collapse).

What Is Quantum?

Classical physics

$$U = \frac{1}{2} kx^2$$



Harmonic oscillator energy
(classical theory)

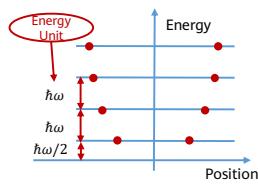
For example, the relationship between the system energy of a small ball (harmonic oscillator) bound to a spring and its initial position can be expressed by a parabolic curve.



Classical oscillator

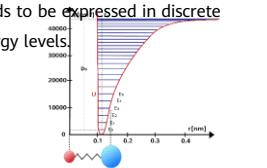
Quantum physics

$$E_n = \hbar\omega \left(n + \frac{1}{2} \right) \quad n = 0, 1, 2, \dots$$



Harmonic oscillator energy
(quantum theory)

For example, the vibration of a diatomic molecule can be approximated as a quantum harmonic oscillator, and its vibrational energy needs to be expressed in discrete energy levels.



Energy levels of molecular vibration of hydrogen chloride

Basic Concepts of Quantum Mechanics



Quantum Superposition

- Similar to waves in classical physics, the result of superposition of arbitrary quantum states will be an effective quantum state.
- Quantum superposition embodies the wave properties of quantum systems.
- Mathematically, quantum superposition comes from the linearity of the Schrodinger wave equation, that is, the linear combination of equation solutions is also the solution of the equation.
- Macro manifestation: double-slit interference pattern

Quantum Entanglement

- Two particles can be in an entangled state: when the first particle spins upward, the second particle spins downward, and vice versa. Neither of the particles has a definite spin orientation before the measurement.
- Now make the two particles in the entangled state leave each other in the opposite direction. Regardless of the distance, the spin orientation of the other particle can be determined immediately after the spin orientation of one particle is measured.

Quantum Uncertainty

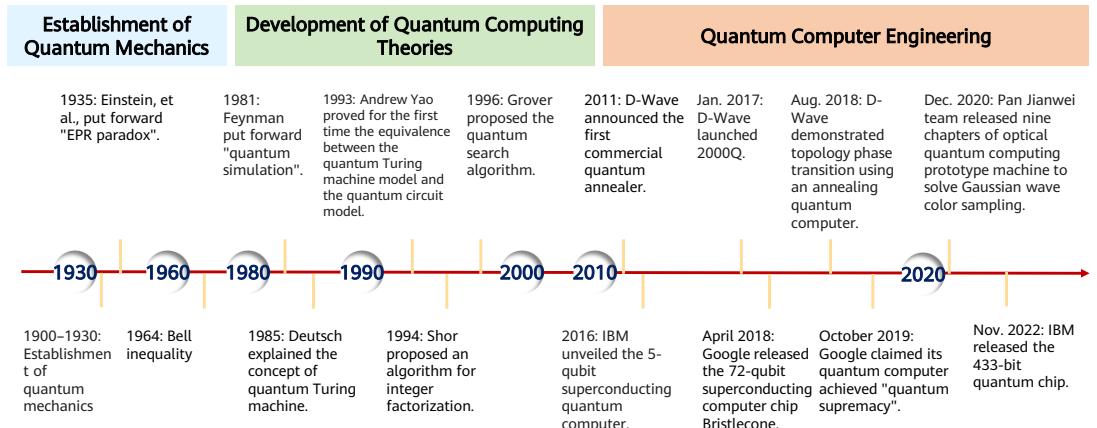
- The position and momentum of a particle cannot be determined at the same time. The smaller the uncertainty of the position, the greater the uncertainty of the momentum, and vice versa.
- Mathematical expression: $\Delta x \Delta p \geq \frac{\hbar}{2}$

<http://3ms.huawei.com/km/static/image/detail.html?fid=56957>

<http://3ms.huawei.com/km/static/image/detail.html?fid=60352>

<http://3ms.huawei.com/km/static/image/detail.html?fid=62577>

Development of Quantum Computing



Application Scenarios and Advantages of Quantum Computing



AI

The quantum-classical hybrid architecture is used to give full play to the advantages of quantum computing in the **NISQ phase**.



Fintech

The quantum optimization algorithm can be used to quickly search for the optimal solution in the **exponential parameter space**.



Biological computing

The functions of biological macromolecules and drug molecules can be **simulated more accurately** by general-purpose quantum computers.



Logistics and transportation

Quantum Approximate Optimization Algorithm (QAOA) can be used to find the optimal solution in **polynomial time**.



Material simulation

Quantum algorithms are used to calculate the energy potential energy surface of molecules, and the maximum **exponential acceleration** can be achieved.



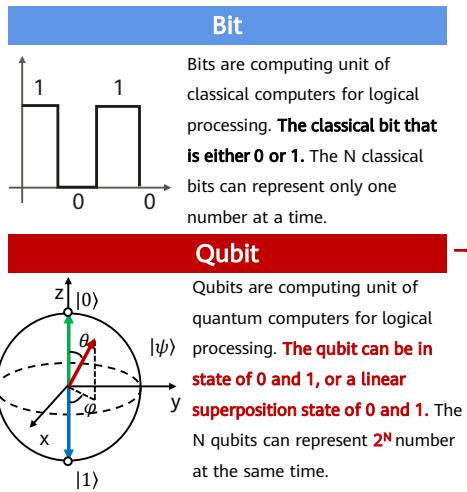
Security

The Shor algorithm can get results in **polynomial time**, whereas the classical algorithm needs **exponential time**. The Shor algorithm can easily crack the existing RSA encryption system, whereas the quantum encryption algorithm has absolute security.

Contents

- 1. Introduction of Quantum Computing**
- 2. Basic Concepts of Quantum Computing**
 - Quantum Bit and Quantum State
 - Quantum Gate
 - Quantum Circuit
 - General-Purpose Quantum Algorithm
 - Variational Quantum Algorithm
- 3. Quantum Machine Learning**
- 4. Quantum Computing Software**

Quantum Bit and Quantum State



9

Huawei Confidential

Base vectors for quantum computing: state 0 is denoted as $|0\rangle$ and state 1 is denoted as $|1\rangle$ (\rangle) is a right vector symbol)

Quantum superposition state: A single-bit quantum state is a vector $|\psi\rangle$ in a two-dimensional complex vector space, and may be represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Herein, α and β are complex numbers, and $|\alpha|^2 + |\beta|^2 = 1$ (the normalized single quantum bit state is a unit vector in the two-dimensional complex vector space).

The base vectors $|0\rangle$ and $|1\rangle$ are calculated and mapped to two orthogonal base vectors in a two-dimensional Hilbert space:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Any single-bit quantum state can be represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$



Quantum Gate

Classical logic gate

OR NOR AND NAND
XOR XNOR Buffer NOT

Quantum logic gate

X Y Z H

Pauli-X (X) Pauli-Y (Y) Pauli-Z (Z) Hadamard (H)

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Different from classical logic gates, quantum logic gates are based on unitary transformation of qubits, which **makes quantum gate operations reversible**.
- Quantum gates often use the matrix representation. A single quantum gate can be represented by a unitary matrix $2^k \times 2^k$. The number of input qubits of a gate must be equal to that of the output qubits.
- **Unitary transformation:** a rotation of a quantum state, so that the inner product of the quantum state remains unchanged before and after the transformation. In the real number domain, it can be simplified as a classical rotation.

$$U^\dagger U = I \Rightarrow U^\dagger = U^{-1}$$

Since we have qubits, can we operate them using logic gates, just like classical bits? Yes, there are gates X, Y, and Z. The effect can be represented by some matrices. For example, gate X can complete the bit-flipping task, from 0 to 1 or from 1 to 0. Another type of quantum gate is rotation operator gate, which rotates qubits around different axes in Hilbert space. These rotation angles can be set randomly, which is the basis for building a trainable neural network. Of course, we also have some relatively complex quantum logic gates which will not be described here.

Basic Quantum Gates

Pauli-X (X)	Pauli-Y (Y)	Pauli-Z (Z)	Hadamard (H)
$ 0\rangle \xrightarrow{X} 1\rangle$ $ 1\rangle \xrightarrow{X} 0\rangle$	$ 0\rangle \xrightarrow{Y} i 1\rangle$ $ 1\rangle \xrightarrow{Y} -i 0\rangle$	$ 0\rangle \xrightarrow{Z} 0\rangle$ $ 1\rangle \xrightarrow{Z} - 1\rangle$	$ 0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$ $ 1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(0\rangle - 1\rangle)$
$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Measurement gate



Measures the built quantum circuits, collects statistics on the collapsed bits at each gate, and finally obtains the statistical result.

CNOT gate

The controlled-NOT (CNOT) gate is a two-qubit gate X that acts based on the input status of the control bit:

- If the control bit is 0, no operation is performed.
- **If the control bit is 1, a gate X is applied to the target bit.**

If the low level is the control bit, a matrix is represented as:

$$CONT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

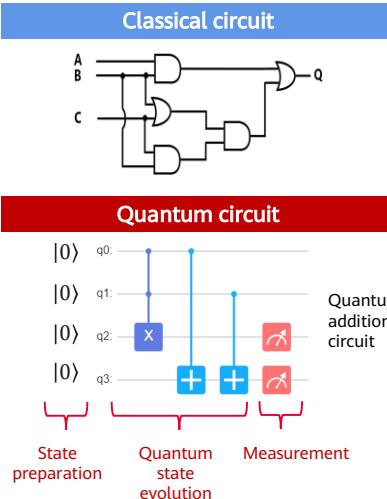


If the high level is the control bit, a matrix is represented as:

$$CONT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Quantum Circuit



- Quantum circuits are some quantum gate sequences that act on qubits so that the quantum states of qubits evolve into new quantum states.
- Quantum circuits can be divided into three parts: **state preparation**, **quantum state evolution**, and **quantum measurement**.
 - State preparation ensures that qubits evolve from state $|0\rangle$.
 - A quantum circuit with a series of quantum gates works on the qubits and the qubits evolve.
 - The qubits are then measured to obtain the output result of the quantum circuit.
- Similar to a quantum gate, a quantum circuit may be considered as a larger **unitary transformation** or may be represented as a large unitary matrix, because the function of a quantum gate is equivalent to matrix multiplication, and the result of multiplying a series of unitary matrices is still a unitary matrix.

With logic gates, the next step is how to organize these logic gates to complete specified tasks. The figure on the left shows an electronic circuit that we are very familiar with. It processes the electronic levels and completes certain logic operations. It is proven that only a few basic logic gates are required to complete all the logic operations. When it comes to qubits, we can also organize quantum logic gates and operate qubits to complete certain operations. For example, in the figure on the left, two qubits are in the initial state 0. If we apply a Hadamard gate to the first bit and then a NOT gate, an entangled quantum circuit as we mentioned earlier is ready.

Preparation of Bell-state Quantum Circuits

The maximum entangled state of two qubits is called the Bell state. There are four Bell states:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

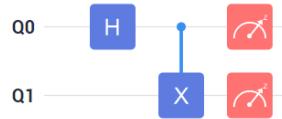
Generate Bell state

To generate Bell state $|\beta_{00}\rangle$, prepare a qubit in the $|00\rangle$ state, apply the Hadamard gate to change it to the maximum superposed state, and then use the CNOT gate.

$$\begin{aligned} U_{CNOT}(H \otimes I)|00\rangle \\ = U_{CNOT}(|0\rangle + |1\rangle)|0\rangle/\sqrt{2} \end{aligned}$$

The Bell states generated vary depending on the initial states:

Input	Output
$ 00\rangle$	$(00\rangle + 11\rangle)/\sqrt{2} \equiv \beta_{00}\rangle$
$ 01\rangle$	$(01\rangle + 10\rangle)/\sqrt{2} \equiv \beta_{01}\rangle$
$ 10\rangle$	$(00\rangle - 11\rangle)/\sqrt{2} \equiv \beta_{10}\rangle$
$ 11\rangle$	$(01\rangle - 10\rangle)/\sqrt{2} \equiv \beta_{11}\rangle$



- Quantum circuit for generating entangled states

Basic Quantum Algorithms - Deutsch Algorithm

Problem

For $x \in \{0, 1\}$, given an unknown Boolean function $f(x) \in \{0, 1\}$, there are two possibilities of $f(x)$:

Case 1 (constant): $f(0) = f(1)$

Case 2 (balance): $f(0) \neq f(1)$

To know which case $f(x)$ belongs to, the classical algorithm needs to evolve the function at least twice for comparison— $f(0)$ and $f(1)$.

Deutsch quantum algorithm

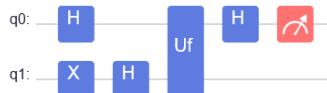
1. Generate the superposed state: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
2. Apply function $f(x)$:

$$\frac{1}{\sqrt{2}}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)$$

For case 1, the generated quantum state is $|+\rangle$; for case 2, the generated quantum state is $|-\rangle$.

3. To discriminate between state $|+\rangle$ and state $|-\rangle$, you can use the Hadamard gate: $H|+\rangle = |0\rangle$, $H|-\rangle = |1\rangle$. Then, measure the quantum state. If state $|0\rangle$ is obtained, it is case 1. If state $|1\rangle$ is obtained, it is case 2.

With the quantum algorithm, the function only needs to be evolved once.

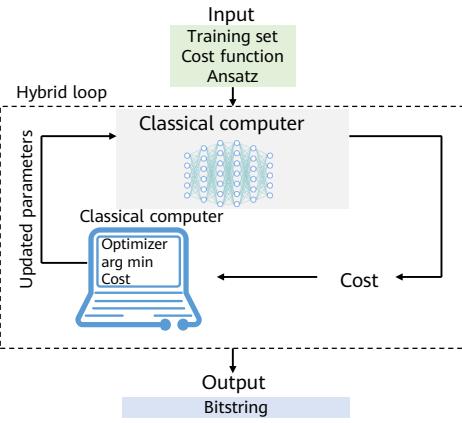


General-Purpose Quantum Algorithms

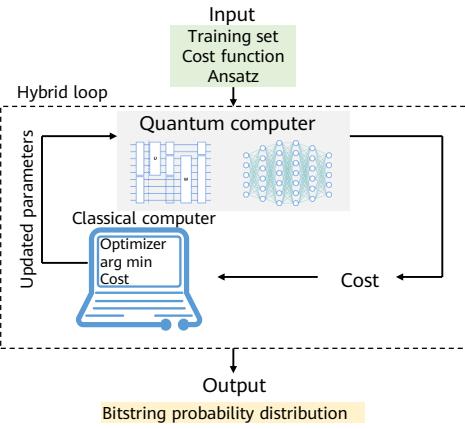
Shor	Grover	HHL
$p_1 \cdot p_2 = N$ $f(x) = a^x \bmod N$ <ul style="list-style-type: none"> • Problem: integer factorization • Complexity <p>Classical: $\mathcal{O}(e^{1.9(\log N)^{\frac{1}{3}}(\log \log N)^{\frac{2}{3}}})$</p> <p>Quantum: $\mathcal{O}((\log N)^3)$</p> • Application scenarios and advantages: <p>The Shor algorithm can get results in polynomial time, whereas the classical algorithm needs exponential time. The Shor algorithm can easily crack the existing RSA encryption system, whereas the quantum encryption algorithm has absolute security.</p> 	$f(x) = \begin{cases} 0, & x \neq x_{target} \\ 1, & x = x_{target} \end{cases}$ <ul style="list-style-type: none"> • Problem: unordered database search • Complexity <p>Classical: $\mathcal{O}(N)$</p> <p>Quantum: $\mathcal{O}(\sqrt{N})$</p> • Application scenarios and advantages: <p>The Grover algorithm is used to accelerate various algorithms and can get results in sub-exponential time, whereas the classical algorithm requires exponential time. It provides asymptotic acceleration for brute force cracking of the symmetric key algorithm (including collision attacks and original image attacks) and can crack 128-bit symmetric encryption keys in about 2^{64} iterations, or 256-bit symmetric encryption keys in about 2^{128} iterations.</p> 	$A x\rangle = b\rangle$ <ul style="list-style-type: none"> • Problem: linear equation solving • Complexity <p>Classical: $\mathcal{O}(Nk)$</p> <p>Quantum: $\mathcal{O}(\log(N) \kappa^2)$</p> • Application scenarios and advantages: <p>Compared with classical algorithms, the HHL algorithm achieves exponential acceleration in solving linear equations and has advantages in machine learning and numerical computing scenarios. Combined with the Grover algorithm, it will be a key technology for breakthroughs in fields such as quantum machine learning and artificial intelligence in the future.</p>

Variational Quantum Algorithm

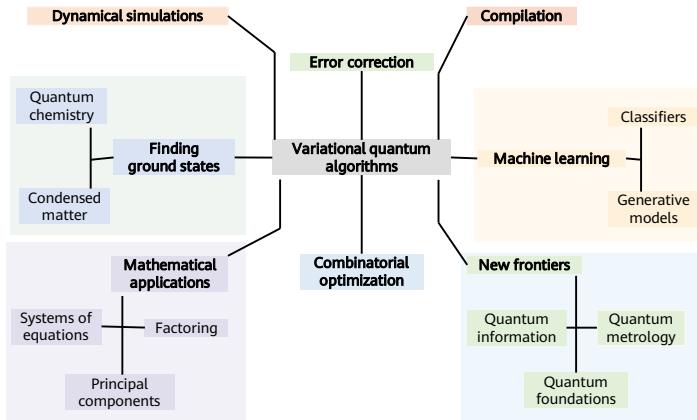
Classic Machine Learning Algorithm Flow



Variational Quantum Algorithm Flow



Application Scenarios of Variational Quantum Algorithms



VQE for Quantum Chemistry

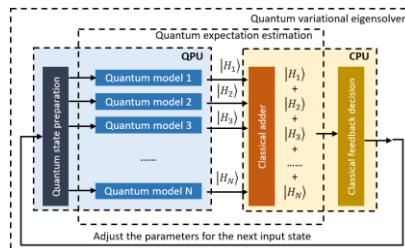
Challenge

Quantum chemistry aims to apply quantum mechanics to chemical systems, such as calculating the ground state energy of molecules by calculating the numerical solution of the Schrodinger equation. Quantum chemistry has become an important means to study the physical and chemical properties of materials. The exact solution of the Schrodinger equation has exponential complexity, and the scale of the chemical system that can be simulated is severely limited. However, only polynomial time is required for quantum computing.

Quantum Chemistry Computing Method

The core problem of quantum chemistry is to solve the Schrodinger equation. In quantum chemistry simulation, the variational method is usually used, that is, a trial wavefunction containing parameters is constructed, and the wavefunction with the lowest expected energy is found by continuously optimizing the parameters.

Quantum Chemistry VQE Algorithm Process



Variational Quantum Eigensolver (VQE): Because a wavefunction needs to be constructed, quantum computers based on quantum state evolution have natural advantages and can efficiently evolve to obtain the trial wavefunction. After the output (the expected energy value of the wavefunction) is obtained by using the quantum computer, the variational parameters in the quantum circuit may be updated by using the classical optimizer, and iterations are performed repeatedly until the ground state energy and the ground state wavefunction are found.



Quantum Approximate Optimization Algorithm(1)

Quantum Approximate Optimization Algorithm

Quantum Approximation Optimization Algorithm (QAOA) is a quantum algorithm used to solve combinatorial optimization problems. For a given NP-Hard problem, QAOA can find a good approximation solution in **polynomial time**. In addition, QAOA has a better approximation rate than any known classical polynomial-time algorithm and can be used in various scenarios such as **transportation, logistics, and finance**.

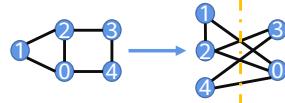
Max-Cut Problem Quantization

1. Each vertex in the graph is assigned a qubit. When the vertex is distributed to the left, the vertex is set to the $|0\rangle$ state. When the vertex is distributed to the right, the vertex is set to the $|1\rangle$ state.
2. A proper Hamiltonian quantity is selected, so that when connected vertices are in the same quantum state (on the same side), the expected value of the Hamiltonian quantity is 0, and when connected vertices are in different quantum states (on different sides), the expected value of the Hamiltonian quantity is -1.
3. An approximate optimal solution can be obtained by minimizing the expected value of the Hamiltonian quantity.
Overall, we turn the max-cut problem into **finding the ground state of the Hamiltonian quantity**. So, all we need to do is setting up a proper ansatz circuit based on the Hamiltonian quantity and continuously optimize the circuit parameters until the optimal solution is found. The figure on the right shows the detailed process.

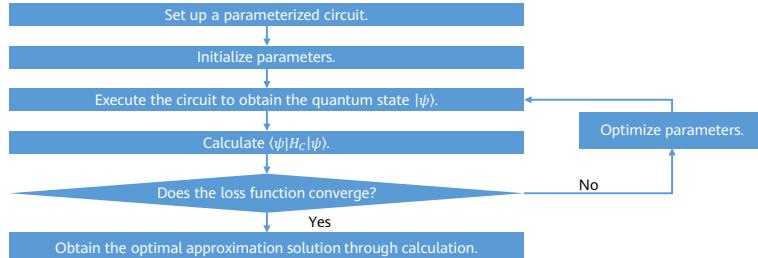
Quantum Approximate Optimization Algorithm(2)

Max-Cut Problem

The max-cut problem is an NP-complete problem in the graph theory. It needs to divide vertices of a graph into two parts and make the most edges be cut, as shown in the figure on the right.



Max-Cut Problem Solving Process



Contents

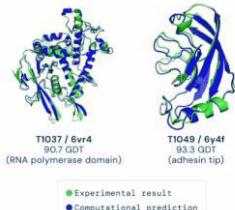
- 1. Development of Quantum Computing**
- 2. Basic Concepts of Quantum Computing**
- 3. Quantum Machine Learning**
 - Classical Machine Learning
 - Quantum Machine Learning
 - Advantages of Quantum Machine Learning
 - Iris Classification Algorithm
- 4. Quantum Computing Software**

Classical Machine Learning

Classical machine learning includes supervised learning, unsupervised learning, and reinforcement learning, and is widely used in various fields such as classification, image recognition, and behavior learning.



In 2017, AlphaGo trained in deep reinforcement learning defeated Ke Jie, who is the world's number one Go player.

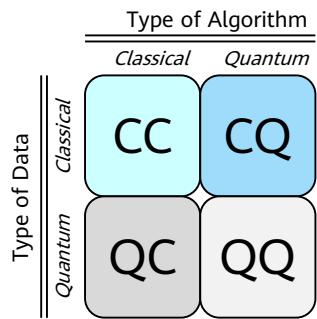


In 2020, the accuracy of protein structure prediction using AlphaFold 2 reaches 1.6 Å.

- Bottlenecks faced by classical machine learning: As data grows, the volumes overload classical computers.
- The unique **superposed state** and **state entanglement** properties of qubits can implement **exponential parallelism** to reduce computing complexity and improve data processing capability.
- For a 28 x 28 MNIST dataset, only **10 qubits** are required for encoding, outcompeting **classical 784 bits**.

Quantum Machine Learning

Machine learning can be classified into the following four types by **data and algorithm**:



- **CC:** classical data + classical algorithm (traditional machine learning)
- **CQ (AI4Quantum):** classical data + quantum algorithm, that is, classical machine learning is applied to the quantum field. Quantum data, such as quantum states and Hamiltonian quantities, can be represented by classical neural networks and tasks can be completed by training the parameters of the classical neural networks.

- **QC (Quantum4A):** quantum data + classical algorithm , which is classified into two types:
 - **A quantum version of traditional machine learning algorithms.** such as quantum principal component analysis (QPCA), quantum support vector machine (QSVM), and quantum reinforcement learning. These algorithms can obtain quantum acceleration from corresponding quantum algorithms, but are not suitable for near-term quantum devices.
 - **QNN:** Parameterized quantum circuits replace neural networks. For example, quantum convolutional neural networks (QCNNs) are used for classification, and quantum generative adversarial networks (QGANs) are used for generation.

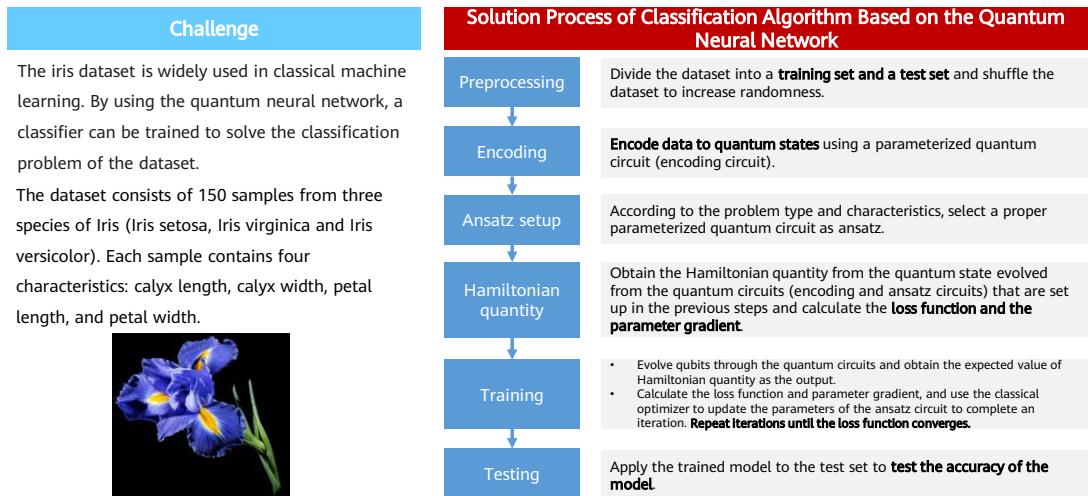
QQ: quantum data + quantum algorithm, that is, fully quantum machine learning.
Application scenario: Quantum data is unknown, and the data can be regenerated in other quantum systems through quantum machine learning.

Advantages of Quantum Machine Learning

- By using properties such as quantum state superposition and entanglement in quantum computing, a plurality of qubits can represent more complex states or implement more complex operations, thereby implementing quantum acceleration.
- Quantum systems are more suitable for linear algebra operations (distance estimation of high-dimensional vectors).
- Quantum algorithms with polynomial or exponential acceleration: HHL, quantum PCA (qPCA), quantum-enhanced SVM (QSVM), Grover search, etc.
- The NISQ phase requires a QC hybrid architecture that uses classical machine learning to control parameters in quantum circuits.

Algorithm	Complexity
Bayesian inference	$O(\sqrt{N})$
Online perceptron	$O(\sqrt{N})$
Least-squares fitting	$O(\log N)$
Classical Boltzmann machine	$O(\sqrt{N})$
Quantum Boltzmann machine	$O(\log N)$
Quantum PCA	$O(\log N)$
Quantum SVM	$O(\log N)$
Quantum reinforcement learning	$O(\sqrt{N})$

Iris Classification Algorithm



25 Huawei Confidential

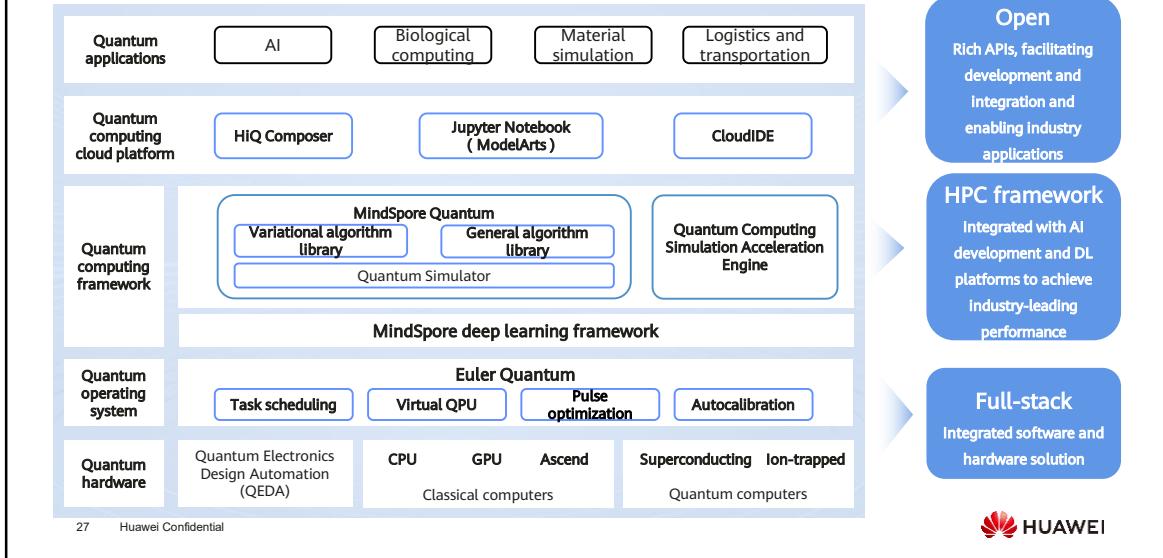


For details about the solution process, see the HiQ quantum computing tutorial on the official website.

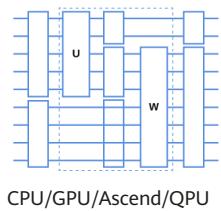
Contents

- 1. Development of Quantum Computing**
- 2. Basic Concepts of Quantum Computing**
- 3. Quantum Machine Learning**
- 4. Quantum Computing Software**
 - MindSpore Quantum Computing Framework
 - HiQ Quantum Computing Cloud Platform
 - Quantum Software Programming Practice

HiQ Quantum Computing Full-Stack Solution



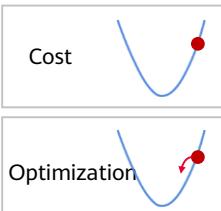
MindSpore Quantum



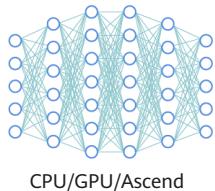
Predict

Update

CPU/GPU/Ascend/QPU



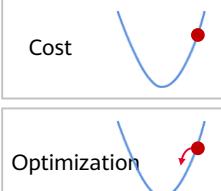
Search



Predict

Update

CPU/GPU/Ascend



Search

MindSpore
Quantum

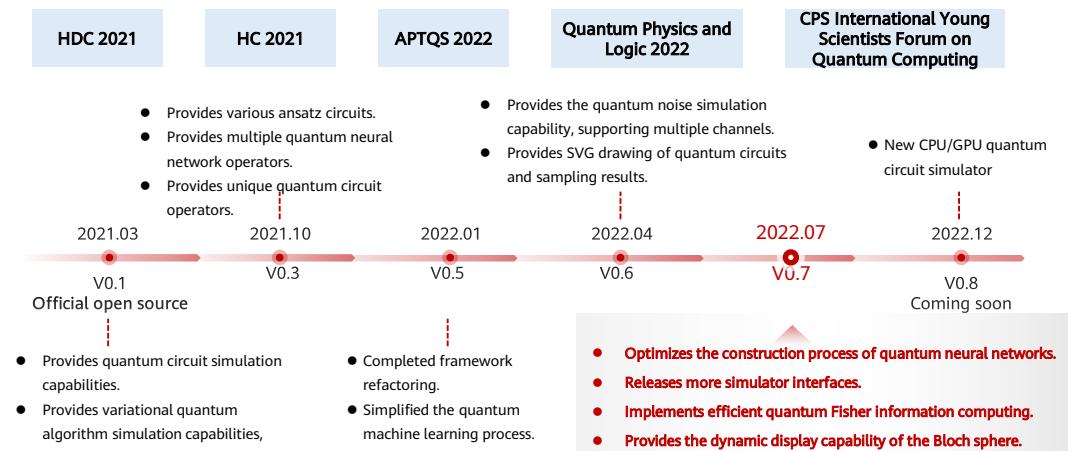
Quantum
computing framework

[M]^s 昇思
MindSpore

Deep learning framework



Evolution of MindSpore Quantum: Pursuing Excellence and Continuous Innovation



29 Huawei Confidential



- 2022.04 Fourth Asia-Pacific Workshop on Trapped Quantum Systems
- 2022.07 2022 Open Atom Global Open Source Summit

MindSpore Quantum: Making Quantum Computing Reachable



Superb experience

- Rich **variational circuit algorithm libraries**, enabling frontiers
- Efficient and convenient **template-based** quantum neural networks
- In-depth integration with **MindSpore**
- **Various hardware platforms**
- **Visualized circuit rendering**
- Non-Hermitian operations such as **gradient calculation of expected values**

Ultimate performance

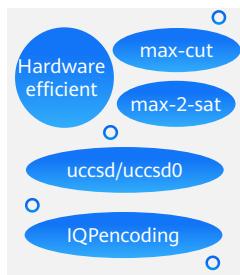
- Industry-leading **variational quantum computing performance**
- Industry-leading **28-qubit** quantum chemical simulation and **32-qubit** acceleration (QuPack)
- **High-performance simulator**, supporting analog computing of 30+ qubits.
- Up to 10x and 2x increase in GPU and CPU performance (compared with version 0.7)

Simplified development

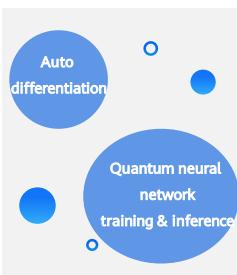
- **Pre-integrated with HiQ cloud services**, installation-free, and out-of-the-box
- **One-click PIP installation**, supporting various operating systems
- Built-in quantum machine learning, chemical simulation, and combinatorial optimization modules, providing **rich tutorials**
- Rich **programming Interfaces** and development instances, simple and easy to use

Superb Experience of MindSpore Quantum

Rich variational circuit algorithm libraries



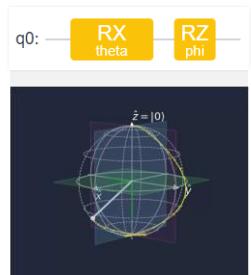
In-depth integration with MindSpore



Various hardware platforms

CPU/GPU/**Ascend/QPU**

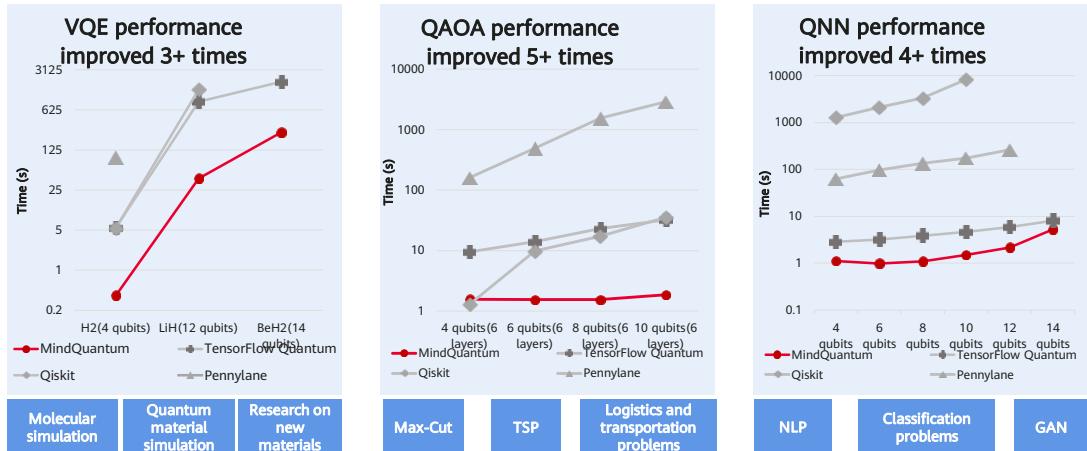
Multi-format visualized circuit rendering



Based on the high-performance in-depth AI development platform, MindSpore Quantum provides a rich algorithm library, enabling developers to easily develop quantum software and algorithms based on the framework.

- hardware efficient
- max-cut
- max-2-sat
- uccsd / uccsd0
- IQPencoding

Ultimate Performance of MindSpore Quantum: Industry-leading Variational Quantum Algorithms



32 Huawei Confidential

Beta version

MindQuantum 0.5.0 (2022.02.28)
qiskit-terra 0.19.2 (2022.02.09)

TensorFlow Quantum 0.6.1 (2022.02.04)
PennyLane 0.21.0 (2022.02.08)



In addition to rich application scenarios, MindSpore Quantum provides industry-leading performance. For example, in terms of VQE and QAOA problems, we have 300% and 500% acceleration compared with TensorFlow Quantum. In terms of QNN problems, our acceleration is at least 1x that of competitors.

<https://gitee.com/mindspore/mindquantum/releases>

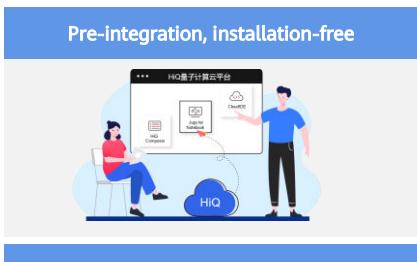
https://qiskit.org/documentation/release_notes.html#terra-0-19-0

<https://github.com/tensorflow/quantum/releases>

<https://github.com/PennyLaneAI/pennylane/releases>

Simplified Development on MindSpore Quantum

Pre-integration, installation-free

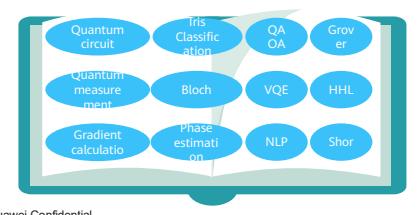


One-click installation, efficient and fast

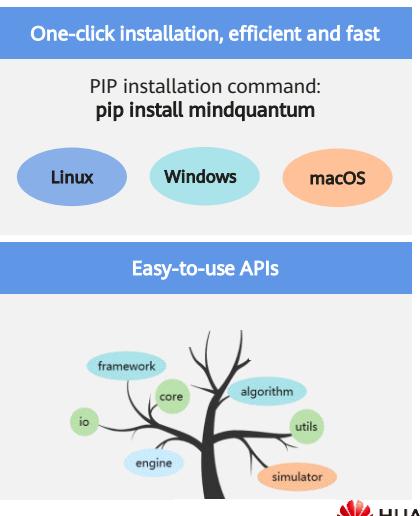
PIP installation command:
`pip install mindquantum`

Linux Windows macOS

Rich tutorials, easy to get started



Easy-to-use APIs



HUAWEI

33 Huawei Confidential

Rich APIs, providing development instances, and easy-to-use

MindSpore Quantum Provides Rich Tutorials for Learning and Scientific Research

Step 1

Quantum computing beginner and advanced tutorials & training videos

Step 2

Tutorial for solving typical application problems of quantum computing

Step 3

Academic papers and solution verification, providing cases for quantum software research and innovation

2022 QWorld Quantum Computing and Programming Course

<https://hiq.huaweicloud.com/tutorial>



MindSpore Quantum Tutorial and Developer Guide

<https://hiq.huaweicloud.com/tutorial>

Download Notebook

Download Sample

View source on Gitee

MindSpore Quantum papers, open source code

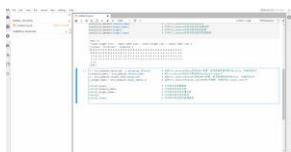
<https://hiq.huaweicloud.com/consult/paper>

HiQ Quantum Computing Cloud Platform Provides Various Frontend and Backend Services



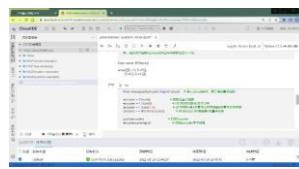
HiQ Composer
GUI tool
(For beginners)

Drag-and-drop quantum circuit setup, easy to use.



Jupyter Notebook
Interactive IDE
(Teaching)

Interactive programming IDE to facilitate quantum algorithm development.



CloudIDE
Cloud Native IDE
(Lightweight development)

A professional programming interface similar to VS Code and highly integrated with tools such as Git



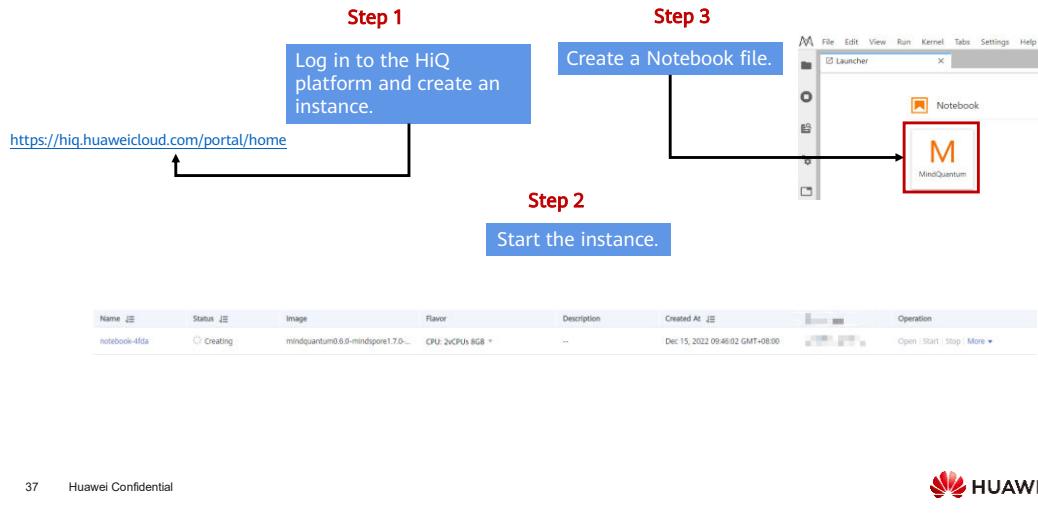
ModelArts
AI development platform
(High-performance computing)

Paid services with more powerful computing resources to meet the requirements of large-scale algorithm research

Contents

- 1. Development of Quantum Computing**
- 2. Basic Concepts of Quantum Computing**
- 3. Quantum Machine Learning**
- 4. Quantum Computing Software**
 - MindSpore Quantum Computing Framework
 - HiQ Quantum Computing Cloud Platform
 - MindSpore Quantum Programming Practice

Creating a Quantum Programming Development Environment



The HiQ quantum computing cloud platform is installation-free and free of charge. You do not need to install dependency packages, or MindSpore Quantum. You can log in to the HiQ platform, select Jupyter Notebook or CloudIDE, and create an instance for quantum programming.

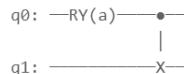
Practice 1: Setting Up a Quantum Circuit

1. Import the dependency package.

```
import numpy as np
# import quantum gate
from mindquantum.core.gates import X, H, RY
# import quantum circuit
from mindquantum.core.circuit import Circuit
```

3. Print the quantum circuit.

```
print(circ)
```



2. Create a quantum circuit and add a quantum gate.

```
circ = Circuit()      # Create a quantum circuit.
circ += RY('a').on(0) # Add a RY gate to bit 0 of the circuit.
circ += X.on(1, 0)    # Add an H gate to bit 1 of the circuit.
```

Let's try the simplest quantum programming, that is, setting up a quantum circuit.

Step 1: Import the dependency package. If you do not know which packages need to be imported, import all packages.

Step 2: Import and create a quantum circuit and add quantum gates.

Step 3: Print the quantum circuit directly or print after rendering.

```
from mindquantum import * # Import the dependency package.
```

```
circ = Circuit()      # Create a quantum circuit.
```

```
circ += RY('a').on(0) # Add a RY gate to bit 0 of the circuit.
```

```
circ += X.on(1, 0)    # Add an H gate to bit 1 of the circuit.
```

```
print(circ)          # Print the measurement circuit.
```

```
# circ.svg()
```

```
# circ.svg('light')
```

```
# circ.svg('dark')
```

Practice 2: Variational Quantum Computing

4. Create the Hamiltonian quantity.

```
ops = QubitOperator('X0 X1') # Create the Hamiltonian quantity wrapper.  
ham = Hamiltonian(ops) #Create the Hamiltonian quantity.  
print(ham) # Print the Hamiltonian quantity.
```

1 [X0 X1]

5. Calculate the expected value and gradient.

```
sim = Simulator('projectq', 2) # Create a 2-qubit simulator.  
# Obtain a function that returns the forward value and the gradient of the circuit parameter.  
grad_ops = sim.get_expectation_with_grad(ham, circ)  
grad_ops(np.array([1.0])) # Calculate the gradient and expected value.
```

(array([[0.84147098+0.j]]), Expected value
array([[0.54030231+0.j]])) Gradient

Next, let's try variational quantum computing programming.

The first three steps are the same as the previous task.

Step 1: Import the dependency package.

Step 2: Import and create a quantum circuit and add quantum gates.

Step 3: Print the quantum circuit directly or print after rendering.

Step 4: Create the Hamiltonian quantity.

Step 5: Calculate the expected value and gradient.

```
from mindquantum import *
```

```
import numpy as np
```

```
from IPython.display import display_svg
```

```
circ = Circuit() # Create a quantum circuit.
```

```
circ += RY('a').on(0) # Add a RY gate to bit 0 of the circuit.
```

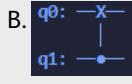
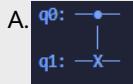
```
circ += X.on(1, 0) # Add an X gate to bit 1 of the circuit.
```

```
print(circ) # Print the quantum circuit.
```

```
# circ.svg() # Print the quantum circuit at the end.  
# circ.svg('dark')  
# display_svg(circ.svg('light')) # Print the quantum circuit in the middle.  
  
ops = QubitOperator('X0 X1') # Create the Hamiltonian quantity wrapper.  
ham = Hamiltonian(ops) #Create the Hamiltonian quantity.  
  
print(ham) # Print the Hamiltonian quantity.  
  
sim = Simulator('projectq', 2) # Create a 2-qubit simulator.  
grad_ops = sim.get_expectation_with_grad(ham, circ) # Obtain a function that returns the  
forward value and the gradient of the circuit parameter.  
  
grad_ops(np.array([1.0])) # Calculate the gradient and expected value.
```

Quiz

1. (True or false) The Shor algorithm can efficiently perform integer factorization, which is much faster than the classical algorithm.
 - A. True
 - B. False
2. (Single-choice) Which of the following quantum circuits corresponds to MindSpore Quantum X.on (0,1)?



Recommendations

- Huawei official websites:
 - HiQ quantum computing: <https://hiq.huaweicloud.com/home>
 - MindSpore: <https://mindspore.cn/>
- Quantum software
 - MindQuantum: <https://gitee.com/mindspore/mindquantum>
 - HiQ quantum computing cloud platform:
<https://hiq.huaweicloud.com/portal/home>



HiQ official website



MindSpore official website



MindQuantum community



HiQ 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©2023 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.

