Artificial Intelligence Foundation of Smart Ocean

Xiaofeng Li, Fan Wang, Yuan Zhou, and Keran Chen

1 The Development of Artificial Intelligence

Artificial intelligence (AI) is the core driver for the fourth technological revolution, following the revolutions in steam technology, electricity technology, and computers and information technology. Since its emergence in the 1950s, AI has fully improved productivity, affected and changed the production structure and production relations. Understanding the history of AI plays an indispensable role in the subsequent research and the development of AI technologies. AI can be divided into three generations, according to the difference in the drive mode. The subsequent subsections introduce each of these three generations of AI.

1.1 The First-Generation AI

Turing proposed the "Turing Test" in 1950 [49]. It states that if a machine can answer a series of questions posed by a human tester within five minutes, and more than 30% of its answers can deceive the tester into thinking that they are answered by a human, then the machine can be considered intelligent. In the same year, Turing predicted the feasibility of intelligent machines. The "Turing test" can be seen as the genesis of AI. Newell and Simon [41] developed the first heuristic program in the world: Logic Theorist. It successfully proved 38 theorems in the book: "Principles of Mathematics", by simulating human thinking activities. This program successfully

X. Li (✉) · F. Wang
CAS Key Laboratory of Ocean Circulation and Waves, Institute of Oceanology, Chinese Academy of Sciences, Qingdao 266071, China
e-mail: xiaofeng.li@ieee.org

Y. Zhou · K. Chen
School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China

X. Li and F. Wang (eds.), *Artificial Intelligence Oceanography*,
https://doi.org/10.1007/978-981-19-6375-9_1

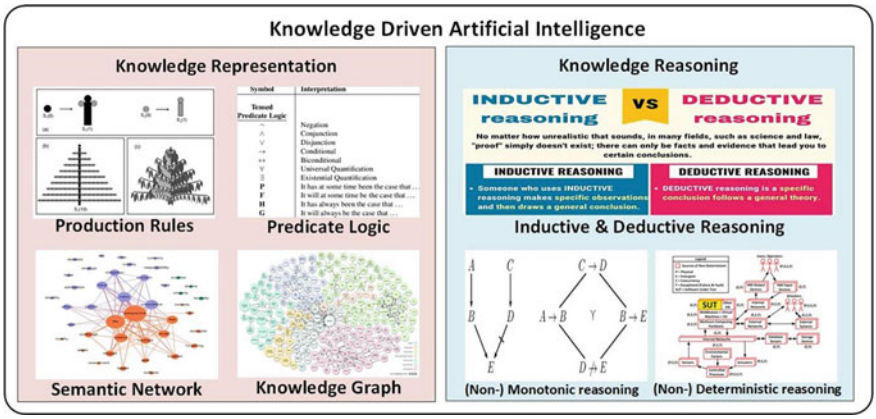

Fig. 1 The research field of the first generation of artificial intelligence. The first generation of AI is knowledge-driven AI, which mainly conducts research on knowledge representation and reasoning. Production rules, predicate logic, semantic network and knowledge graph are common knowledge representations. Inductive reasoning, deductive reasoning, Monotonic reasoning and deterministic reasoning are mainstream reasoning methods

demonstrated the feasibility of the predictions posed by Turing, and it is considered the first successful AI program. In August of the same year, the concept of "artificial intelligence" was first introduced by John McCarthy, Herbert Simon, and a group of scientists from different fields at Dartmouth College. Thus, AI stands on the stage of history as an independent discipline. Newell and Shaw [40] invented the first AI programming language, the information processing language (IPL). It used symbols as basic elements and proposed a reference table structure instead of storing addresses or arrays. McCarthy [36] developed a list processing language based on the IPL, which was widely used in the AI community.

The first generation of AI is known as knowledge-driven AI; these AIs allow machines to learn by imitating the process of human reasoning and thinking. As shown in Fig. 1, the core steps can be divided into two parts, knowledge representation and knowledge reasoning.

Knowledge representation is required to allow machines to achieve intelligent behavior. It represents human-understood knowledge in a certain data structure that allows machines to understand and complete the processing. The methods of knowledge representation include predicate logic, production rules, semantic network representation, and knowledge graphs.

Predicate logic can describe how the human mind works. From the logical system, propositional logic is the simplest logical system, and it is used to describe declarative sentences using truth values. For example, "the sea is blue"; the "true" and "false" of each proposition is called the truth value. Propositions can be divided into atomic propositions and positions. An atomic proposition is a proposition that cannot be split into simpler declarative sentences. Compound propositions are detachable propositions consisting of atomic propositions and connectives. However, they both have

limited expressive power and can only represent established facts. Thus, predicate logic is developed based on propositional logic. It uses connectives and quantifiers to describe objects, and predicates on objects to represent the world. Predicates of objects refer to the properties of objects or the relationships between objects. A constant symbol, a predicate symbol and a function word comprise a predicate. Constant symbols represent objects and predicate symbols represent relationships or attributes. For example, the constant symbol Susan, the predicate symbol mother, and the function word nurse compose the predicate logic "nurse(mother(Susan))", which indicates that the mother of Susan is a nurse. Predicate logic representation has certain advantages: naturalness, accuracy, rigor, and ease of implementation. However, it cannot represent uncertain knowledge, and when it is used to describe too many things, it becomes inefficient.

Production rules are used to describe the cause-effect relationships between things. For example, if an animal is a mammal and has a long trunk, then the animal is an elephant. The generative system consists of a rule base, comprehensive database, and control system. The rule base is used to represent a set of rules for inferring conclusions from premises. The database is used to store known conditions, intermediate results, and final conclusions. The control system is used to select suitable rules for inference from a rule base.

Semantic network representation is a network graph that represents knowledge through entities and their semantic relationships. It consists of nodes and arcs. Nodes represent entities, which are used to describe various things, concepts, situations, attributes, states, events, actions, etc., and arcs represent semantic relations, such as the instance relations, classification relations, membership relations, attribute relations, inclusion relations, temporal relations, location relations, etc. These basic units are interconnected to form a semantic network.

A knowledge graph is essentially a semantic network that reveals the relationships between entities, allowing a formal description of entities and their interrelationships in the objective world. The study of knowledge graphs originated from a semantic web. Tim Berners Lee [2] proposed the concept of a semantic web at the XML Conference in 2000, expecting to provide services such as information proxy, search proxy, and information filtering by adding semantics to web pages. In 2005, Metaweb was established in the United States to develop an open knowledge base for web semantic services. It extracts entities (people or things) in the real world and relationships between them based on public datasets such as Wikipedia and the United States Securities and Exchange Commission (SEC), and then stores them with a graph structure on a computer. In 2010, Google acquired MetaWeb and acquired its semantic search technology. In 2012, Google formally put forward the concept of a knowledge graph, aiming to improve the capability of the search engine and enhance the search experience of users based on knowledge graphs.

Reasoning is a form of thinking that logically derives new conclusions from known premises. Knowledge reasoning represents the process of using knowledge, which is one of the core issues in AI research. It uses previous knowledge to derive conclusions by reasoning, and solves the corresponding problems. Reasoning can be divided into

deductive reasoning and inductive reasoning, depending on how the conclusion is derived. Where deductive reasoning is a reasoning process from general to special, inductive reasoning is a reasoning process from special to general. Reasoning can also be divided into monotonic and nonmonotonic reasoning, depending on whether the conclusions derived in the reasoning process increase monotonically. Monotonic means that the number of propositions known to be true strictly increases as the reasoning progresses. According to the certainty of reasoning, reasoning can also be divided into deterministic and non-deterministic reasoning, where deterministic means that the knowledge used in reasoning and the conclusions derived are either true or false, whereas non-deterministic means that the knowledge used in reasoning and the conclusions derived are probabilistic.

In the era of knowledge-driven AI, the emergence of expert systems has brought AI into a period of vigorous development. An expert system is an intelligent computer program that introduces the knowledge of a specialized field. Through knowledge representation and reasoning, it can simulate the decision-making process of human experts to solve the problems in this field and provide suggestions for the users. The first expert system was DENDRAL [4], developed by Feigenbaum in 1968. It was used to analyze the molecular structure of organic compounds by mass spectrometry. In the 1970s, the idea of expert systems was gradually accepted. A series of expert systems were developed to solve problems in different fields at that time, such as MYCIN [46] for diagnosis and treatment of blood infection diseases, MACSYMA [35] for symbolic integration and theorem proving, and PROSPECTOR [14] for seismic exploration. Subsequently, the application field of expert systems expanded rapidly, and the difficulty of dealing with problems increased continuously. Several tool systems for building and maintaining expert systems have been developed. In the 1980s, the development of expert systems gradually became specialized, creating huge economic benefits.

Although scientists at that time had great expectations for knowledge-driven AI, there were some fundamental problems in its development. The first problem was the interaction problem. Traditional methods could only simulate the thinking process of human beings, but could not simulate the complex interactions between humans and the environment. The second problem was the expansion problem. Traditional methods were only applicable to the development of expert systems in specific fields, and could not be extended to complex systems with larger scales and wider fields. The third problem was the application problem. Research on traditional methods was detached from the mainstream computing (software and hardware) environment, which seriously hindered the practical application of expert systems. Constrained by the above mentioned problems, the first generation of AI eventually declined.

1.2 The Second-Generation AI

First-generation AI is based on symbols, which believe that sensory information is expressed in a certain encoding way. Second-generation AI establishes a stimulus-response connection in a neural network. This ensures the generation of intelligent behavior through connections. Figure 2 illustrates the development of second-generation AI

In 1958, Rosenblatt established the prototype of an artificial neural network, the perceptron, which followed the idea of connectionism. The perceptron was inspired by two aspects. One was the neuron mathematical model proposed by McCulloch and Pitts in 1943 [37]: the threshold logic circuit, which converted the input of neurons into discrete values. The second was from the Hebb learning rate proposed by D.O.Hebb in 1949 [23], that is, the neurons fired at the same time are connected.

In 1969, Minsky and Papert [38], pointed out that perceptron could only solve linearly separable problems. In addition, it was not practical when the number of hidden layers increased as it lacked an effective learning algorithm. The criticism of the perceptron posed by Minsky proved to be fatal; thus, second-generation AI declined for more than 10 years.

Regarding difficulty, through the joint efforts of many scholars, significant progress has been made in both neural network models and learning algorithms. In addition, mature theories and technologies have gradually formed over the past 30 years.

For example, the gradient descent method was proposed by the French mathematician Cauchy [5]. The method is used to solve the minimum value along the direction of gradient descent or to solve the maximum value along the direction of the gradient rise.

Another example is the back-propagation (BP) algorithm [43]. The algorithm consists of a forward propagation process and a BP process. In the forward propagation process, the input information enters the hidden layer through the input layer. Then, the information is processed layer by layer and passed to the output layer.

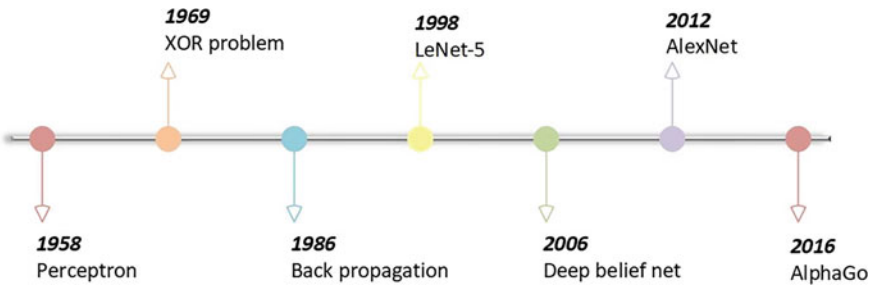

Fig. 2 The development history of the second generation of artificial intelligence. The second-generation AI has developed since the advent of the perceptron in 1958. The birth of AlphaGo in 2016 entered a period of rapid development

If the desired output value cannot be obtained in the output layer, the sum of the squares of the error between the output value and the expected value is taken as the objective function. In the BP process, the partial derivative of the objective function to the weight of each neuron is obtained layer by layer as the basis for modifying the weight. The learning of the network is completed using the weight modification process. When the error reaches the expected value, the network learning ends.

Regarding the loss function, a series of improvements have been made, such as the cross-entropy loss function [28]. Cross entropy is an important concept in Shannon's information theory, and is mainly used to measure the difference in information between two probability distributions. The performance of a language model is typically measured using cross-entropy and complexity. Cross-entropy means the difficulty of text recognition within a model or the average number of bits used to encode each word from a compression viewpoint. Complexity means to use the model to represent the average number of branches in this text. Cross entropy is introduced to the neural network field as a loss function. We used p to represent the distribution of the true markers and q to represent the predicted marker distribution of the trained model. The cross-entropy loss function measures the similarity between p and q.

Algorithm improvements, such as regularization methods, prevent over-fitting [52]. Regularization involves imposing constraints that minimize the empirical error function. Such constraints introduce prior distributions to the parameters and have a guiding effect. When optimizing the error function, they tend to choose the direction that reduces the gradient and satisfies the constraints; hence, the final solution tends to conform to prior knowledge. At the same time, regularization solves the ill-posedness of the inverse problem. The resulting solution exists uniquely and depends on the data. The influence of noise on the ill-posed is weak. If the regularization is appropriate, the solution will not overfit, even if the number of uncorrelated samples in the training set is small.

New network architectures have been developed, such as convolutional neural networks (CNNs) [13], recurrent neural networks (RNNs) [33], long short-term memory neural networks (LSTM) [25], and deep belief networks (DBN) [24].

CNNs are a type of feedforward neural network (FNN) that includes convolution calculations and has a deep structure. A CNN has the abilities of representation learning and can perform the shift-invariant classification of the input information according to its hierarchical structure. A CNN is constructed by imitating the biological visual perception mechanism, which can perform both supervised and unsupervised learning. The convolution kernel parameter sharing in the hidden layer and the sparsity of inter-layer connections enable the CNN to perform smaller calculations to extract features.

An RNN is a type of recursive neural network in which all node cyclic units are connected in a chain. RNNs have applications in natural language processing (NLP) fields, such as speech recognition, language modeling, machine translation, and other fields. An RNN can be combined with a convolution operation to handle computer vision problems.

The LSTM network is a time RNN, which is specifically designed to solve the long-term dependence problem in general RNNs. When receiving new input infor-

mation, the network first forgets all the long-term information that it does not require. Afterward, it learns which part of the new input information has use value and saves them in long-term memory. Finally, the network learns which part of the long-term memory can work immediately.

A DBN is a deep neural network with multiple hidden layers. It is a combination of unsupervised feature learning and supervised parameter adjustment. It performs unsupervised greedy learning using stacked restricted Boltzmann machines to extract high-level and abstract features from the original data. It uses a BP neural network to reversely fine-tune the parameters to realize the supervised learning of data.

Together, these works ushered in a new era of second-generation AI based on deep learning. Owing to the universality of deep neural networks, these networks can approximate any function. Therefore, using deep learning to determine the function of the data has a theoretical guarantee.

In 2014, deep learning was pointed out to be vulnerable to spoofing and attacks. Owing to the uncertainty of observation and measurement data, the acquired data must be incomplete and contain noise. In this case, the choice of the neural network structure is extremely important. If the network is too simple, there is a risk of underfitting; if it is complicated, overfitting occurs. Although the risk of overfitting can be reduced to a certain extent through various regularization methods, it will inevitably lead to a serious decline in the promotion ability if the quality of the data is poor.

1.3 The Third-Generation AI

The third generation of AI needs to solve the shortcomings of the first and second generations of AI. To establish sound AI theory, developed AI technology must be safe, credible, reliable, and scalable. Only when the above conditions are met can a real technological breakthrough be achieved, which will produce new innovative applications of AI. The best current approach is an organic combination of the first-generation knowledge-driven approaches and the second-generation data-driven approaches. The combined use of knowledge, data, algorithms, and arithmetic power results in a more powerful AI.

Research on third-generation AI must move towards making AI capable of powerful knowledge and reasoning. For this purpose, we need to draw on classic examples, such as the Watson conversational system, which was introduced in 2011. The following lessons on knowledge representation and inference methods from this system are worth learning. First, automatic generation of structured knowledge representations from a large amount of unstructured text. Second, a method for representing knowledge uncertainty based on the knowledge quality scoring. Third, an approach based on multiple reasonings to achieve the uncertainty reasoning. The development of third-generation AI requires strong knowledge and reasoning capabilities, in addition to strong perception. There have been some tentative efforts to apply the principle of sparse discharge to the computation of ANN layers [26]. Specifi-

cally, the network is trained with simple background images, such as "human," "car," "elephant," and "bird", as training samples. The neurons representing these "categories" appear in the output layer of the neural network. The network responds to the contours of the human face, car, elephant, and bird. In this way, the semantic information of the "whole object" is extracted, thus making the neural networks perceptive. However, this approach can only extract part of the semantic information, it cannot extract different levels of semantic information; therefore, further research is needed. Furthermore, the third generation of AI also needs to interact with the environment. Reinforcement learning has made good progress in many areas, such as video games [39, 50], board games [47, 48], robot navigation and control [11, 44], and human-computer interaction. In some tasks, the performance of reinforcement learning approaches used in the networks even surpasses that of humans.

Attempts on the third generation of AI have emerged in the academic community. Zhu et al. [56] proposed the use of a triple-space fusion model, that is, a model that fuses both dual-space and single-space approaches. The model has the opportunity to be interpretable and robust. When the model can convert sensory signals such as vision and hearing into symbols, the machine has the opportunity to develop comprehension capabilities, which will help solve the problem of interpretability and robustness of the model. If the symbols in the machine can be generated by the perception of the machine, then the symbols and symbolic reasoning can generate intrinsic semantics, which can hopefully solve the problem of interpretability and robustness of machine behavior at the root. Among the models proposed by Zhu et al. [56], the single-space model is based on deep learning and suffers from being uninterpretable and having poor robustness. The dual-space model mimics the working mechanism of the brain; however, the model has some uncertainties. For example, the machine can establish "intrinsic semantics" through the reinforcement learning of the environment, yet it is not certain whether these semantics are consistent with the "intrinsic semantics" acquired by humans through perception. Therefore, there are many uncertainties associated with this approach. Despite these difficulties, we still believe that machines that take steps in this direction will edge closer to true AI. The single-space model is based on a deep learning algorithm that fully uses the computational power of the computer. In some aspects, it already outperforms humans. However, there are many uncertainties in this approach, and it is still unknown how much progress can be made through algorithmic improvements. Therefore, to achieve the goal of third-generation AI, the best strategy is to simultaneously advance along two lines, namely, the convergence of the three spaces. This will maximize the working mechanism of the brain and make full use of the computing power of the computer. The combination of the two approaches is expected to lead to a more powerful AI.

The third generation of AI is a new form of AI driven by data and knowledge in concert. It fits perfectly with our need to explore the ocean using the ocean and its data. This new form of AI technology will be a powerful tool for people to understand the ocean and further develop it. The combination of AI technology and research in the ocean field will open a new chapter in oceanographic research.

At present, third-generation AI is still in its initial stage, and the AI used in academia and industry is mainly second-generation. Among the second-generation AI technologies, deep neural network-based AI technologies are the most representative. Therefore, the rest of this paper introduces common neural network structures and their applications.

2 The Architecture of Deep Neural Networks

2.1 Deep Feedforward Neural Network

A deep FNN is a typical deep learning model that can be viewed as a mathematical function. It realizes the complex mapping from input to output using a combination of nonlinear functions. The following section introduces the neuron algorithm, BP algorithm, single-layer FNN, and multi-layer FNN.

2.1.1 Neuron

A biological neuron usually contains multiple dendrites and axons. Dendrites are used to receive signals transmitted from other neurons, and the axon has multiple endings for transmitting signals to other connected neurons. Psychologist McCulloch and mathematician Pitts proposed an abstract neuronal model based on biological neurons in 1943 [37].

A complete neuron can be viewed as a computational process of "input, numerical computation, and output." The "numerical computation" consists of "a linear part and a non-linear part." Figure 3 shows a typical biological neuron and neuron model.

This neuron contains three inputs, a_1, a_2, a_3, and one output. By assigning the corresponding weights w_1, w_2, and w_3 to each input, the output of the linear part can be expressed as

$$z = a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b \tag{1}$$

where b is the bias term.

The nonlinear part is implemented by the activation function, and the output of the neuron can be expressed as

$$y = f(z) \tag{2}$$

where $f(\cdot)$ is the activation function.

The activation function is a nonlinear mapping relationship and must adhere to the following three conditions:

(1) It must be a continuous and differentiable (can be non-differentiable at finite points) nonlinear function.

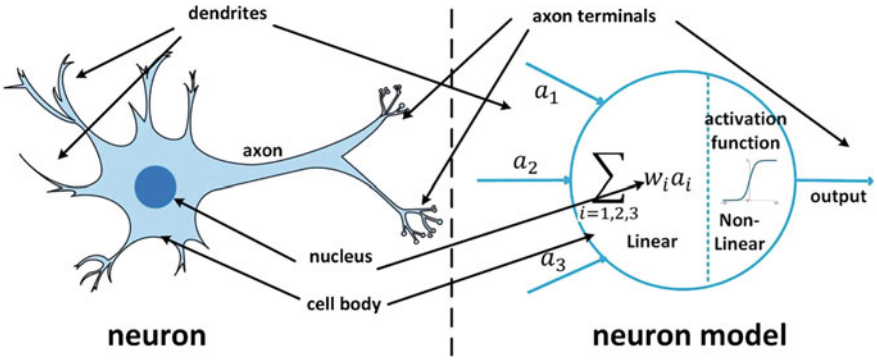

Fig. 3 Comparison of a biological neuron and neuron model. The left side shows a biological neuron. It consists of a nucleus, a cell body, an axon, dendrites and axon terminals. The right side shows a neuron model. It can be viewed as a computational process of "input, numerical computation, and output"

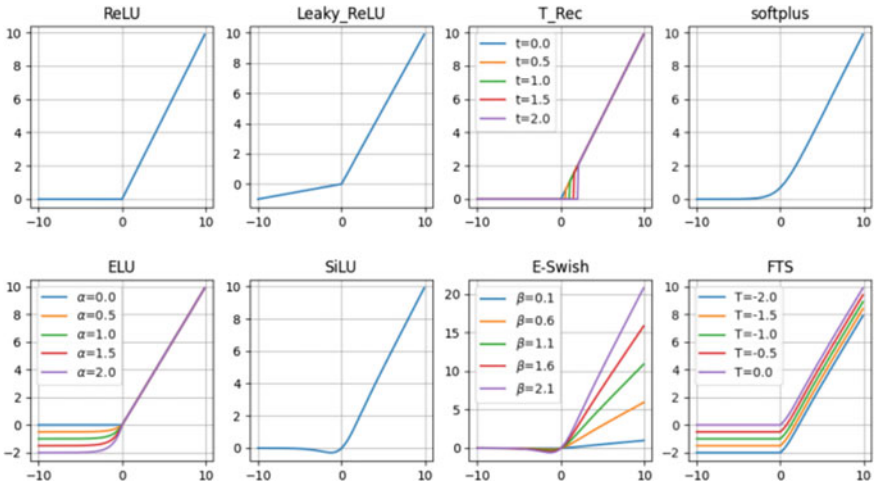

Fig. 4 Function diagrams of some common activation functions

(2) The activation function and its derivative should be simple; overly complex functions are not conducive for network efficiency.
(3) The value domain of the derivative should be limited to a suitable interval; this is beneficial for improving network efficiency.

Common activation functions are shown in Fig. 4.

In a neural network, the process of the input going through the neuron to compute the output is called a forward propagation algorithm.

2.1.2 Backpropagation Algorithm

It is not enough to know only the neuron in neural networks because the parameters w and b need to be learned. Therefore, it is necessary to introduce a BP algorithm to update the parameters.

In supervised learning, we have a dataset containing the inputs and the corresponding outputs. We call the correct output a label. The purpose of training a neural network is to learn the correct mappings of inputs to outputs.

First, we assign random values to the parameters w and b, and generate the predicted values using the forward propagation algorithm. We define a cost function $J(w, b)$ to represent the closeness of the predicted output to the label. We use cross-entropy as the cost function in classification problems where the output is a discrete variable, and the mean-squared error is used as the cost function in regression problems where the output is a continuous variable.

The optimization formulas for the cross-entropy cost function and the mean-squared error cost function are expressed as follows:

$$\min J(w, b) = -\sum_{i=1}^{K} y_i \log p_i \tag{3}$$

$$\min J(w, b) = \frac{1}{K} \sum_{i=1}^{K} (y_i - p_i)^2 \tag{4}$$

where K is the number of samples, y_i is the label value of the i-th sample, and p_i is the predicted value of the i-th sample.

Thus, the objective is transformed into solving for the parameter values that minimize the cost function $J(w, b)$. The gradient descent algorithm is generally used to solve optimization problems. The gradient indicates that the directional derivative of a function at that point has a maximum value along that direction. The specific process is as follows: it first randomly selects a set of parameter values. Then, it computes the predicted output and the cost function $J(w, b)$, and computes the gradient of the cost function $J(w, b)$ on w and b. Finally, we use the gradient to update w and b so that the cost function takes the minimum value. The modified expressions for the parameters w and b are expressed as:

$$w := w - \alpha \frac{\partial J(w, b)}{\partial w} \tag{5}$$

$$b := b - \alpha \frac{\partial J(w, b)}{\partial b} \tag{6}$$

where α is the learning factor and denotes the step length of each gradient descent.

The process of updating the parameters w and b along the gradient direction to minimize the cost function is called the gradient descent algorithm.

The BP algorithm is a specific implementation of the gradient descent method on deep neural networks. Owing to the deepening of the network layers, the gradient of the parameters of each layer must be computed from backward to forward for the cost function and be continuously updated.

2.1.3 Single-layer Feedforward Neural Network

The FNN is the most common type of neural network in which neurons are arranged in layers. Each layer has several neurons, and each neuron is connected only to the neurons in the previous layer. Neurons in each layer only receive the input signal from the previous layer and output the processed signal to the neurons in the next layer. The first layer is called the input layer and the last layer is called the output layer. The remaining intermediate layers are called hidden layers; these can be one or more layers. The signal enters the network from the input layer and is transmitted layer by layer to the output layer. There is no feedback in the entire network, and the output does not affect the network model or network input. This section depicts a single-layer FNN containing one hidden layer as an example; this will allow us to introduce the basic principles of FNNs.

Figure 5 shows the general structure of a single-layer FNN, where $x_m^{(n)}$ denotes the n-th feature of the m-th sample, and the number of neurons in the input layer is the same as the number of features of the sample. $y_m^{(k)}$ denotes the k-th output of the m-th sample and k is the number of neurons in the output layer, where $k \geq 2$ for the classification problem and $k = 1$ for the regression problem.

The learning process of single-layer FNNs consists of forward propagation and BP; this is illustrated below with a simple network in Fig. 6. The input is propagated forward along the direction of the network structure to the output layer, and then the weights and bias are updated by the BP algorithm.

In the forward propagation, information is propagated from the input layer to the output layer after being processed by the hidden layer, and the state of the neurons in each layer only affects the state of neurons in the next layer. Assuming that the input layer is the 0-th layer, the output of the neurons in the hidden layer and the output layer can be expressed as:

$$h_1^{(1)} = \varphi^{(1)}(x_m^{(1)} * w_1^{11} + x_m^{(2)} * w_1^{21} + x_m^{(3)} * w_1^{31} + b_1^{(1)}) \tag{7}$$

$$h_2^{(1)} = \varphi^{(1)}(x_m^{(1)} * w_1^{12} + x_m^{(2)} * w_1^{22} + x_m^{(3)} * w_1^{32} + b_2^{(1)}) \tag{8}$$

$$y = \varphi^{(2)}(h_1^{(1)} * w_2^{11} + h_2^{(1)} * w_2^{21} + b_1^{(2)}) \tag{9}$$

where w_l^{ij} denotes the connection weight of the i-th neuron in layer $l-1$ to the j-th neuron in layer l, $b_j^{(l)}$ denotes the bias used to compute the linear weighted summation

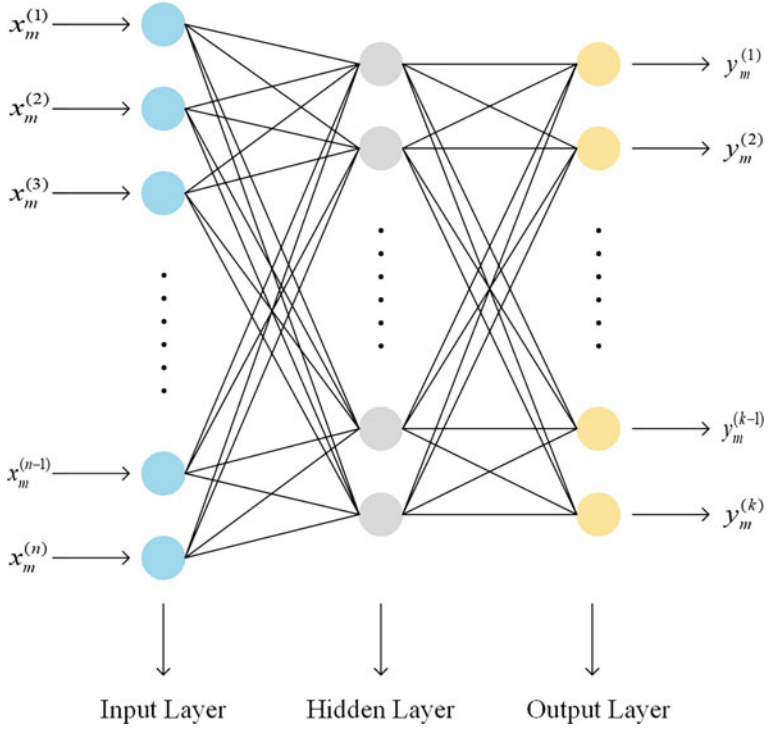

Fig. 5 The general structure of single-layer feedforward neural networks. The blue, gray, and yellow neurons form the input layer, hidden layer, and output layer of the neural network, respectively

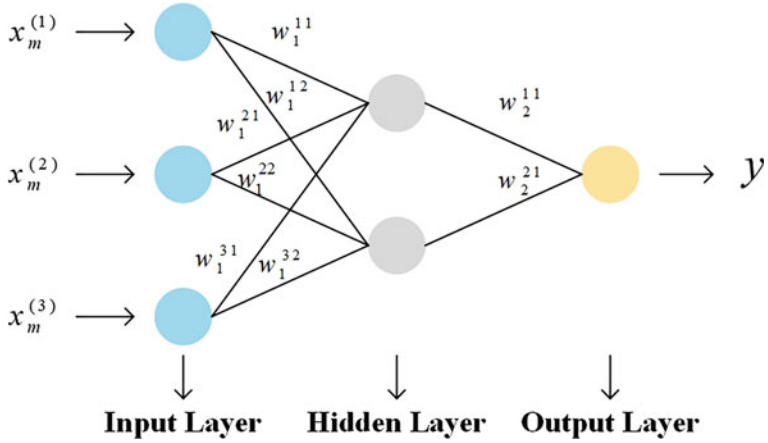

Fig. 6 A simple single-layer feedforward neural network

of the j-th neuron in layer l, $\varphi^{(l)}$ denotes the activation function of the neuron in layer l, $h_j^{(1)}$ denotes the output of the j-th neuron in the hidden layer, and y denotes the output of the neuron in the output layer.

In BP, the gap between the network output and the real value is first calculated; this is called the loss function. Subsequently, the gradient of the parameters, such as the weight and bias term, is calculated, and the parameters are updated by the gradient descent to minimize the loss function. The learning process of the network is continuously iterated to optimize the network model until the loss function is sufficiently small or the maximum number of learning times is reached.

When the structure and weights of a neural network are determined, the network forms a nonlinear mapping from the input to the output. For a single-layer FNN, if the number of neurons in the hidden layer is large enough, it can approximate any continuous function on a bounded region with arbitrary accuracy and can solve complex nonlinear classification tasks well.

2.1.4 Multi-layer Feedforward Neural Network

A network that contains multiple hidden layers between the input and output layers is called a multi-layer FNN, which can also be called a deep FNN. Depth refers to the number of layers in a neural network model. The greater the number of layers designed, the greater the depth and complexity of the model. A deep FNN has better classification and memory ability, as well as a stronger function fitting ability. It can handle more complex data structures or data whose structures are difficult to predefine. Figure 7 shows a neural network containing four hidden layers.

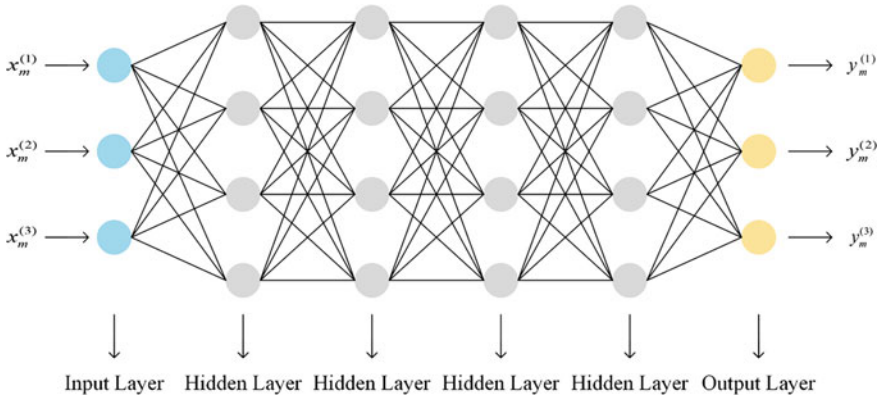

Fig. 7 A multi-layer feedforward neural network containing four hidden layers

The output of the network can be expressed as follows:

$$z^{(l)} = W^{(l)} * a^{(l-1)} + b^{(l)} \tag{10}$$

$$a^{(l)} = f_l(z^{(l)}) \tag{11}$$

where $f_l(\bullet)$ denotes the activation function of neurons in layer l, $W^{(l)}$ and $b^{(l)}$ denote the weight matrix and bias from layer $l-1$ to layer l, respectively, $z^{(l)}$ and $a^{(l)}$ denote the net input and output of neurons in layer l.

Deep FNN also uses a BP algorithm to optimize the network model. The representation ability of the network is greatly enhanced because of the increase in the number of layers and parameters. However, the network is prone to overfitting; that is, small errors can be obtained from the training data, and large errors are obtained from the test data. Therefore, appropriate regularization techniques need to be applied to improve the generalization ability of the network model.

Deep FNNs are the basis of many AI applications. They can perform complex data processing and pattern recognition tasks. However, they cannot process image data well. The networks require a large number of neurons and parameters when processing the image data; with high computational ability requirements and low computational efficiency, these networks are prone to overfitting.

2.2 Deep Convolutional Neural Network

The biggest advantage of CNNs over FNNs is the reduction of parameters. This allows researchers to build and design larger models to solve complex problems. For example, a picture in jpg format with a resolution of 480 × 480 is represented in the computer as a 480 × 480 × 3 tensor, and the three dimensions correspond to the height, width, and the number of channels of the 3D tensor. If this image data is fed into an FNN, each neuron in the first hidden layer in this network needs to be connected to 691,200 (480 × 480 × 3) tensor elements. The number of parameters required for one of these neurons is over 600,000. Thus, the number of single-layer network neurons required for FNNs to deal with complex problems is enormous. Therefore, the hidden layer of the FNN requires a large number of parameters to extract the tensor features. This fully connected mechanism of FNNs is inefficient in handling large input data. Compared with the fully connected layer of the FNN, the convolutional layer requires fewer parameters to extract the tensor features. CNNs are widely used in many fields because of their efficient features.

This section contains four subsections. Section 2.2.1 introduces the mechanism of the CNNs and their basic structures. Section 2.2.2 introduces the mechanism of full CNNs. Section 2.2.3 introduces typical CNN structures. Section 2.2.4 introduces the problems and shortcomings of deep convolutional networks.

2.2.1 Mechanism of Convolutional Neural Network

CNNs obtain their names from the mathematical linear operations between the matrices called convolutions. A CNN is a representation learning method with a multilayer structure, which mainly consists of a convolutional layer, pooling layer, and a fully connected layer. The convolutional and fully connected layers contain parameters, whereas the pooling layer does not. As shown in Fig. 8, the image is input to the CNN and then passes through the convolutional and pooling layers alternately, which flattens the image features into a feature vector of dimension one. The CNN finally outputs the result through the fully connected layer. The convolutional and pooling layers are equivalent to feature extraction structures, which are used to extract features from the input tensor. The fully connected layer is equivalent to a classifier, which is used to classify the flattened feature vector.

As the name implies, the convolutional layer is the most important operation in a CNN, and the parameters of the convolutional layer are mainly located in the convolutional kernel. The convolution kernel is usually represented by a small-size tensor that only acts on a local region within the space of the input tensor.

Taking the most widely used two-dimensional convolution as an example, the specific process of the convolution operation is shown in Fig. 9a. The convolution operation selects all local regions in the spatial dimension of the input features that are consistent with the size of the convolution kernel. The calculation is shown in Fig. 9a. The input tensor shares the convolution kernel parameters in the channel dimension when performing the convolution operation. The convolution operation is a three-dimensional inner product operation between the shared convolution kernel parameters along the channel and the features of the same spatially localized region on different channels. The corresponding result is a scalar of the center position of the corresponding spatially localized region. The output corresponding to a single convolution kernel is a two-dimensional feature map, and the number of convolution kernels in the convolution layer is the same as the number of channels in the output tensor.

There is some degree of information overlap between the output results of the convolutional layer and its neighboring outputs in the spatial dimension. This may lead to information redundancy. As shown in Fig. 9b, the input feature map size of the convolution operation is 7 × 7. If the selected convolution kernel size is 3 ×

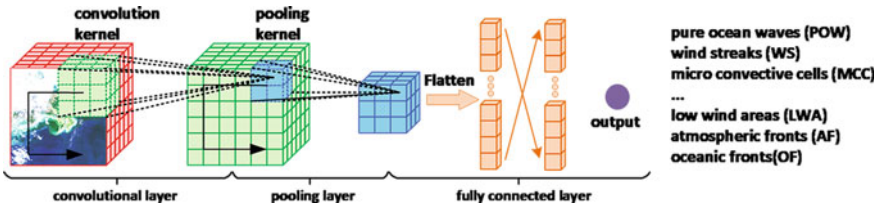

Fig. 8 Overview of the convolutional neural network architecture. The architecture mainly consists of several convolution layers, pooling layers and full connection layers

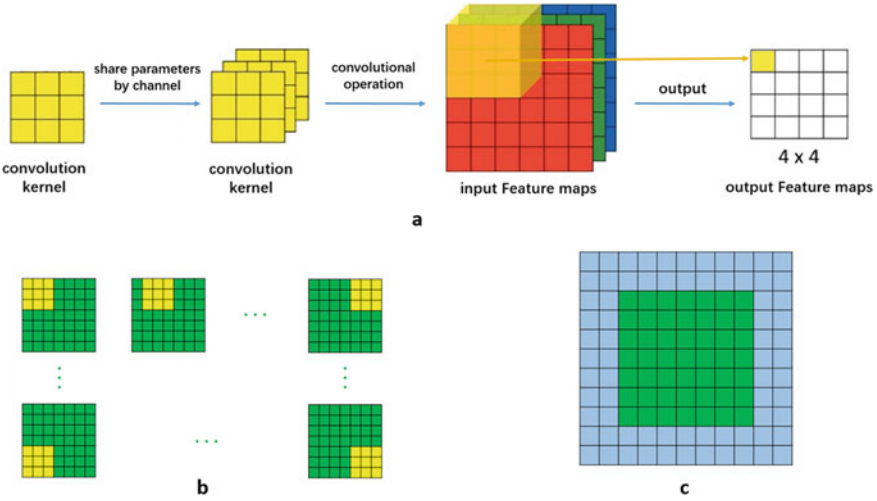

Fig. 9 Illustration of Convolution Implementation Mechanism. **a** Convolution kernel parameters are shared in the input tensor channel dimension. **b** The sliding mechanism of the convolution kernel in the dimension of the input tensor space. **c** Padding operation in the process of convolution implementation

3, then there will be an overlap of six elements between two adjacent convolution operations. If the step size parameter is set to two, there will be an overlap of three elements between two adjacent convolution operations. The size of the output feature map of the convolution operation is 3×3. A reasonable increase in the step size reduces the overlap between adjacent convolution operations and reduces the size of the output feature.

Convolution operations lose information about edge features, which is an inherent drawback of convolution operations. A simple and effective method is padding. As shown in Fig. 9c, if the size of the convolution kernel is 5×5 and the padding parameter is set to two, the size of the output feature map of the convolution operation can be kept consistent with the input to avoid the loss of the edge information.

Although the convolutional layer can serve to reduce the number of connections between the output neurons and the input features, the number of neurons is not significantly reduced during the execution of the convolutional operation, and the feature dimensionality remains high. To make the model easier to optimize, pooling layers are introduced to the CNN. The pooling layer is often located between two convolutional layers and is designed to perform feature selection and reduce the dimensionality of the feature mapping. There are two mainstream pooling approaches: maximum pooling and average pooling. As shown in Fig. 10, a 2×2 filter is used on the input features with a spatial dimension of 4×4 and slides with a step size of two. The maximum and average feature values in the corresponding position of each filter are determined, and the maximum pooled feature map and the average pooled feature map are the outputs.

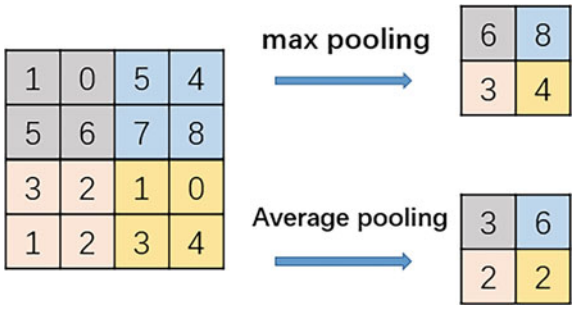

Fig. 10 Comparison of two different pooling operations

The fully connected layer in a CNN is composed of the single-layer feedforward network introduced in the previous section. The number of layers of this fully connected layer can be chosen according to the specific needs of the task. Compared with the convolutional layer, the fully connected layer contains a large number of parameters and dense connections. Therefore, more fully connected layers will lead to a more difficult model optimization; thus, the number of fully connected layers in mainstream CNNs does not exceed three.

2.2.2 Mechanism of a Fully Convolutional Neural Network

In contrast to the CNN, the fully CNN does not contain a fully connected layer; its structure is shown in Fig. 11. Each element of the output layer of the convolutional and pooling operations represents the local information of the input features, whereas the fully connected layer depicts the global information of the input features; thus, the convolutional and pooling operations can preserve the spatial dimensional information of the input tensor. A fully CNN composed of all convolutional and pooling operations can output a feature map that retains the spatial location information. Compared to traditional CNNs, fully CNNs are more suitable for tasks such as image segmentation, where both the input and output are images.

2.2.3 Common Convolutional Neural Networks

During the development of CNNs, several representative networks have emerged, such as VGGNet, ResNet, and DenseNet. In this section, we introduce these networks.

VGGNet has two structures of 16 and 19 layers, as shown in Fig. 11. All the convolution kernels in the VGGNet network are of size 3×3. VGGNet cascades three sets of convolution operations with a kernel size of 3×3 and a step size of one. These three sets of convolution operations are equivalent to one convolution operation with a kernel size of 7×7. This has two main benefits. First, a deeper network structure will learn more complex nonlinear relationships, which will lead to

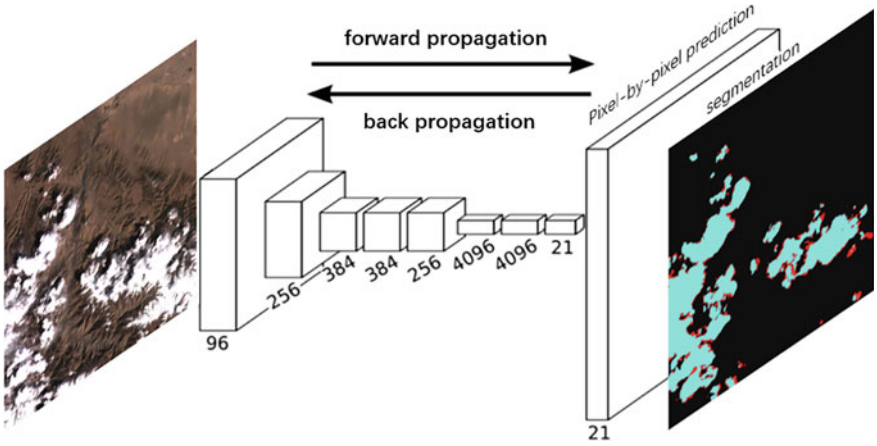

Fig. 11 Overview of the fully convolutional neural network architecture. The input and output of this architecture are pictures

better results for the model. Second, this reduces the number of parameters required to perform the convolutional operations.

In 2015, Kaiming He proposed a 152-layer ResNet to win the 2015 ILSVRC competition with a top-1 error record of 3.6%. The proposed ResNet was revolutionary because the network introduced a residual mechanism. Compared to the traditional convolutional module, the convolutional module in the residual network learns only a small variation in the input features. The output of the residual convolution module is equivalent to the superposition of the input features and the amount of variation in the input features. When BP is performed, the residual structure retains some of the gradient information and alleviates the gradient disappearance problem. Therefore, the ResNet model is deeper and has a better performance.

ResNet uses jump connections to pass gradients directly from the back layer to the front layer. However, features that have been jump-connected and features that have been convolutionally transformed need to be summed before they can be output. This may affect the information propagation in the network. In response, Huang et al. [27] proposed the network structure of DenseNet, which is based on ResNet. Unlike ResNet, which only forms a jump-connected module with the previous layer, DenseNet achieves feature reuse by directly connecting each layer to its preceding layer. Compared to ResNet, DenseNet not only reduces the error rate but also reduces the number of parameters in the network.

2.2.4 The Shortcomings of Convolutional Neural Network

The convolutional operations of CNNs can be divided into one-dimensional, two-dimensional, and three-dimensional convolutions according to the dimensions of the convolutional kernel. One-dimensional convolution is mainly applied to tasks

related to one-dimensional sequence signals, such as EEG signal analysis, speech signal processing, and radio signal classification. Two-dimensional convolution is mainly used in the image field, such as image super-resolution and image denoising; image restoration, such as the image processing field, image recognition, and target detection, and semantic segmentation, such as the computer vision field. Three-dimensional convolution is commonly used in medical CT image segmentation and video motion recognition, etc.

Although CNNs have powerful feature extraction capabilities, they are based on the assumption of mutual independence between consecutive input samples. Therefore, the network is difficult to apply to tasks where there is an inherent logical relationship between successive inputs.

2.3 Deep Recurrent Neural Network

2.3.1 Mechanism of Convolutional Neural Network

Jordan [29] proposed Jordan Network in 1986 and designed a memory mechanism that fed back the output of the entire network to the input layer of the network the next moment. One of the foundational works done on RNNs is the simple recurrent networks (SRNs), which was proposed by Elman in 1900 [12]. The SRN was modified on the Jordan Network, and the output of the hidden layer in the network is shown below. Feedback to the input layer occurs at any given moment. The Jordan Network uses the entire network as a loop, whereas the SRN only uses the hidden layer as a loop. Therefore, the SRN is more flexible to use; it also avoids the problem of conversion between network output dimensions and input dimensions.

The structures of the SRN and the widely used RNN are similar. The output value of the RNN at the next moment is jointly determined by multiple past moments. In fact, there is often a problem that the output of the network is affected by the future inputs.

Driven by similar ideas, Schuster and Paliwal [45] improved the traditional RNN by designing a bidirectional cyclic neural network (bidirectional RNN, BRNN). The BRNN is a superposition of two RNNs in opposite directions. Each hidden layer must record two values for both the positive and negative directions. The final output value depends on the RNN calculated in the forward and backward directions (Fig. 12).

Hochreiter and Schmidhuber [25] proposed an LSTM. In RNNs, owing to the uncertain attenuation of information in the cyclic structure, it takes a lot of time to learn to store information over a certain time interval through the BP of the periodic structure. In this problem, the solution used by LSTM is the structure of the input gate, and the output gate is designed in the RNN to control the state and output of the loop unit at any given time. Gers et al. [16] proposed that LSTM did not have a clear prior end mark when processing long sequence inputs. Hence, the original LSTM added a forget gate mechanism, which allowed the LSTM to learn to reproduce at the appropriate time and set itself to release the internally stored information. One year

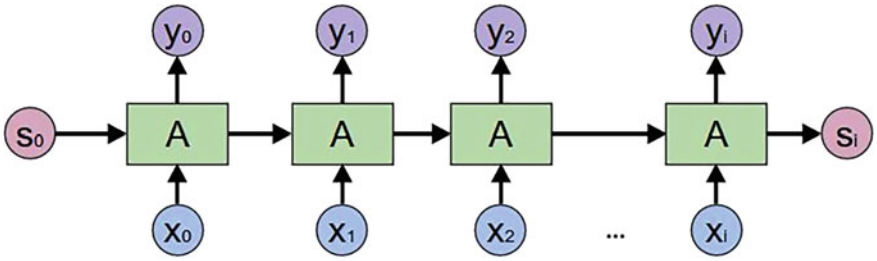

Fig. 12 The structure of the recurrent neural network. The blue neuron represents the time series of sequential input. The purple neuron represents the output of the recurrent network. The pink neurons represent the information transmitted in the middle

later, Gers and Schmidhuber [15], made improvements to the LSTM again, adding a peephole connection; the value memorized in the last moment was also used as the input of the gate structure; thus the structure of LSTM gradually improved to its present-day state.

Graves and Schmidhuber [18] proposed a bidirectional LSTM (BLSTM), which combined the BRNN and LSTM. Graves et al. [19] applied the LSTM to handwritten text recognition tasks, which overcame the difficulty faced by traditional models in segmenting scribbled and overlapping text. The offline text recognition rate was 74.1%, which was the best at that time. Graves et al. [20] combined deep networks and RNNs and proposed a deep recurrent network based on LSTM, which was applied to speech recognition tasks. The error in the TIMIT dataset was only 17.7%, which was the best at that time. In 2014, Cho et al. [7], simplified its gate structure by replacing the input gate, forget gate, and output gate of LSTM with an update gate and reset gate, and proposed a gated recurrent unit for the first time. For traditional statistical machine translation, the encoder-decoder model was proposed.

Owing to the further development of the LSTM, the LSTM has been widely used in various applications of natural language processing and many variants have been developed. Next, we introduce LSTM and ConvLSTM in detail.

2.3.2 Mechanism of LSTM

Recurrent connections can improve the performance of neural networks by leveraging their ability to understand sequential dependencies. However, the memory produced from recurrent connections can be severely limited by the algorithms employed for training the RNNs. To date, all models have been affected by the explosion or disappearance of gradients in the training phase, which renders the network unable to learn the long-term order dependence in the data. The LSTM was specifically designed to solve this problem.

LSTM is one of the most commonly used and effective methods for reducing the vanishing gradient and exploding gradient. It changes the structure of the hidden

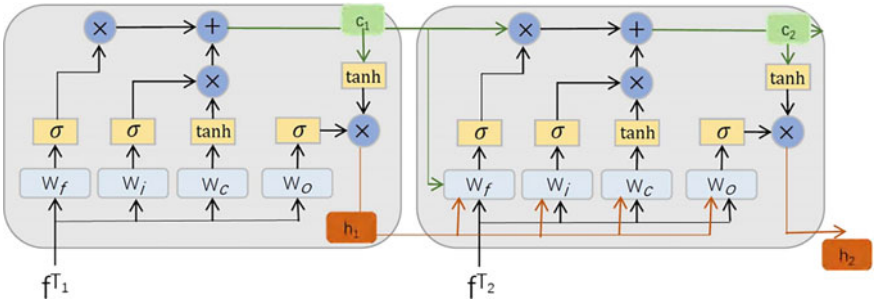

Fig. 13 The structure of LSTM

unit, and the input and output of the neural unit are controlled by the gates. These gates control the information flow of hidden neurons and retain the features extracted from the previous time steps.

As illustrated in Fig. 13, an LSTM-based recurrent layer maintains a series of memory cells c_t at time step t. The activation of the LSTM units can be calculated by:

$$h_t = o_t \times \tanh(c_t) \tag{12}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, and O_t is the output gate that controls the extent to which the memory content is exposed. The output gates are updated as:

$$o_t = \sigma(W_o^T \times f^{T_t} + U_o^T \times h_{t-1} + b_o) \tag{13}$$

where W_o and U_o are the input-output weight matrix and memory-output matrix, respectively, and b_o is the bias. The memory cell c_t is updated by partially discarding the present memory contents and adding new contents of the memory cells $\widetilde{c_t}$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \widetilde{c_t} \tag{14}$$

where $\otimes$ is the element-wise multiplication. The new memory contents are

$$\widetilde{c_t} = \tanh(W_c^T \times f^{T_t} + U_c^T \times h_{t-1} + b_c) \tag{15}$$

Here, W_c and U_c are the input-memory weight matrix and hidden memory coefficient matrix, respectively; b_c is the bias; i_t is the input gate, which modulates the extent to which the new memory information is added to the memory cell; f_t is the forget gate, which controls the degree to which the contents of the existing memory cells are forgotten. The gates are computed as follows:

$$i_t = \sigma(W_i^T \times f^{T_t} + U_i^T \times h_{t-1} + V_i^t \times c_{t-1} + b_i) \tag{16}$$

$$f_t = \sigma(W_f^T \times f^{T_t} + U_f^T \times h_{t-1} + V_f^t \times c_{t-1} + b_f) \tag{17}$$

where W_i, U_i, and V_i are the input-memory weight matrix of the input gate, hidden-memory coefficient matrix, and output weight matrix of the previous cell state, respectively; W_f , U_f, and V_f are the input-memory weight matrix of the output gate, hidden-memory coefficient matrix, and output weight matrix of the previous cell state, respectively; b_i and b_f are the biases.

2.3.3 Mechanism of convLSTM

The CNN does not have additional complex operations of artificial neural networks for preprocessing and spatial distribution. Therefore, it uses a unique fine-grained feature extraction method to automatically process the spatial data. When dealing with time features, LSTM can effectively avoid the disappearance of valid information because of the long data interval span. There are certain limitations if parallel CNN and LSTM are used to extract spatial and temporal features. For example, in a parallel structure composed of a CNN and LSTM, the input and output of the two are relatively independent, and the extraction of the relationship between different features is ignored.

Conv-LSTM was born out of a precipitation prediction problem [54]. The problem is as follows: given a map of the precipitation distribution for the first few hours, predict the precipitation distribution for the next few hours. This was accomplished by replacing the input-to-state and state-to-state parts of the LSTM from feedforward calculations to convolution calculations. A cell diagram is shown in Fig. 14.

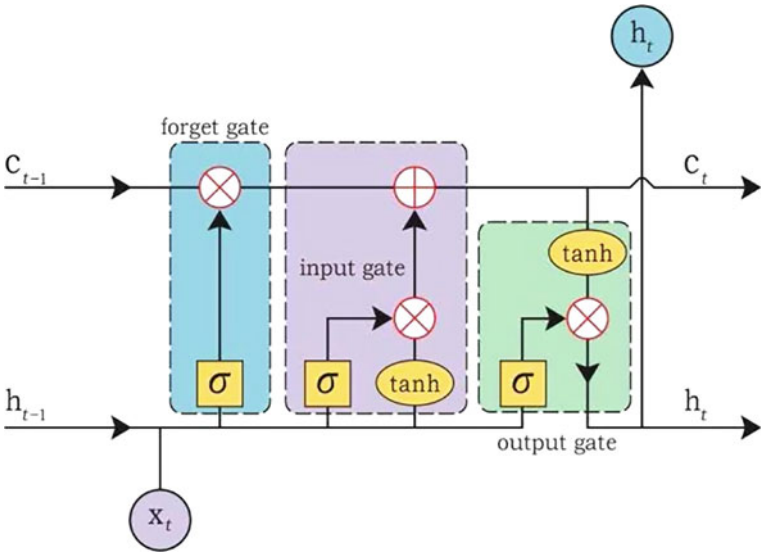

Fig. 14 The structure of Conv-LSTM

The principle of Conv-LSTM can be expressed by the following formula:

$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \tag{18}$$

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \tag{19}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(Wxc * x_t + W_hc * h_{t-1} + b_c) \tag{20}$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{c0} \circ c_{t-1} + b_o) \tag{21}$$

$$h_t = o_t \circ \tanh(c_t) \tag{22}$$

where $*$ denotes the convolution and x, c, h, i, f, o are tensors. We can imagine Conv-LSTM as models that work on the eigenvectors of two-dimensional grids. It can predict the time-space features of the central grid based on the time-space features of the points around them.

2.4 Deep Generative Adversarial Network

CNNs and RNNs have been widely used in various fields, and have achieved good results. However, these methods need to rely on a large amount of labeled data. In actual research, we often encounter insufficient training sample data. This situation will lead to a decline in the recognition accuracy of our model. A generative adversarial network (GAN) can generate realistic sample data. If these generated sample data are used to train the model, the problem regarding the amount of training sample data can be solved. At present, GANs have been widely used in the fields of image and vision.

2.4.1 Architecture of Generative Adversarial Network

GAN [17] originated from the two-person zero-sum game. The two-person zero-sum game is a concept in game theory, which says that the sum of the interests of both parties in the game is always zero or remains unchanged. If one party gains, the other party must have a corresponding loss. GAN is composed of two parts: a generator and a discriminator. These two parts can be regarded as the two parties of the game. The optimization process of a GAN is equivalent to the two-person zero-sum game process.

The purpose of the GAN is to learn the distribution of real data, which can generate realistic data. The implementation of a GAN is shown in Fig. 15.

The generator G is used to capture the distribution of real data, and random noise is used as an input to generate the sample data. To capture the real data distribution, first, a random noise z that obeys the prior distribution $P_z(z)$ is given. Then the

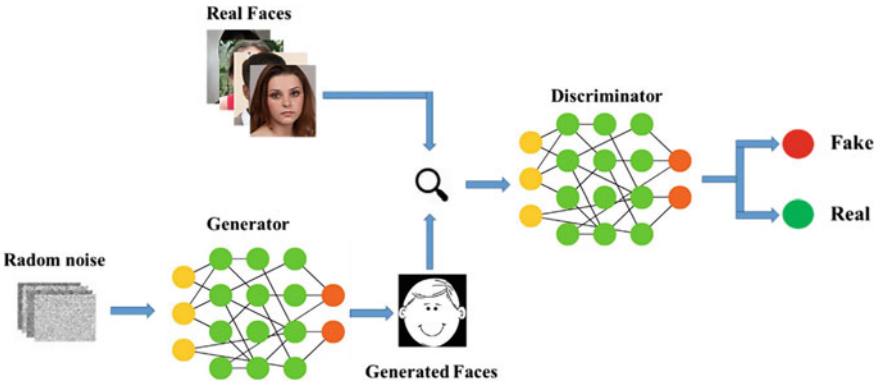

Fig. 15 The basic structure of the generative adversarial network

mapping space $G(z;\theta_g)$ is constructed using $P_z(z)$. The mapping space $G(z;\theta_g)$ is a generative model of the parameters θ_g. The random noise z is used as the input of the generator, and sample data is generated through the mapping space $G(z;\theta_g)$.

The discriminator D is used to determine whether the input sample is a real sample or a generated sample, which is equivalent to a two-classifier. Defining the mapping function $D(z;\theta_d)$, the input sample outputs a scalar between zero and one through the mapping function. This scalar represents the probability that the input sample is a real sample; the mapping function $D(z;\theta_d)$ is a discriminator for the parameters θ_d. It should be noted that the input of the discriminator consists of two parts. One part is the sample generated by the generator, and the other part is the real sample x that obeys the real data distribution.

2.4.2 Training of Generative Adversarial Network

In the GAN training process, generator G and discriminator D compete with each other, continuously alternating the iterative optimization. Finally, they gradually reach an equilibrium. The optimization function of GAN is as follows:

$$\max_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[\log D(x)] + E_{z\sim P_z(z)}[log(1-D(G(z)))] \quad (23)$$

where *D(x)* represents the probability that the real sample x is identified as a real sample after passing through the discriminator. *G(z)* represents the sample data generated by random noise z through the generator. *D(G(x))* represents the probability of the generated sample data being judged as a real sample after passing through the discriminator.

It can be seen that the optimization function of the GAN is equivalent to a min-max optimization problem. This optimization function has two steps. The first step is to optimize the discriminator D, and the second step is to optimize generator G.

The objective function can be regarded as alternately optimizing the following two objective functions:

$$\max_{D} V(D, G) = E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_z(z)}[log(1 - D(G(z)))] \quad (24)$$

$$\min_{G} V(D, G) = E_{z \sim P_z(z)}[log(1 - D(G(z)))] \quad (25)$$

The label of the real sample is artificially defined as one, and the label of the generated sample is zero. The purpose of the discriminator is to distinguish between true and false samples. Thus, the discriminator hopes that *D(x)* is as close to one as possible, and *D(G(x))* is as close to zero as possible. Using sample data to optimize the discriminator based on these two conditions is equivalent to maximizing *V(D,G)*. The purpose of the generator is to generate sufficiently realistic sample data; hence, the generator hopes that *D(G(z))* is as close to one as possible. For this purpose, using sample data to optimize the generator is equivalent to minimizing *V(D,G)*.

It should be noted that the parameters of *G* or *D* are always fixed, and the parameters of the other part are updated during training. Finally, *P(z)* and $P_{data}(x)$ are infinitely close. The generator can generate samples in which the discriminator cannot distinguish authenticity.

2.4.3 Typical Generative Adversarial Networks

The conditional GAN [30] adds constraints to the standard GAN to guide the data generation process, thereby generating controllable samples. It solves the problem of GAN image generation being too free and difficult to control. The constraint condition can be the category label or semantics of the image. The implementation process of the conditional generation confrontation network is illustrated in Fig. 16.

The deep convolution generative adversarial network (DCGAN) [51] combines CNN and GAN to optimize the original GAN model from the network structure. DCGAN replaces the generator and discriminator in the original GAN with two CNNs to improve the quality of sample generation and the speed of model convergence.

2.4.4 Application of Generative Adversarial Network

A GAN can generate real-like samples without explicitly modeling any data distribution in the process of generating samples. Therefore, GANs have a wide range of applications in many fields, such as images and text.

One function of the GAN is to generate the data. A limitation of the development of deep learning is the lack of training data; only GAN-generated data can compensate for this shortcoming. For example, given the text description of a bird, such as some

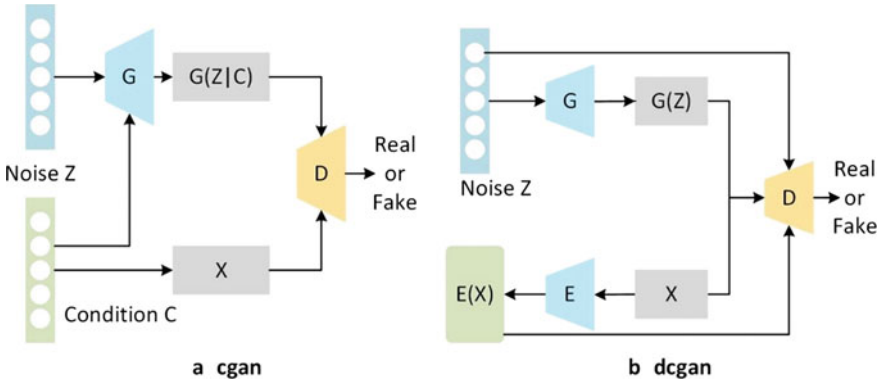

Fig. 16 Common Generative Adversarial Network Structure. **a** conditional generative adversarial nets. A network with conditional constraints (green tensors in the figure) **b** deep convolutional generative adversarial networks. A network composed of convolutional layers

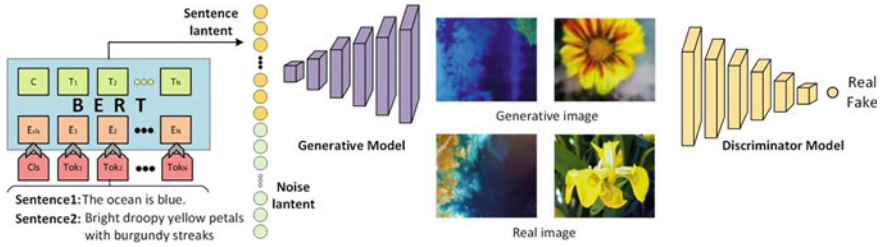

Fig. 17 Cross-modal image generation. The generative adversarial network can realize the task of generating from text to image

black and white on its head and wings, and a long orange beak, a trained GAN can generate images that match the description. Figure 17 shows the application of GAN in image generation

Another important application of GAN is image super-resolution, which refers to the process of recovering high-resolution (HR) images from low-resolution (LR) images. This is an important class of image processing techniques in computer vision and image processing. It enjoys a wide range of real-world applications, such as medical imaging, surveillance, and security amongst others. Other than improving image perceptual quality, it also helps to improve other computer vision tasks. However, generally, this problem is very challenging and inherently ill-posed since there are always multiple HR images corresponding to a single LR image. As shown in Fig. 18, relying on powerful image generation capabilities, GAN can decode and encode LR images into HR images.

The task of image translation can also be achieved through GAN. Image translation is the conversion of one (source domain) image to another (target domain) image. During the translation, the content of the source domain image will remain

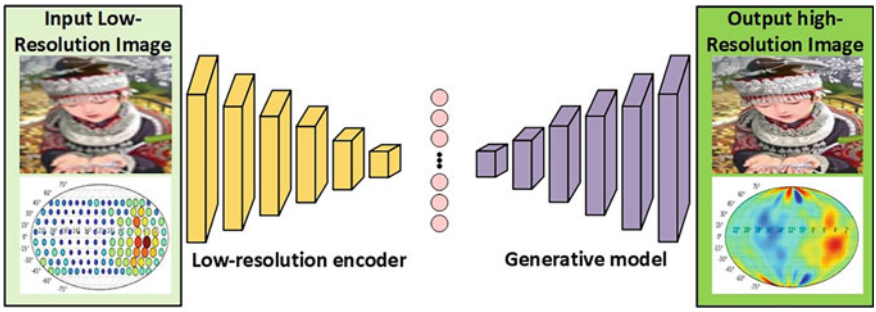

Fig. 18 Schematic diagram of super resolution. In the computer vision community, the super-resolution task makes blurry images clear. In oceanography, the purpose of the super-resolution task is to improve the temporal and spatial resolution of the numerical field

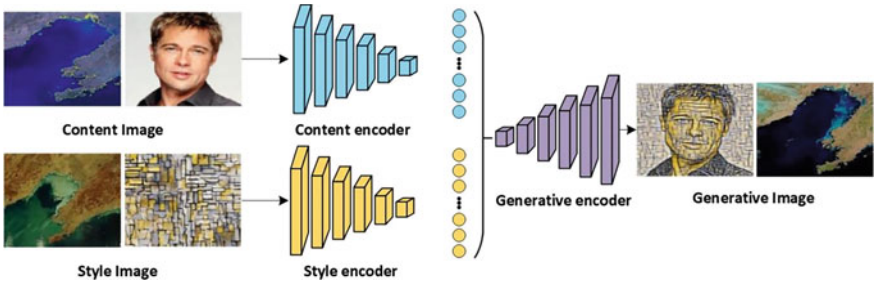

Fig. 19 Schematic diagram of image translation. This task merges two images to get a brand-new image, the generated image retains the content of one input and the style of the other input

unchanged. Nevertheless, the style or other attributes will be the same as the target domain, as shown in Fig. 19.

Image restoration is a technology that uses the learned image information to complement or modify the damaged image. Image restoration has various applications such as image completion and image deblurring. Owing to its good ability to fit the real distribution, GAN has shown good results in image restoration.

2.4.5 Problems of Generative Adversarial Network

GAN has become a popular research topic in recent years. Despite its recent genesis, it has developed rapidly and has made important contributions in many fields. However, owing to problems such as model collapse and gradient disappearance, its generation effect, training efficiency, and application range are still restricted.

(1) Low image-generation diversity

The diversity of image generation has always been an important issue in the field of GAN research. Traditional GAN algorithms can only fit simple datasets with small sizes, and the complexity of image generation is low. Therefore, the GAN

algorithm has been developed for image diversity. The existing GAN algorithm can generate indistinguishable high-quality images; however, many factors restrict the development of image diversity, which often conflict with other factors such as image size and model complexity.

(2) Insufficient model training efficiency

GAN has training instability problems, which are caused by model collapse and gradient disappearance. In addition, the complex model structure and redundant information causes the training cycle to be too long.

(3) The application field has not been extensively studied

GAN has been used in many fields in a relatively short period; Nonetheless, it is mostly limited to image processing. Many algorithms mention only their achievable functions without explaining their use-value. This development is slow in other fields such as NLP.

3 Perceptual Understanding Based on Neural Network

3.1 Recognition Based on Neural Network

Various neural network architectures support a wide variety of perceptual understanding applications. Currently, research on neural networks in natural language processing, visual data processing, speech signal processing, etc. is progressing rapidly. They have been widely used in industrial fields such as intelligent security, medical health, and industrial inspection. Figure 20 briefly depicts the applications of neural network-based deep learning research in some important fields. This section introduces neural network-based recognition, segmentation, and prediction applications.

3.1.1 Problem Description

Neural network-based recognition tasks involve both the extraction of features from the model input content and the establishment of mapping relationships between the extracted features and the identifiable attributes of the sample (category, location, etc.). The advent of CNNs has led to the rapid development of neural networks for visual recognition tasks. In this subsection, the classification, localization, and detection in recognition tasks are described.

Figure 21 depicts the flow of a neural network-based recognition task. First, the input image is subjected to a CNN to extract key features and represent them in a one-dimensional feature vector. Then, this feature vector is input to the classification module, localization module, or target detection module according to the different tasks; the corresponding output results are subsequently obtained. The classification module is used to determine the category of the target in the input image. The localization module is used to determine the location of the target in the picture.

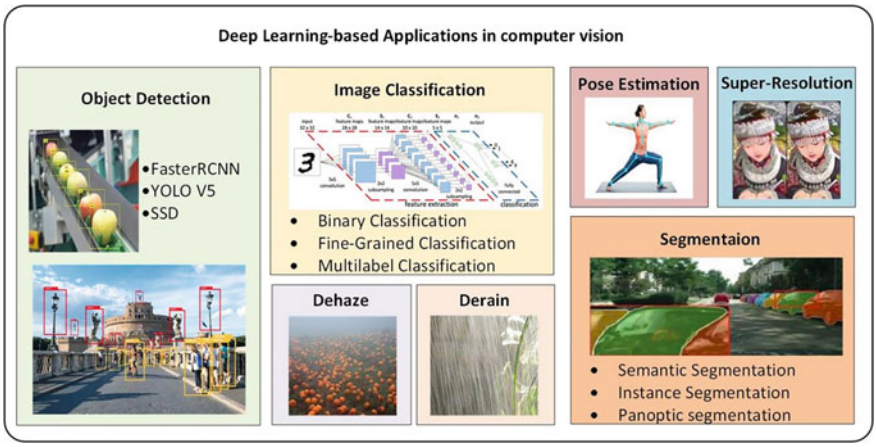

Fig. 20 Deep learning-based applications in computer vision. It briefly depicts applications of neural network-based deep learning research in some important fields

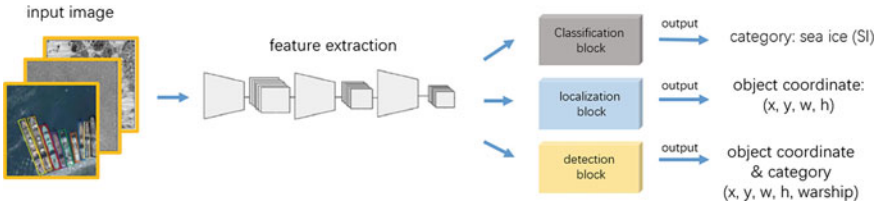

Fig. 21 Overview of a neural network-based recognition task. It contains the three basic computer vision tasks of classification localization and detection

The target detection module is a combination of classification and localization, that is, it determines the location of the target in the input image and its corresponding category. The next section describes each of these three recognition tasks and their corresponding modules.

3.1.2 Classification

In computer vision, image classification is a crucial job. The objective of image classification is to discern what category the object in the image belongs to, such as whether it is a cat or a dog. As shown in Fig. 22, according to the difficulty of the classification task, it can be subdivided into dichotomous classification task, multi-classification task, or multi-label classification task, etc. The image classification task can be expressed by the following equation.

$$C = f_{fc}[f_{conv+pool}(x)] \tag{26}$$

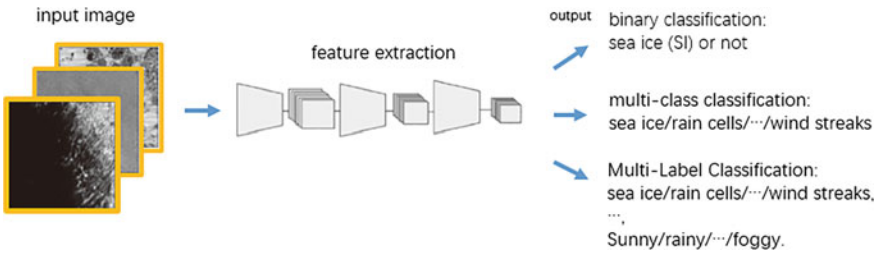

Fig. 22 Overview of a classification-based neural network. It includes binary classification, multi-class classification and multi-label classification

C denotes the category, $f_{\text{conv+pool}}$ denotes the convolutional and pooling layers, X denotes the input image, and f_{fc} denotes the fully connected layer.

Binary classification is the most basic type of image classification. It is used to identify whether the input image contains a certain category; the classification results are represented by zeros and ones. The binary classification task is used to identify whether the target is visible in the input. If the model result is zero, then it means that the input image does not contain the target object and vice versa.

Multi-category image classification is more widely used than binary classification. Its purpose is to classify the corresponding target class of an image that contains only one target class. Multi-category tasks are used to identify the specific category of the target in the input image, and the picture input to the network often contains only one category of targets. Multi-category image classification has now been integrated into all aspects of life and has been effectively used in a variety of sectors, such as facial recognition.

Multi-label classification is used to identify all the categories present in the input images. The pictures processed by this task often contain several different labels, and these labels are compatible with each other. The multi-label classification task can describe the information of pictures more graphically and has a more realistic meaning.

The success of deep learning classification tasks is inextricably linked to the development of supervised learning. The construction of large-scale datasets and the development of computational resources have made it possible to train the neural network parameters. The loss function, a metric used to measure how well a model predicts results, is an important component of the classification task and plays an important role in BP to update the network parameters. The purpose of the loss function is to update the parameters of the model to achieve better prediction results. Take the most commonly used cross-entropy loss function as an example, its loss function can be written as follows.

$$Loss = -\sum_{i=1}^{n} y_i \log(p(x_i)) \tag{27}$$

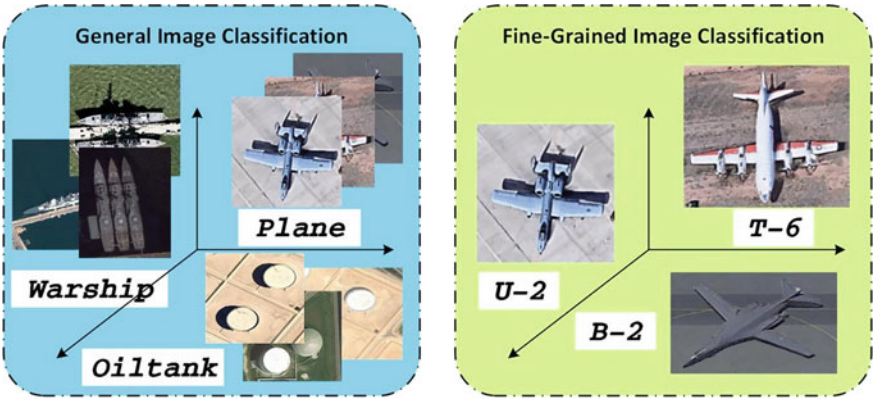

Fig. 23 Comparison of categories between traditional classification and fine-grained image classification. Fine-grained image classification can be used to distinguish different types of fighter jets

where $p(x_i)$ denotes the probability value that sample x is predicted to be the i-th category, y_i denotes the true label of the corresponding input sample x in the i-th category, which is one if it belongs to the i-th category, and zero otherwise.

For datasets such as ImageNet, which has more than 10 million images and 20,000 classes, the image classification level computer has surpassed that of humans; however, deep neural networks are not effective when recognizing subclasses under traditional categories, that is, discriminating between magpies and sparrows, etc., under the category of birds. Furthermore, the training of the model requires a large number of manually labeled tags, which is expensive. The cost of labeling increases exponentially with the number of targets and the difficulty of discernibility. To address these two challenges, fine-grained image classification and unsupervised image classification have emerged.

The distinction of fundamental categories and the performance of finer subcategories are the foundations of fine-grained image classification. Fine-grained image classification may be used to discriminate between sub-categories, such as various types of fighter airplanes, as illustrated in Fig. 23. Fine-grained image categorization may also be used to discern between different automobile types and battleship models, for example. This categorization has a wide range of practical uses.

Fine-grained images have a more similar appearance and features than coarse-grained images. In addition, there are the effects of poses, perspective, illumination, occlusion, and background interference, etc. in the acquisition. As a result, the data has a huge inter-class variability and a modest intra-class variability. This makes classification more difficult.

The above classification tasks are achieved via supervised learning. Each sample has its corresponding label, and the deep neural network is used to continuously learn the features corresponding to each label and achieve the classification. In this case, the size of the dataset and the quality of the labels often play a decisive role

in the performance of the model. High-quality datasets naturally bring difficulties in labeling and need a lot of human and financial resources. The aim of unsupervised image classification is to classify samples into several classes without using label information, thus greatly reducing the labor and time costs associated with data labeling.

3.1.3 Localization

The purpose of image localization is to determine the location of the target in the input image. An image is input to the model, and the model outputs the center coordinates, width, and height of the target location in the image. The result is a rectangular box with the location of the target in the input image, represented by the horizontal and vertical coordinates of the rectangular box and its width and height.

For example, a typical regional proposal network (RPN) incorporates a dichotomous classification problem, that is, whether the object in the location is an object or not. Rectangular boxes of different sizes and aspect ratios are first generated on a sliding window and labeled positively or negatively. The sample data for the RPN is organized as a binary classification labeling problem with multiple rectangular boxes within the input image and the presence or absence of objects within each rectangular box. The RPN maps each sample to a probability value and four localization values. The probability value reflects the probability of having an object in a rectangular box, and the four localization values are used to regress the center horizontal and vertical coordinates and the width and height of the target object location.

3.1.4 Detection

The image detection problem is equivalent to a combination of localization and classification problems. It needs to locate the location of the target and to deduce the class of the located target.

Current mainstream target detection algorithms can be divided into two-stage and one-stage detection. The former frames detection as a “coarse to fine” process, while the latter defines it as a “one-step completion”

The two-stage detection algorithm is relatively slower but more effective. Take the most typical two-stage target detection algorithm, faster RCNN, as an example. As shown in Fig. 24a, the first stage locates the target location in the input image through the region suggestion network. Then in the second stage, the localized targets are classified; finally, the rectangular box of the localized targets and their categories are obtained. There is a certain sequence between the two stages of this model, which makes a stronger connection between localization and classification. This approach allows for more robust results. The single-stage detection algorithm is fast but less accurate. Taking the most typical two-stage target detection algorithm YOLO v3 as an example, YOLO directly generates both coordinates and probabilities for each category at a time using regression, as shown in Fig. 24b. This improves the

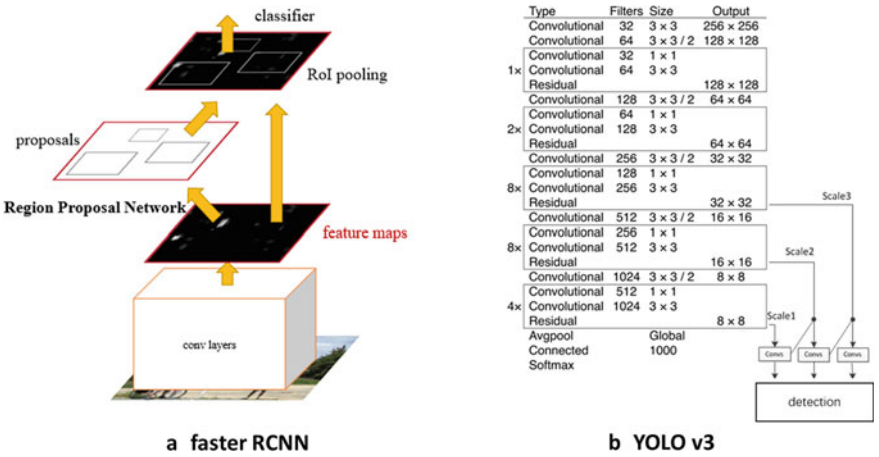

Fig. 24 Two representative detection framework architecture diagrams. **a** Faster RCNN, most typical two-stage target detection algorithm. **b** YOLO v3, an one-stage target detection algorithm

computational speed of the model but leads to poorer results because of the lack of correlation between localization and classification.

There are two main measures for the target detection task: IoU and confidence. The IoU was introduced in the previous section, and the other metric, confidence, is used to measure the confidence level of the detected targets. The higher the confidence level, the more certain the model is about the output.

Object detection has both classification and localization capabilities and has numerous applications. Examples include face detection, text detection, remote sensing target detection, pedestrian detection, and automatic detection of traffic signs and traffic signals. Target detection is widely used in military investigations, disaster rescue, and urban traffic management, and has gained wide attention in the fields of automatic driving, video surveillance, and criminal investigation, etc.

3.2 Segmentation Based on Neural Network

Neural network-based segmentation is most often found in vision tasks. The following section introduces image segmentation as an example.

Image segmentation is different from image classification and monitoring. The task of image classification is to identify the content of an image, whereas the task of image monitoring is to identify the content of an image and also monitor its location. Image segmentation is a pixel-level image classification task based on classifying each pixel of an image.

3.2.1 Problem Description

Image segmentation is a key operation in image processing. It refers to the use of several disjoint regions to represent a complete image, based on the references of grayscale, luminance, texture, and other characteristics of the image. It can simplify the representation of the image. Features show similarity or consistency in the same region, while they show clear differences in different regions.

Existing image segmentation is generally divided into semantic segmentation, instance segmentation, and panoramic segmentation, as shown in Fig. 25.

The semantic segmentation is to give each pixel a class label without distinguishing each instance of the same category. Instance segmentation is a combination of object detection and semantic segmentation. First, it detects the object in the image and then uses semantic segmentation on the detected objects. Moreover, it also distinguishes between different instances of the same kind. Panoramic segmentation is a combination of semantic segmentation and instance segmentation. It gives each pixel a class label while also distinguishing between different instances of the same kind.

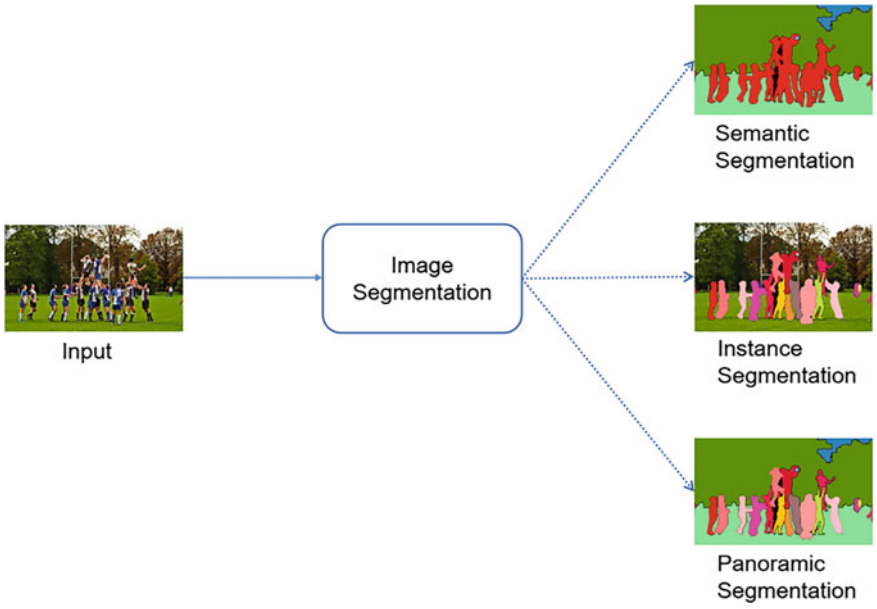

Fig. 25 Division of image segmentation tasks. According to the different segmentation tasks, image segmentation can be divided into semantic segmentation, instance segmentation and panoramic segmentation

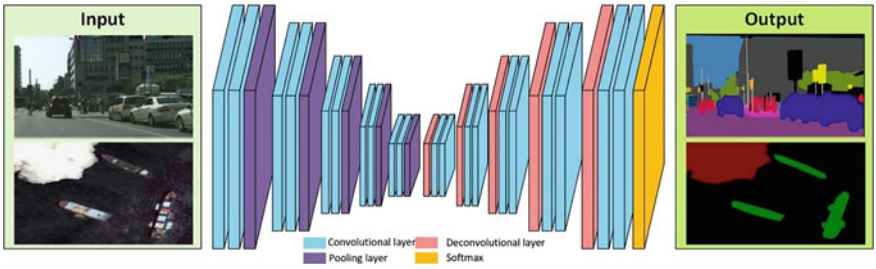

Fig. 26 Example of semantic segmentation based on neural networks. The image above represents an application in city street view. The image below represents an application in a marine environment

3.2.2 Semantic Segmentation

Semantics refers to the contents of an image. Semantic segmentation is performed on a pixel-by-pixel basis to label the class based on the semantics of the image, as shown in Fig. 26. Semantic segmentation uses feature extraction and classification to mark pixels of cars as blue, pixels of trees as green, pixels of buildings as gray, and so on.

There are three algorithm evaluation metrics for neural network-based semantic segmentation: accuracy, execution time, and memory consumption. Among these, execution time is the most important measure. In practical applications, datasets are generally very large, and computer hardware facilities are limited, and only a short execution time can make image segmentation more popular in daily applications. While memory consumption is also an important factor affecting semantic segmentation, memory is expandable in most scenarios. Accuracy is the most critical metric for semantic segmentation. Pixel accuracy and mean IoU are the common forms of accuracy. The number of successfully categorized pixels divided by the total number of pixels is the pixel accuracy. The mean IoU calculates the ratio of the intersection and the union of two sets, and in the field of semantic segmentation, the true and predicted values are the embodiment of the two sets.

The initial neural network-based semantic segmentation models are AlexNet, VGGNet, and ResNet. The emergence of fully convolutional networks later broke the previous segmentation method; thus, the accuracy in the PASCSL VOC dataset has substantially improved. FCN has greatly promoted the development of semantic segmentation algorithms. SegNet [1], RefineNet [32], PSPNet [55], and DeepLab [6] have been proposed one after another, and all of them have achieved good results.

3.2.3 Instance Segmentation

The network architecture of semantic segmentation aims to optimize the accuracy of segmentation results and improve segmentation efficiency; this should allow for applications in the field of image semantic real-time processing. However, seman-

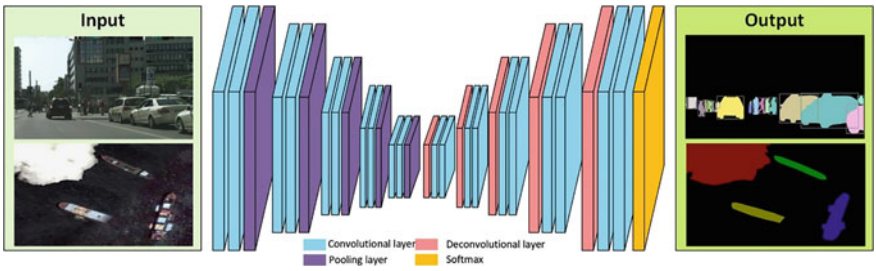

Fig. 27 Example of semantic segmentation based on neural networks. The mask and label of each object in the figure need to be segmented

tic segmentation can only judge categories and cannot distinguish individuals, and it is impossible to accurately understand semantic information or parse scenes in many complex real scenarios. The emergence of an instance segmentation algorithm effectively solves this problem.

The concept of instance segmentation was first proposed by Hariharan et al. [21], who aimed to detect objects in the input image and assign category labels to each pixel of the object. As shown in Fig. 27, unlike semantic segmentation, instance segmentation is able to distinguish between different instances with the same semantic category in the foreground.

Instance segmentation is essentially a combination of semantic segmentation and object detection. It not only has the characteristics of semantic segmentation to classify images at the pixel level but also has the characteristics of object detection to locate different instances of the same category in an image. As shown in Fig. 27, the instance segmentation technique based on deep neural networks usually consists of three parts: the image input, instance segmentation model, and the segmentation result output. First, a deep network model is designed according to the actual requirements, and the original image data is directly input to the network to extract image features. After the high-level abstract features are obtained, the instance segmentation model is used to process. The processing can first determine the location and category of object instances via object detection. It then performs segmentation in the selected region, or it can first implement the semantic segmentation task and then distinguish different instances. The final output is an instance segmented image with the same resolution as the input image.

In recent years, instance segmentation techniques have been rapidly developed. Mask R-CNN [22], which was developed based on the two-stage detector faster RCNN [42], is a direct and effective instance segmentation method. It has become the basic framework for some instance segmentation tasks because of its high accuracy and stability. YOLACT [3], an instance segmentation algorithm extended from a single-stage detector with high-speed detection, achieves real-time segmentation of video information and can obtain efficient processing ability with a small loss of accuracy.

3.2.4 Panoptic Segmentation

Panoramic segmentation combines the tasks of semantic segmentation and instance segmentation to generate a global unified segmented image. Instance segmentation only detects the objects in the image and segments the detected objects. Panoramic segmentation detects and segments all objects in the image including the background, achieving a panoramic understanding of the image. Figure 28 shows the panoramic segmentation. The information predicted by the panoramic segmentation is the most comprehensive, which includes not only the category classification of all pixel points using semantic segmentation but also the function of distinguishing between different instances in the instance segmentation task.

Compared to semantic segmentation, the difficulty of panoramic segmentation is to optimize the design of the fully connected network so that its network structure can distinguish between different categories of instances. The goal of panoramic segmentation is to assign a semantic label and an instance an ID to each pixel in the image, where the semantic label refers to the category of the object and the instance ID corresponds to different numbers of similar objects. Therefore, the overlapping phenomenon in instance segmentation cannot occur in panoramic segmentation.

The basic process of panoramic segmentation is shown in Fig. 29; it is mainly divided into feature extraction, semantic segmentation and instance segmentation processing, and sub-task fusion. The purpose of feature extraction is to obtain the feature representation of the input image and provide the necessary information for the two subsequent tasks. It relies on deep neural networks, and the main networks used include VGGNet, ResNet, MobileNet. The extracted features are shared by semantic

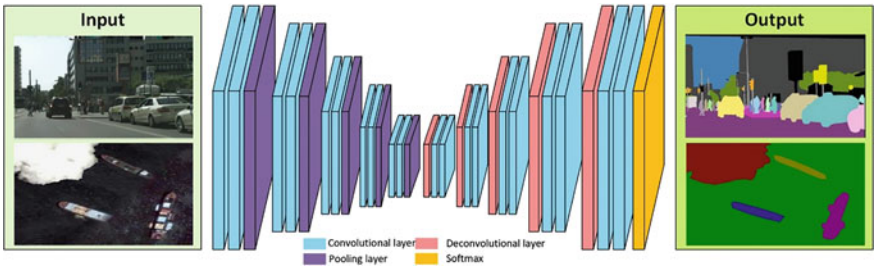

Fig. 28 Example of panoramic segmentation based on neural networks. Panoramic segmentation task is a combination of semantic segmentation task and instance segmentation task

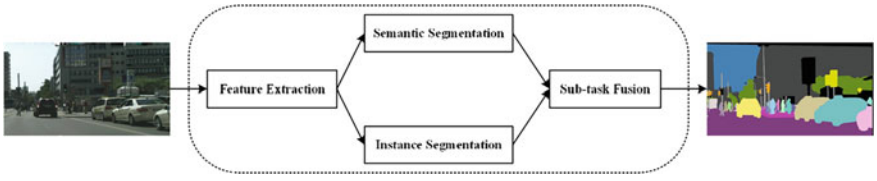

Fig. 29 The processing flow of panoramic segmentation

segmentation and instance segmentation. The semantic segmentation branch produces semantic segmentation predictions, while the instance segmentation branch produces instance segmentation predictions. The subtask fusion processes and fuses the prediction results of the above two branches in an appropriate way to produce the final panoramic prediction. Many works have been done to adopt the above basic process, such as JSIS-Net [9], AU-Net [31], Single network [10], OANet [34], etc.

Image segmentation can convert images into more meaningful and analyzable content expressions, which can effectively improve the processing efficiency of subsequent vision tasks. It is the basis of the computer vision scene being able to understand images and plays an important role in many scenes. In the field of medical image processing, by processing CT images of the organs of patients, it can accurately locate the boundaries of lesions; it can also automatically determine the location, shape, and size of the diseases, and assist doctors in lesion detection. In the field of remote sensing image processing, it can efficiently survey and plan the geographic spatial information such as topography and landform, water pattern direction, urban distribution, and farming planning. In the field of automatic driving, it can judge the surrounding environment of the road based on real-time road scenes, including lane line direction, traffic signs, and safety position of oncoming pedestrians or vehicles; it aims to provide correct guidance to vehicles and ensure driving safety. In the field of intelligent security, the object in the surveillance video is located and screened; it aims to play the role of security warning or object tracking. Furthermore, image segmentation can also be applied to augmented reality, text extraction, and industrial sorting, etc.

With the improvement of computer performance and the continuous optimization of image segmentation algorithm architecture, image segmentation technology based on deep neural networks has become an important task. While pushing the network in the direction of being lightweight, real-time, and highly accurate, more attention should be paid to technology implementation and scene promotion.

3.3 Prediction Based on Neural Network

This section introduces the application of neural networks to prediction problems. We will expand on regression, time-series prediction, one-dimensional signal prediction, and two-dimensional video prediction.

3.3.1 Regression

The link between the independent and dependent variables is predicted using regression [8]. The learning of the regression issue is similar to that of function fitting: select a function curve that fits the known data and accurately predicts the unknown data.

According to the number of independent variables, regression problems are split into unary and multiple regressions. It is separated into linear and nonlinear regression based on the relationship between the independent and dependent variables.

The square loss function is the most often used loss function for regression learning. The least-squares approach may be used to tackle the regression problem in this scenario.

Tasks in many fields can be formalized as regression problems. For example, regression can be used in the business field as a tool for market trend forecasting, product quality management, customer satisfaction surveys, and investment risk analysis.

3.3.2 Time Series Prediction

A time series is a sequence of numbers arranged in a specific order, and this order is usually determined by time [53]. It is an important means for people to understand the objective world and natural phenomena.

The development of time-series prediction is divided into two periods. The early-stage was before World War II, and financial and economic forecasting was the key. The second stage was from the mid-war to the 21st century. In this period, the application areas were more extensive; these include meteorology, aerospace, electronic computers, and mechanical vibration, etc. Time-series prediction has become a hot field pursued by experts in academic research.

The rapid development of artificial intelligence has a significant impact on time-series prediction methods. Currently, commonly used time-series prediction methods can be divided into traditional methods and methods based on deep learning.

Traditional time-series prediction methods are usually not ideal for non-wide stationary time series. Moreover, they are limited in forecasting by complex and highly nonlinear time series.

The emergence of neural networks has solved these problems. Neural networks have good learning capabilities. They can learn the underlying laws of the time series through multiple iterations based on the data itself. Compared to traditional methods, neural network methods are more accurate and can be applied to most time series. Among them, RNNs usually perform better when dealing with time-series problems.

3.3.3 One-dimensional Signal Prediction

Neural networks are also widely used in the prediction of one-dimensional signals.

In recent years, some scholars have discovered that high-frequency oscillation signals have a certain correlation with particular diseases, which may help to improve the accuracy of lesion location and promote the success rate of clinical operations. On the other hand, these findings and applications can help us understand the pathophysiological mechanism of human brain electrical activity and explore preventive treatments that predict disease.

3.3.4 Two-dimensional Video Prediction

Video prediction technology predicts subsequent video frames when several lengths of continuous video frames are provided [55]. It is an important topic in the field of computer vision and has significant application prospects.

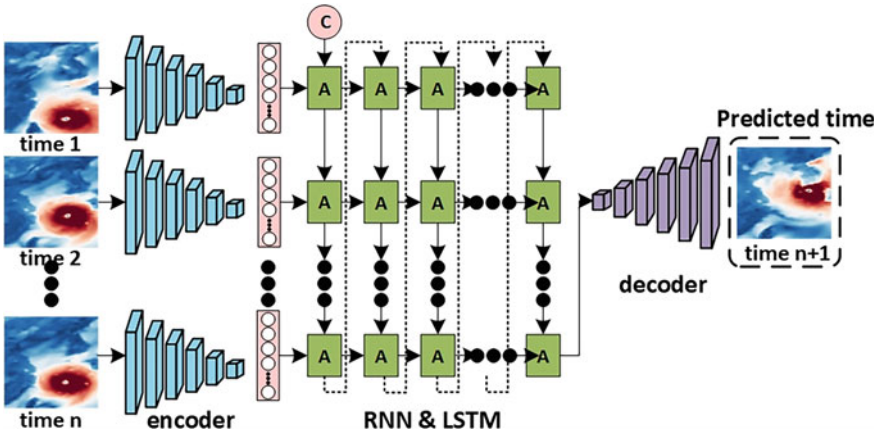

Fig. 30 Two-dimensional time series prediction. The typhoon position at time-1 to time-n are known, and the typhoon position at time n +1 is predicted

For example, in unmanned driving tasks, researchers can use the image information of historical frames to analyze the trajectory information of pedestrians and vehicles outside the car. In this way, the computer can predict the location of the objects outside the vehicle and make judgments in advance. Traffic accidents can also be avoided, and the safety of unmanned driving can also be improved.

In oceanography, the forecast of meteorological elements and ocean elements in a certain sea area is similar to the forecast of video sequences. The gridded two-dimensional sea area corresponds to a frame in the video, and the change of the sea area in a certain period corresponds to the change of the video frame on the timeline. As shown in Fig. 30, the figure depicts the neural network prediction of a typical typhoon path in the seas of eastern China. The neural network encodes the offshore wind current field in the eastern China sea at each known moment and predicts the process of the wind current field in the future through the RNN.

References

1. Badrinarayanan V, Kendall A, Cipolla R (2017) SegNet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495

2. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284(5):34–43
3. Bolya D, Zhou C, Xiao F, Lee YJ (2019) Yolact: real-time instance segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9157–9166
4. Buchanan B, Sutherland G (1968) Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry. Technical report, Stanford University California Department of Computer Science
5. Cauchy A (1847) Méthode générale pour la résolution des systemes d'équations simultanées. Comp Rend Sci Paris 25(1847):536–538
6. Chen LC, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 801–818
7. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078
8. Cox DR (1972) Regression models and life-tables. J R Stat Soc: Ser B (Methodological) 34(2):187–202
9. De Geus D, Meletis P, Dubbelman G (2018) Panoptic segmentation with a joint semantic and instance segmentation network. arXiv:1809.02110
10. de Geus D, Meletis P, Dubbelman G (2019) Single network panoptic segmentation for street scene understanding. In: 2019 IEEE intelligent vehicles symposium (IV). IEEE, pp 709–715
11. Duan Y, Chen X, Houthooft R, Schulman J, Abbeel P (2016) Benchmarking deep reinforcement learning for continuous control. In: International conference on machine learning, PMLR, pp 1329–1338
12. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
13. Fukushima K, Miyake S (1982) Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. Springer, pp 267–285
14. Gaschnig J (1982) Prospector: an expert system for mineral exploration. In: Introductory readings in expert systems. Gordon and Breach Science Publishers, New York
15. Gers FA, Schmidhuber J (2000) Recurrent nets that time and count. In: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks. IJCNN 2000. Neural computing: new challenges and perspectives for the new millennium, vol 3. IEEE, pp 189–194
16. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput. 12(10):2451–2471
17. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144
18. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw 18(5–6):602–610
19. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2008) A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell 31(5):855–868
20. Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 6645–6649
21. Hariharan B, Arbeláez P, Girshick R, Malik J (2014) Simultaneous detection and segmentation. In: European conference on computer vision, Springer, pp 297–312
22. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
23. Hebb DO (2005) The organization of behavior: a neuropsychological theory. Psychology Press
24. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
25. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
26. Hu X, Zhang J, Li J, Zhang B (2014) Sparsity-regularized HMAX for visual recognition. PloS One 9(1):e81813

27. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
28. Janocha K, Czarnecki WM (2017) On loss functions for deep neural networks in classification. arXiv:1702.05659
29. Jordan MI (1997) Serial order: a parallel distributed processing approach. In: Advances in psychology, vol 121. Elsevier, pp 471–495
30. Li X, Zhang Y, Zhang J, Chen Y, Li H, Marsic I, Burd RS (2017) Region-based activity recognition using conditional GAN. In: Proceedings of the 25th ACM international conference on Multimedia, pp 1059–1067
31. Li Y, Chen X, Zhu Z, Xie L, Huang G, Du D, Wang X (2019) Attention-guided unified network for panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7026–7035
32. Lin G, Milan A, Shen C, Reid I (2017) RefineNet: multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1925–1934
33. Lipton ZC, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv:1506.00019
34. Liu H, Peng C, Yu C, Wang J, Liu X, Yu G, Jiang W (2019) An end-to-end network for panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6172–6181
35. Martin WA, Fateman RJ (1971) The MACSYMA system. In: Proceedings of the second ACM symposium on symbolic and algebraic manipulation, pp 59–75
36. McCarthy J (1959) Lisp: a programming system for symbolic manipulations. In: Preprints of papers presented at the 14th national meeting of the Association for Computing Machinery, pp 1–4
37. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
38. Minsky M, Papert SA (2017) Perceptrons: an introduction to computational geometry. MIT press
39. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
40. Newell A, Shaw JC (1957) Programming the logic theory machine. In: Papers presented at the 26–28 Feb 1957, western joint computer conference: techniques for reliability, pp 230–240
41. Newell A, Simon H (1956) The logic theory machine-A complex information processing system. IRE Trans Inf Theory 2(3):61–79
42. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99
43. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
44. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv:1707.06347
45. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
46. Shortliffe EH (1974) MYCIN: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Technical report, Stanford University of California Department of Computer Science
47. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
48. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A (2017) Mastering the game of go without human knowledge. Nature 550(7676):354–359

49. Turing AM (2009) Computing machinery and intelligence. In: Parsing the turing test. Springer, pp 23–65
50. Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575(7782):350–354
51. Viola J, Chen Y, Wang J (2021) FaultFace: deep convolutional generative adversarial network (DCGAN) based ball-bearing failure detection method. Inf Sci 542:195–211
52. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using dropconnect. In: International conference on machine learning, PMLR, pp 1058–1066
53. Weigend AS (2018) Time series prediction: forecasting the future and understanding the past. Routledge
54. Xingjian SHI, Chen Z, Wang H, Yeung DY, Wong WK, Woo Wc (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810
55. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2881–2890
56. Zhu J, Hang SU, Zhang B (2020) Toward the third generation of artificial intelligence. Scientia Sinica Informationis 50(9):1281. https://doi.org/10.1360/SSI-2020-0204