With the continuous development of artificial intelligence, mankind faces not only technical difficulties, but also many future uncertainties.

The main idea at 90s was to create an expert system to assist software engineers during software development [24]. In [23], this proposal is called the programmer's Apprentice Project. The purpose of this project is to achieve human-computer interaction, which is similar to the relationship between programmers and assistants in reality. To achieve the purpose of increasing productivity. At first, this expert system could only implement the simplest directional programming. With training, the expert system can implement more complex programming, but the key logic processing still needs to be processed by human programmers. Immediately afterwards, the expert put forward an idea. The programmer provided a logical idea, allowing the expert system to focus on generating functions, data structures, and even the entire program [8]. At this stage, the problems that artificial intelligence can handle in software engineering are [23]:

1. Disambiguating natural language requirements.
2. Developing knowledge-based systems and ontologies to manage the requirements and model problem domains.
3. The use of computational intelligence to solve the problems of incompleteness and prioritization of requirements.

For this part of the expert system, I personally think that the biggest challenge is not only the problem of the machine, but also the influence of human factors. The amount of data is too small, the computing power of the computer is low, and the algorithm is insufficient. The artificial intelligence in the early stage is not ‘smart’. Then scientists proposed an algorithm named machine learning, especially deep learning is the most famous.

Deep learning (deep structured learning) is a method of machine learning, which includes supervised, semi-supervised, and unsupervised [是在represent learning中 ]. According to this paper, the biggest challenges of deep learning at present are [1]:

1. A large amount of labeled data is required for deep learning to achieve a large amount of labeled data, which makes researchers in the field of computer vision tend to do research in fields with rich data resources , Instead of doing research in important areas. Although there are some methods that can reduce the dependence on data, such as transfer learning, few sample learning, unsupervised learning and weakly supervised learning. But so far, their performance has not been comparable to supervised learning.
2. Overfitting the benchmark data. Deep neural networks perform well on benchmark data sets, but on real-world images outside the data sets, the effect is not satisfactory.
3. Excessively sensitive to image changes, deep neural networks are very sensitive to standard adversarial attacks. These attacks can cause changes in images that are imperceptible to humans but may change the neural network's perception of an object. Moreover, the neural network is too sensitive to changes in the scene.

For deep neural networks to handle all the problems, it seems that an infinite data set is needed, which brings huge challenges to training and testing data sets.

So, what impact do these have on software engineering? I think it can be considered from the software life cycle. Then the conclusion is that artificial intelligence can automate requirements analysis, code analysis, coding, software testing, and fault diagnosis. As I mention before, in artificial intelligence, uncertainty appears on the human side. Specifically, my understanding is the uncertainty of knowledge and the uncertainty of software development. The uncertainty of knowledge is mainly reflected in the uncertainty of common-sense knowledge and language. Among them, common sense knowledge can be expressed in natural language, and the corresponding concepts show obvious uncertainties such as vagueness and randomness. The uncertainty of knowledge is bound to cause uncertainty in artificial intelligence. There is also uncertainty about software quality. The mainstream of computer software engineering is object-oriented technology and methods, which mainly include object design, object analysis, and object realization. However, with the continuous improvement of software complexity and the continuous increase of software scale, the quality of software products has gradually become more difficult to control and grasp. The implementation process of each sub-project can be regarded as a process of human-computer interaction. In the human-computer interaction, the human factor is a particularly important influencing factor. The reliability of the actual software is mainly affected by the operator. Therefore, until now, artificial intelligence has not been able to play its maximum role in software engineering.

[1] https://arxiv.org/abs/1805.04025

[8] Hewett, Micheal, and Rattikorn Hewett (1994). 1994 IEEE 10th Conference on Artificial Intelligence for Applications

[19] Meziane, F. and Vadera, S., (2010). Artificial Intelligence in Software Engineering Current Developments and Future Prospects, In "Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects", IGI Global

[23] Partridge, Derek, ed. (1991). Artificial Intelligence and Software Engineering. New Jersey: University of Exeter, 1991.

[24] Phil B. (1999). The Use of Artificial Intelligence for Program Development, [http://www.philforhumanity.com/The\_Use\_of\_Artificial\_Intelligen ce\_for\_Program\_Development.html](http://www.philforhumanity.com/The_Use_of_Artificial_Intelligen%20ce_for_Program_Development.html)

But AI still has many challenges. For example, labeling data is expensive, large models are inconvenient to use on mobile devices, and large calculations require expensive material and time costs. How to learn effectively from small samples like people? How to expand from cognitive tasks to decision-making tasks? These are all issues to be resolved. But mankind cannot give up the opportunity for development. Humans could create and grow automatically without the need for human engineers to hard code it. As the meta-learning system identifies frequently occurring problem-solving patterns, these patterns will be transformed into reusable subroutines (just like functions and classes in software engineering) and added to the global library. In this way, abstraction and extreme generalization capabilities can be achieved. In the end, AI that can learn from itself first.