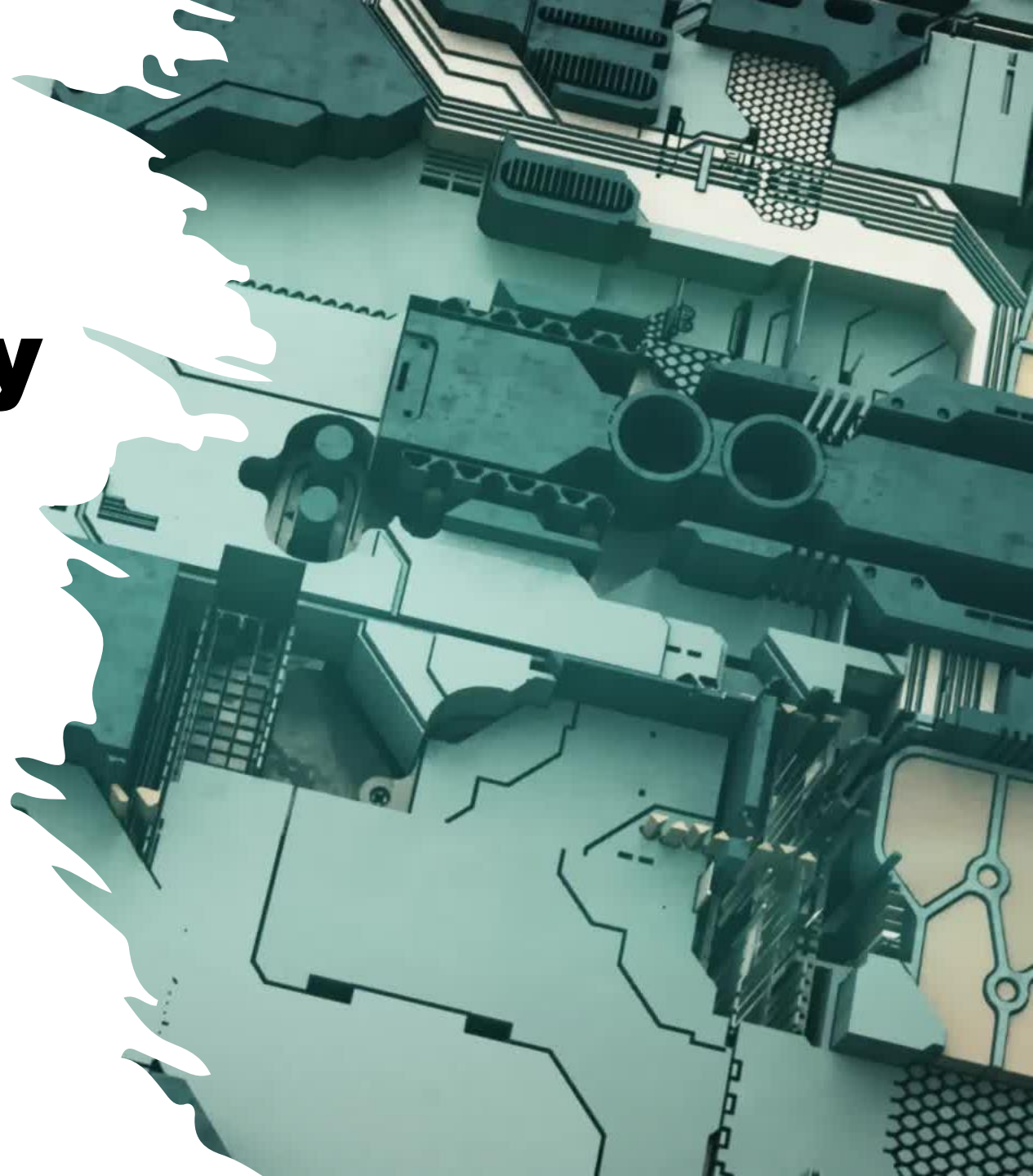
Module 1: Current Computer Security Landscape



Prepared By:

Prof. Sridhar C Iyer

Assistant Professor

Dept. of Computer Engineering

Dwarkadas J Sanghvi College of Engineering

Introduction

Importance of Cybersecurity

Discusses the critical role that cybersecurity plays in safeguarding computer systems, networks, and data from various threats, emphasizing its significance in the modern digital landscape.

Significance of computer and network security in the digital age.

As digital technology becomes increasingly integrated into everyday life, ensuring the security of computer systems and networks becomes more crucial than ever.

Growing Threats

Escalating volume and sophistication of cyber threats.

Indicates that the number of cyber threats is not only increasing but also becoming more sophisticated, posing greater challenges to security professionals.

Targets: individuals, organizations, and governments.

Identifies that cyber threats are not limited to specific entities; individuals, organizations, and even governments are susceptible to various forms of cyber attacks.

Evolution of Cybersecurity

Emergence of cybersecurity specialists.

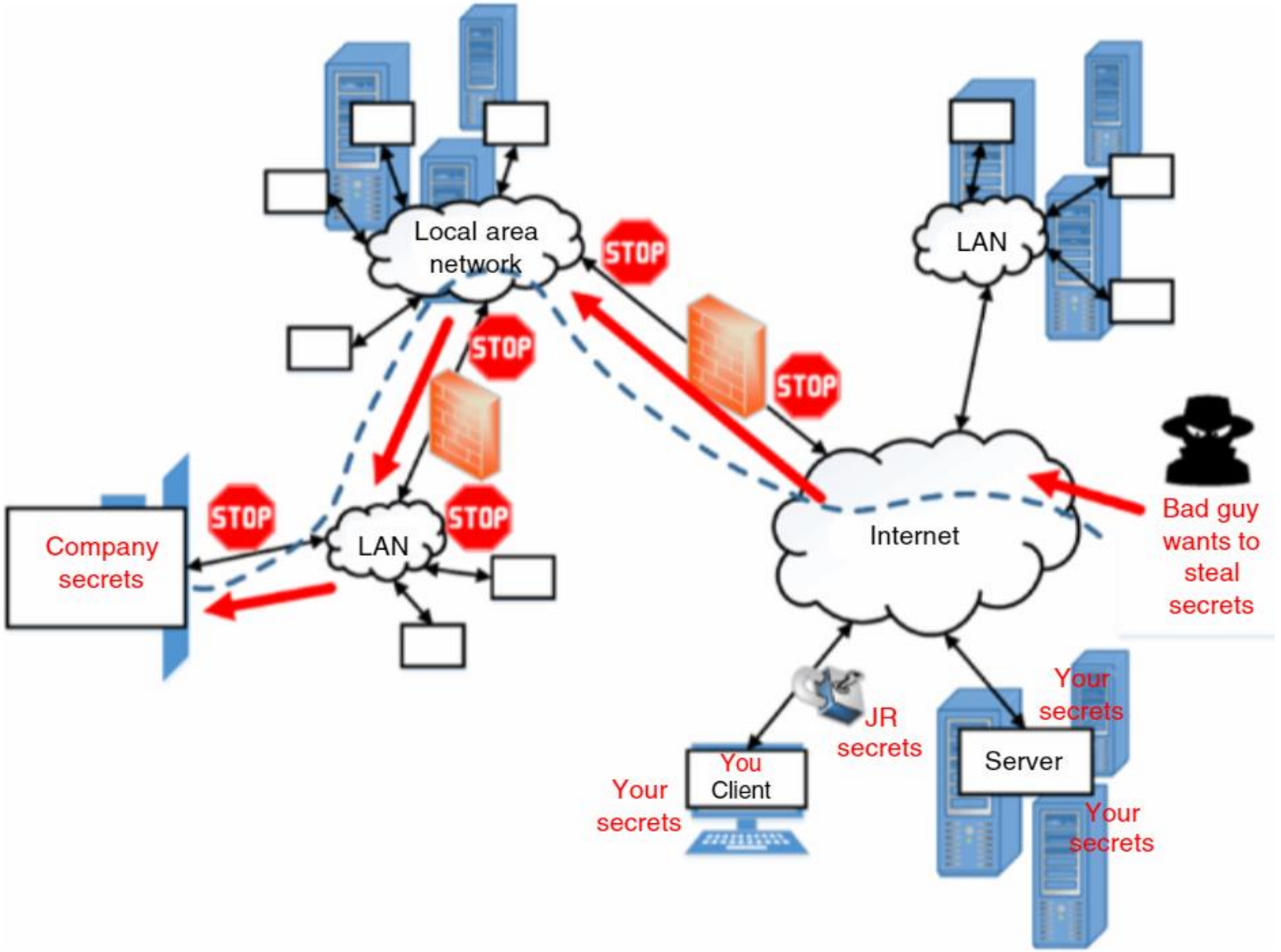
Role: Protecting systems from diverse threats.

Changing Landscape

Impact of computerization, digitization, and Internet interconnection.

Evolution in security mechanisms and defense strategies.

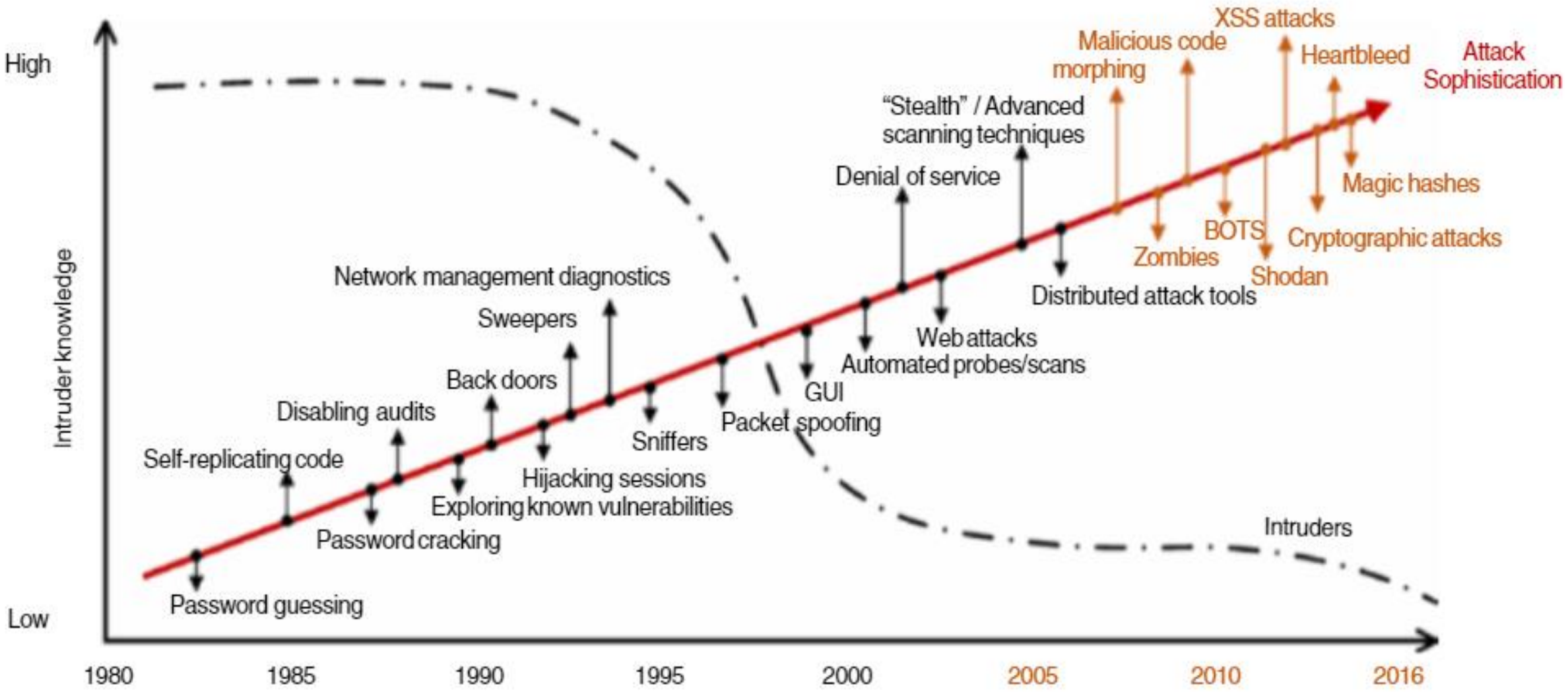
- The proliferation of Internet-connected devices in recent decades has surged.
- Modern products are designed with high-speed Internet capabilities, generating massive amounts of data that require efficient handling.
- Despite the benefits, the evolving landscape has also given rise to more intricate cyber exploits, emphasizing the critical need for robust security measures.



Challenges in Modern Technology

Increasing number of Internet-connected devices.

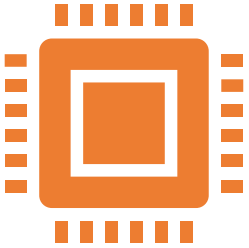
Complexity of cyber threats.


High
Low
Intruder knowledge
1980
1985
1990
1995
2000
2005
2010
2016
Password guessing
Self-replicating code
Password cracking
Disabling audits
Exploring known vulnerabilities
Back doors
Hijacking sessions
Sweepers
Network management diagnostics
Sniffers
Packet spoofing
GUI
Automated probes/scans
Web attacks
Denial of service
"Stealth" / Advanced scanning techniques
Distributed attack tools
Zombies
Malicious code morphing
Shodan
BOTS
XSS attacks
Cryptographic attacks
Heartbleed
Magic hashes
Attack Sophistication
Intruders

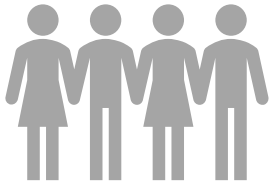
Heartbleed Bug

- The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

Reasons for Growing Cybercrime



Quick adoption of new technologies.



Increased online users from low-income communities.

Financial sophistication of cybercriminals.

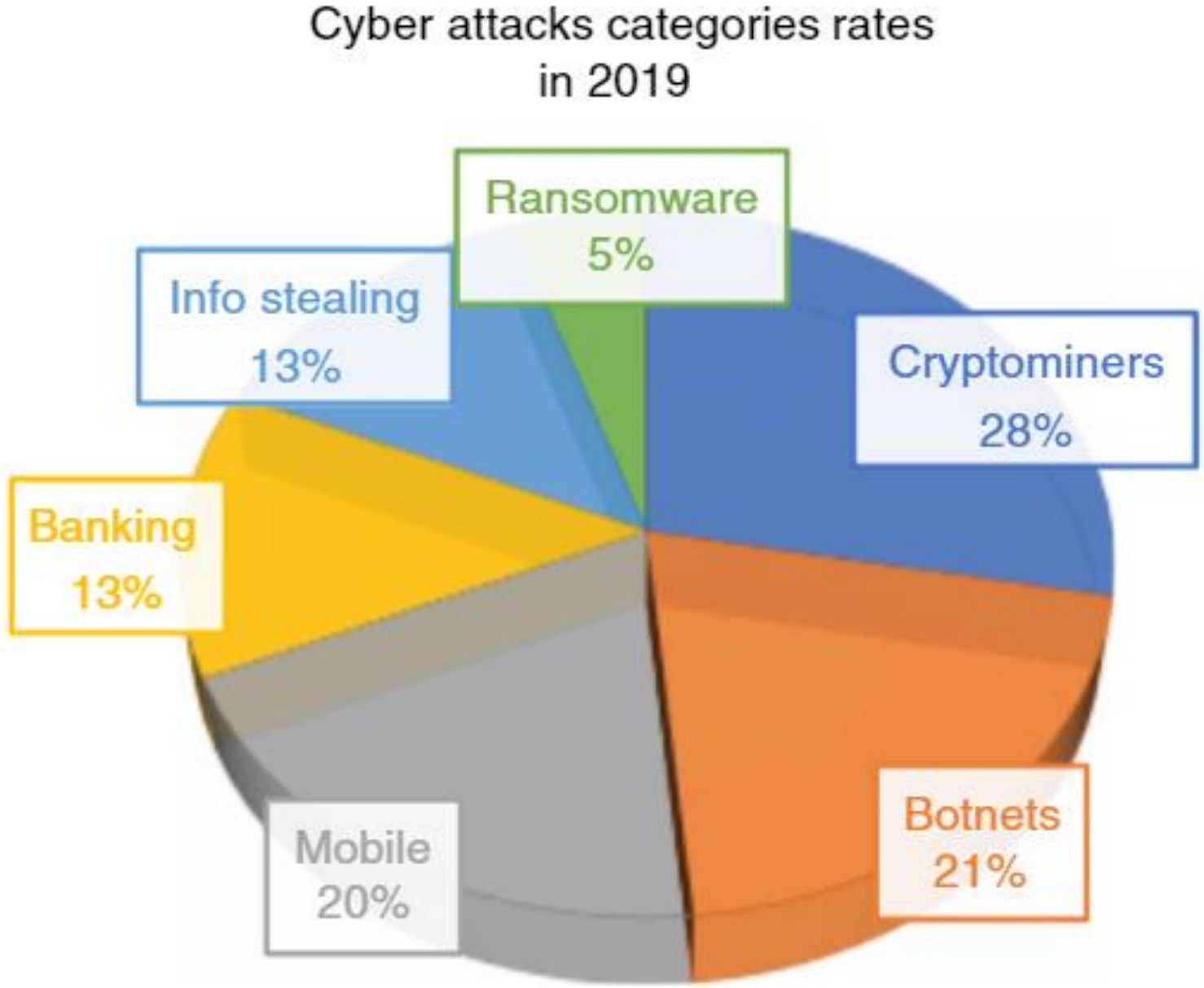
Government Initiatives

- Enhancing resilience against automated, distributed threats.
- Collaborative efforts with international partners.

Some Statistics

Just some recent data. M-trends (2020) reports that in 2019, Mandiant professionals observed that attacks:

- 7% originated with or used compromised third-party access,
- 15% had multiple attackers,
- less than 1% involved an insider,
- 22% had data theft likely in support of intellectual property or espionage end goals,
- 29% were likely for direct financial gain including extortion, ransom, card theft, and illicit transfers,
- 3% of these targeted attacks were for the purpose of reselling access gained in the intrusion,
- 4% likely served no purpose except for creating compromised architecture to further other attacks.


Cyber attacks categories rates
in 2019
Ransomware
5%
Info stealing
13%
Cryptominers
28%
Banking
13%
Botnets
21%
Mobile
20%

Computer Security Basics

Fundamental Concepts: Confidentiality, Integrity, Availability (CIA).

Protection goals against cyber attacks.

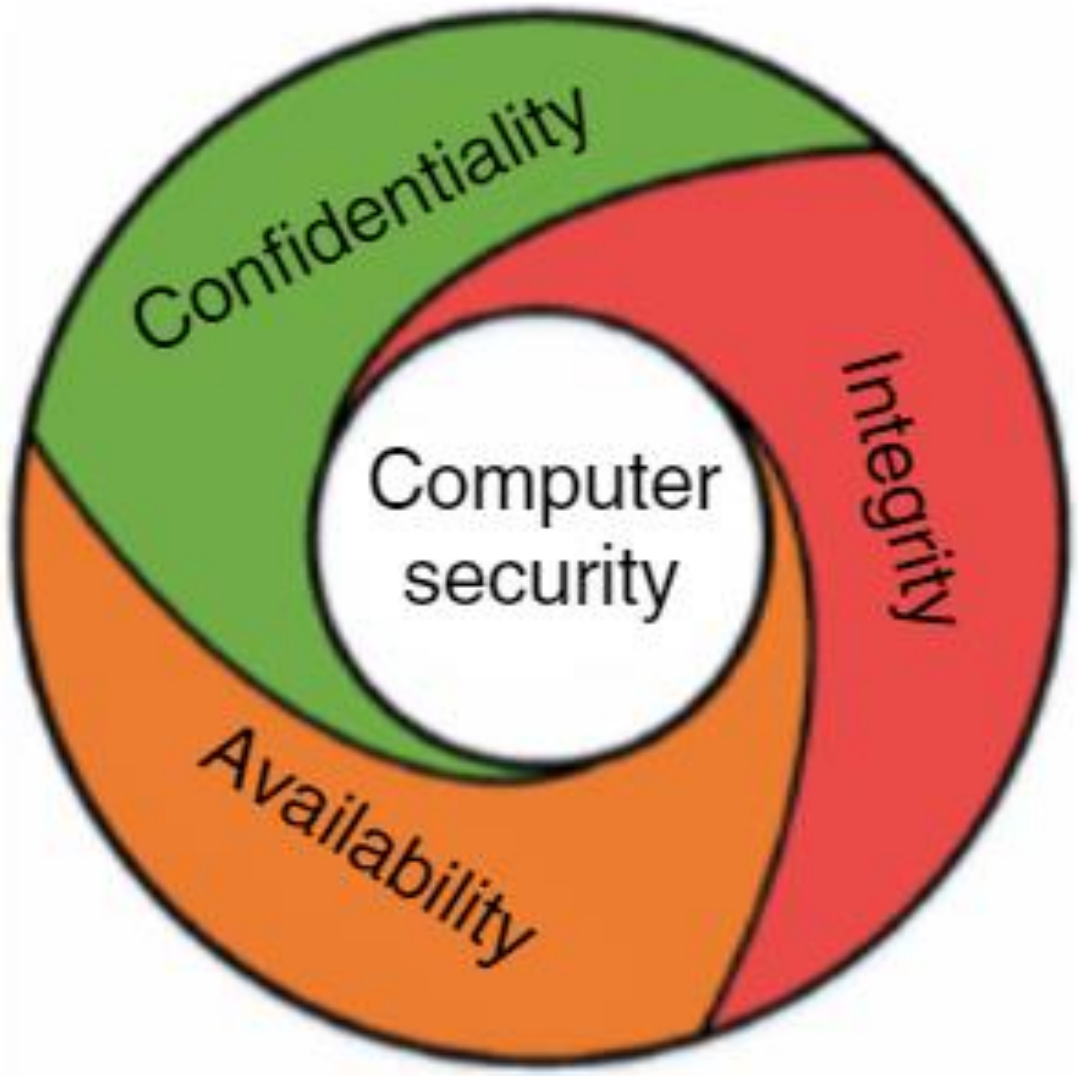


Table 1.1 Possible security threats against assets.

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.		Parts are replaced with no authorization.
Software	Code, programs are removed.	An unauthorized copy of software is made and could be executed.	A code is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted or hidden, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or replace with new files.
Communication channels	Messages are destroyed or deleted. Communications lines or networks are rendered unavailable.	Messages are copied and/or read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Attack in Computer Security

Definition of an attack in computer security.

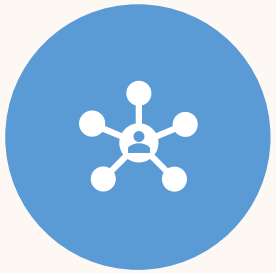
Defensive goals to protect against attacks.

Sources of Security Threats

WEAKNESSES IN INFRASTRUCTURE AND PROTOCOLS.

DESIGN AND IMPLEMENTATION ISSUES.

VULNERABILITIES IN SOFTWARE.



HACKER ACTIVITIES AND SOCIAL ENGINEERING.

Some Case Studies

Example 1.1 Target Data Breach

In 2013, Target Corporation's security and payment system was breached, compromising 40 million credit and debit card numbers, along with 70 million addresses, phone numbers, and other personal information. The State's investigation of the breach determined that cyberattackers gained access to Target's computer gateway servers by using credentials stolen from a third-party vendor in November 2013. With the credentials, the attackers gained access to a customer service database, installed malware on the system, and captured personal information and other sensitive data. Along with affecting 41 million customer payment card accounts, the breach affected contact information for more than 60 million Target customers. According to Jackson (2014), Target's data breach was a real-world example of a design flaw leading to a hack. The environment was "crunchy on the outside and chewy in the middle." As a result, it was "easy to get to the middle where all the data was stored" once the attackers had compromised the point-of-sale system. Target agreed to pay an $18.5 million multistate settlement (McCoy 2017).

Some Case Studies

Example 1.2 Massive Equifax Breach in 2017

Massive Equifax hack that exposed 143 million American's social security numbers, birth dates, addresses, and drivers' licenses has received much publicity. According to a statement released by Equifax, the breach occurred from mid-May through July 2017. They discovered the breach on July 29, which means attackers were actively working well over a month, if not more, at exhilarating this treasure trove of data. Equifax also stated that criminals exploited a vulnerability in their web application to gain access to sensitive data as the means of compromising their site. Equifax was fast to place the blame on a vulnerability within Apache Struts – a framework for developing Java web applications that Equifax developers employed. The Apache Foundation quickly rebutted this claim. Turns out that it was a Struts vulnerability that allowed hackers in, but one that was already disclosed and saw a patch issued in March. The blame still lies at Equifax's door for failure to plug a gaping hole discovered months ago. Also, the audit found out other security problems in the Equifax systems. For example, the password on Equifax's Argentina employee portal was "Admin" that is considered extremely weak as it could be easily recovered by vocabulary attacks against the passwords.

Some Case Studies

Example 1.3 Ransomware Attack on City of Baltimore

On 7 May 2019, the city of Baltimore discovered that it was a victim of a ransomware attack, in which critical files were encrypted remotely until a ransom would be paid. The city immediately notified the FBI and took systems offline to keep the ransomware from spreading, but not before voice mail, email, a parking fines database, and a system used to pay water bills, property taxes, and vehicle citations were affected and went down. Attackers stated that that the city could unlock the seized files for a price: three Bitcoins per system or 13 Bitcoins for them all. The authorities did not name any individuals or groups behind the attack, but they identified the malicious software, or malware, that was employed as "RobbinHood," a relatively new ransomware variant. The creators of RobbinHood most likely scanned a large number of online systems for vulnerabilities to exploit, such as gaps in protocols used to grant remote access to computers (Chokshi 2019) before finding their target at Baltimore.

Some Case Studies

Example 1.4 Ransomware Activity Targeting the Healthcare and Public Health Sector (Alert 2020)

Alert (AA20-302A) joint cybersecurity advisory was coauthored by the Cybersecurity and Infrastructure Security Agency (CISA), the Federal Bureau of Investigation (FBI), and the Department of Health and Human Services (HHS). This advisory describes the tactics, techniques, and procedures (TTPs) used by cybercriminals against targets in the Healthcare and Public Health (HPH) Sector to infect systems with ransomware for financial gain. CISA, FBI, and HHS have credible information of an increased and imminent cybercrime threat to US hospitals and healthcare providers. CISA, FBI, and HHS are sharing this information to provide warning to healthcare providers to ensure that they take timely and reasonable precautions to protect their networks from these threats.

Key Findings:

- CISA, FBI, and HHS assess that malicious cyber actors are targeting the HPH Sector with TrickBot and BazarLoader malware, often leading to ransomware attacks, data theft, and the disruption of healthcare services.
- These issues will be particularly challenging for organizations within the COVID-19 pandemic; therefore, administrators will need to balance this risk when determining their cybersecurity investments.

Physical Security Threats

- Ongoing threat of physical theft.
- Impact on data security, especially in mobile devices and laptops.

Attacks Against IoT and Wireless Sensor Networks

OVERVIEW OF ATTACKS ON IOT DEVICES.

NEED FOR SPECIALIZED SECURITY MEASURES.

Preliminary and Simple Attacks

- Passive Information Gathering.
- Traffic and Node Activity Analysis.
- Device Subversion by Tampering or Destruction.
- Device Malfunctioning.
- Cluster Leader or Aggregate Node Outage.

Active Attacks

- Sensor Stimuli.
- Sybil attack.
- False Node and Byzantine Attack
- Malicious Message Corruption
- Routing Modification and Loops
- Sinkhole/Blackhole or Wormhole
- Denial of Service
- Denial of Sleep
- DOS on Sleep
- Jamming

Sensor Stimuli Attack

The Sensor Stimuli Attack is a type of active attack on wireless sensor networks (WSN) where the adversary targets the sensed readings to cause resource exhaustion in a particular node. Here's a detailed breakdown of this attack:

1. Prerequisite - Subverted Node:

The attacker needs to compromise or subvert a sensor node within the wireless sensor network.

A subverted node implies that the attacker has gained unauthorized access and control over the compromised node.

Sensor Stimuli Attack (cont.)

2. Execution - Resource Exhaustion:

Once a node is compromised, the attacker exploits it to send repeated requests for sensor readings to the network and other nodes.

The objective is to trigger a high volume of requests, causing excessive utilization of resources in the compromised node.

3. Impact - Node Resource Depletion:

The continuous influx of requests for sensor readings consumes computational resources, memory, and energy of the targeted node.

Over time, this leads to resource exhaustion, potentially rendering the node dysfunctional.

Sensor Stimuli Attack (cont.)

4. Countermeasures:

Access Control and Encryption:

- Implement strong access control mechanisms to prevent unauthorized access to sensor nodes.
- Use encryption techniques to secure communication within the wireless sensor network, making it difficult for attackers to interfere with sensor readings.

5. Prevention and Mitigation:

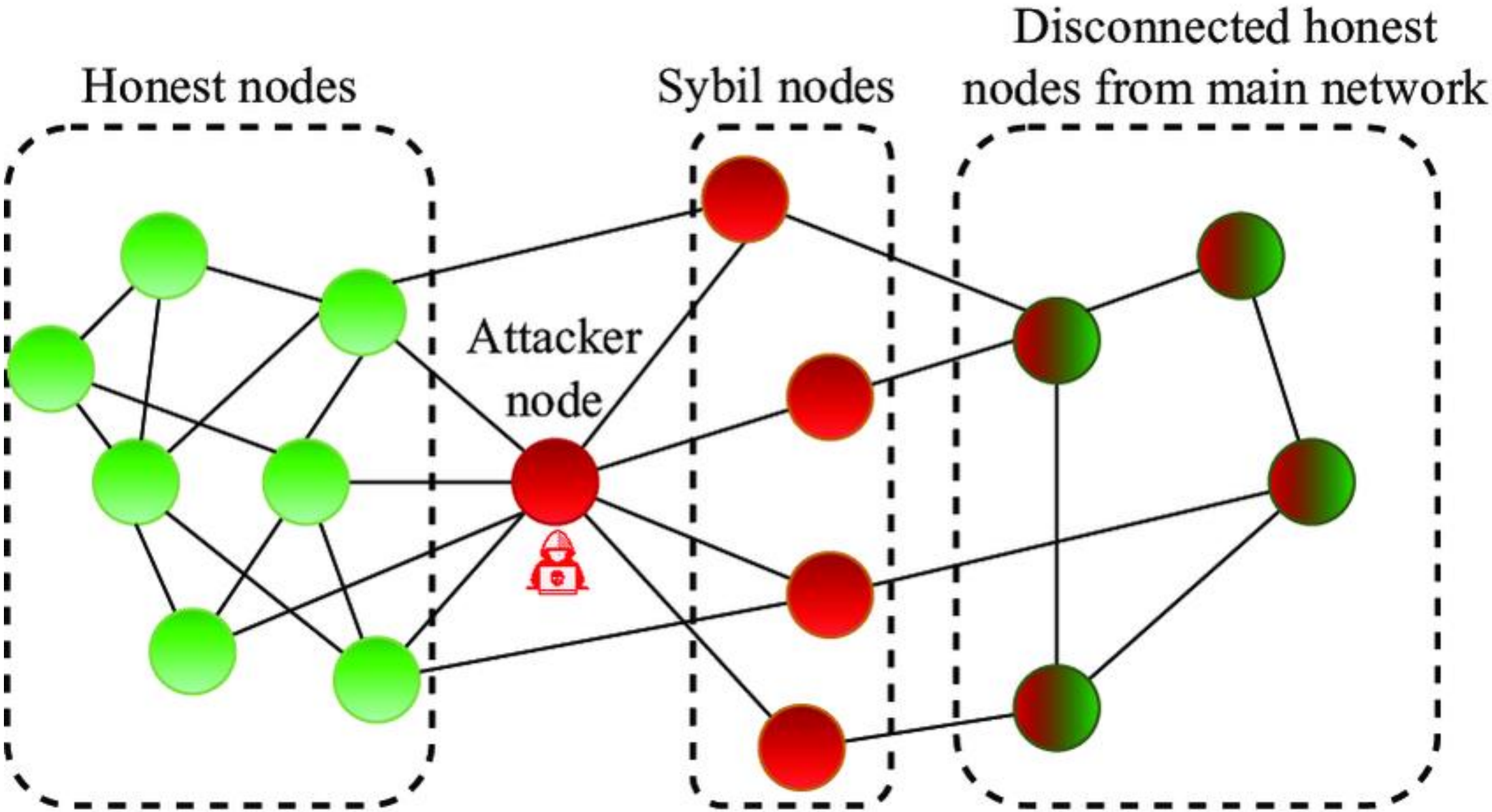
- Regularly monitor network traffic and behavior to detect unusual patterns that may indicate a Sensor Stimuli Attack.
- Employ intrusion detection systems to identify and respond to suspicious activities in real-time.

Sybil Attack

- The Sybil Attack is a sophisticated form of attack on wireless sensor networks (WSN) where the adversary attempts to disrupt communication and compromise the integrity of the network. Here's a detailed breakdown of the Sybil Attack:

1. Execution - Node Cloning:

- The attacker creates multiple illegitimate nodes, known as Sybil nodes, to clone and impersonate genuine nodes within the network.
- Each Sybil node appears as a distinct entity, potentially overwhelming the network with fake identities.


Honest nodes
Sybil nodes
Disconnected honest nodes from main network
Attacker node

Sybil Attack (cont.)

2. Strategy - Routing Table Manipulation:

- The Sybil nodes attempt to manipulate routing information within the network.
- By corrupting or disrupting the routing tables of legitimate nodes, the attacker can control the flow of traffic and cause congestion or disruption.

3. Impact - Congestion and Disruption:

- The compromised routing information leads to congestion and disruption of both control and actual information flow within the network.
- Legitimate nodes may receive false information or be unable to establish proper communication due to the presence of Sybil nodes.

Sybil Attack (cont.)

Prevention and Detection:

- Regularly monitor the network for anomalies in traffic patterns or deviations from expected behavior.
- Employ intrusion detection systems capable of identifying patterns indicative of Sybil Attacks.

Byzantine Attack

A False Node Attack, also known as a Byzantine Attack, is a type of malicious activity in a distributed network where a compromised node intentionally behaves incorrectly or maliciously to disrupt the normal functioning of the network. Here's an in-depth look at the characteristics and implications of a False Node (Byzantine) Attack:

Byzantine Generals Problem:

The term "Byzantine" is derived from the Byzantine Generals Problem, a theoretical scenario where a group of generals must coordinate their actions to attack or retreat but may have traitorous members providing conflicting information.

In the context of distributed networks, a False Node Attack mirrors this problem, with a compromised node disseminating false or conflicting information.

Byzantine Attack (cont.)

Objective - Network Disruption:

The primary objective of a False Node Attack is to disrupt the normal operation of the distributed network.

By spreading misinformation or conflicting messages, the attacker aims to create confusion, compromise consensus, or cause the network to make incorrect decisions.

Multiple Compromised Nodes:

Unlike a Sybil Attack where multiple fake nodes are created, a False Node Attack may involve only one or a few compromised nodes within the network.

These nodes act in concert to deceive the network and compromise its integrity.

Byzantine Attack (cont.)

Countermeasures:

Byzantine Fault Tolerance (BFT):

Implement BFT protocols that enable the network to withstand the malicious behavior of a certain percentage of nodes.

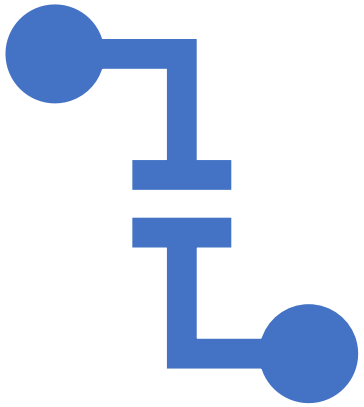
BFT mechanisms involve redundancy and voting schemes to identify and exclude malicious nodes.

Cryptographic Techniques:

Use cryptographic methods to secure communication between nodes and ensure the integrity and authenticity of messages.

Digital signatures and secure communication channels can help in verifying the legitimacy of nodes.

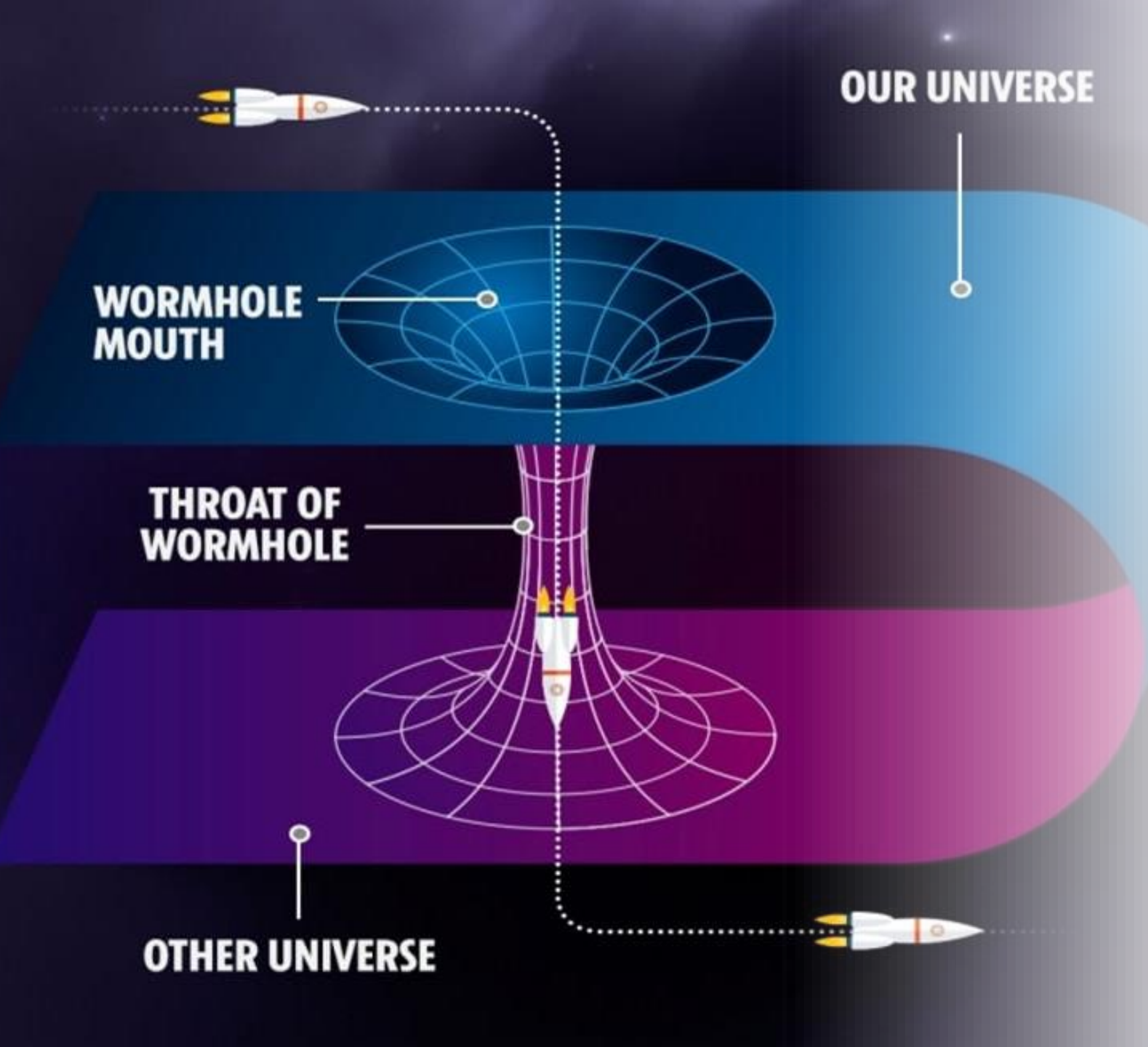
Sinkhole/ Wormhole Attack



- A Sinkhole/Wormhole Attack is a type of security threat in computer networks, particularly in wireless ad-hoc and sensor networks. This attack involves the malicious redirection of network traffic to a centralized point (sinkhole) or a tunnel (wormhole), allowing an attacker to compromise the network's integrity. Here's a detailed explanation of the Sinkhole/Wormhole Attack:

Sinkhole Attack:

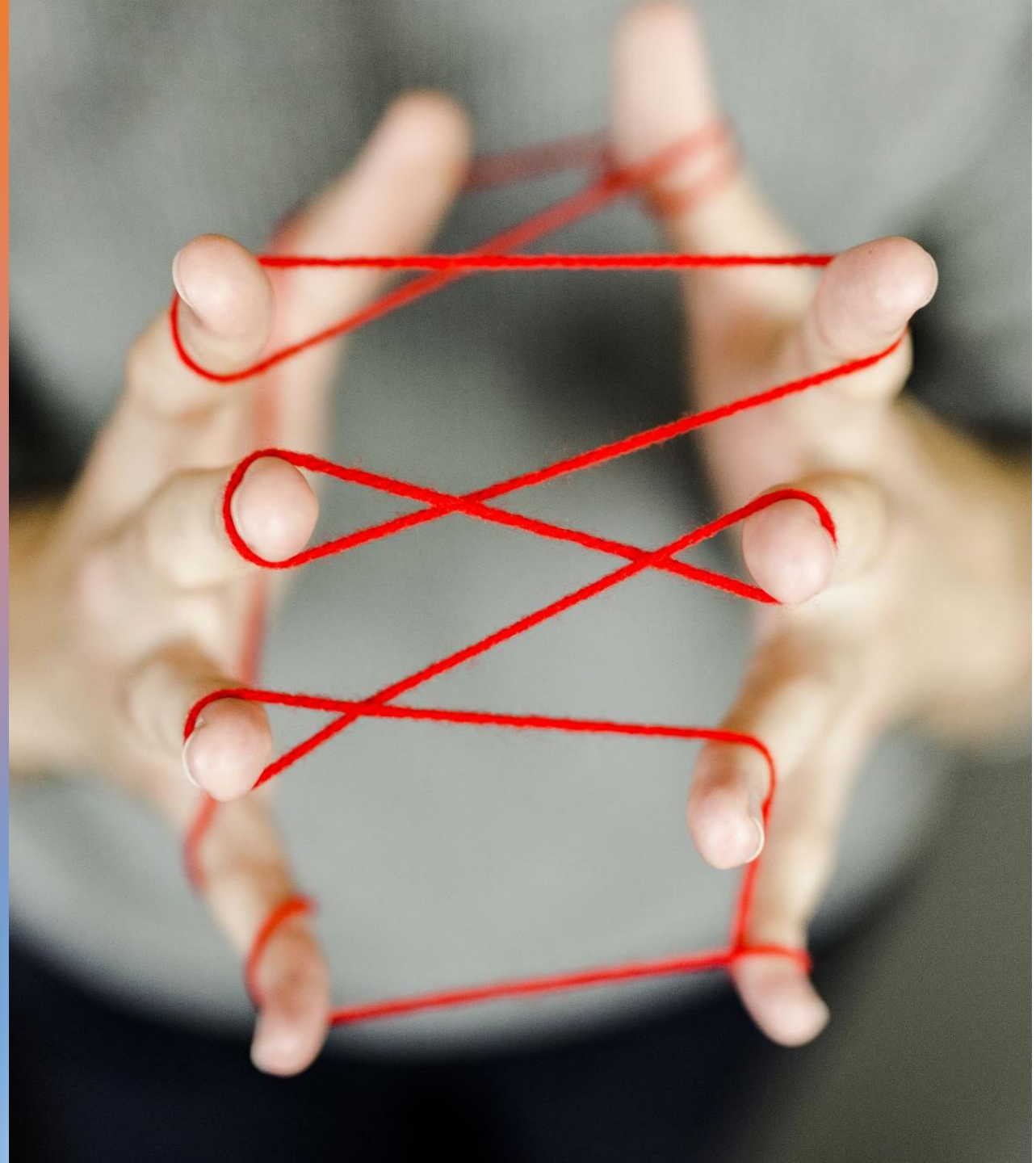
- In a Sinkhole Attack, the attacker attracts and diverts network traffic towards a central point, known as the sinkhole.
- The sinkhole is typically a malicious node or a compromised entity that intercepts and controls the communication flow.



Sinkhole/ Wormhole Attack

Wormhole Attack:

- In a Wormhole Attack, the attacker establishes a covert communication tunnel (wormhole) between two distant points in the network.
- This tunnel allows the attacker to relay packets from one end to the other, making it appear as if the communication is direct, while, in reality, it is being manipulated.



Sinkhole/ Wormhole Attack

Objective - Misdirection and Manipulation:

The primary goal of both Sinkhole and Wormhole Attacks is to misdirect and manipulate network traffic.

By controlling the flow of data, the attacker can eavesdrop on sensitive information, inject malicious packets, or disrupt the normal communication between network nodes.

Sinkhole/Wormhole Attack

Countermeasures:

- **Secure Routing Protocols:**
  - Implement routing protocols that are resistant to attacks and can detect anomalies in the network topology.
  - Secure and authenticated routing can help prevent nodes from being misled by false routing information.
- **Intrusion Detection Systems (IDS):**
  - Deploy IDS to monitor network behavior and detect unusual patterns indicative of a Sinkhole or Wormhole Attack.
  - Anomaly detection techniques can identify deviations from normal communication patterns.
- **Cryptographic Solutions:**
  - Use cryptographic techniques to secure communication channels and data integrity.
  - Authentication mechanisms can ensure that nodes can trust each other, reducing the risk of falling victim to malicious redirection.

Introduction to Artificial Intelligence (AI)

Brief overview of artificial intelligence.

Why AI is needed in Computer Security?

- The volume of available data, the velocity of its generation, the variety of various data sources and the data veracity and quality problem scale move cyber security out of the human abilities range.
- Unfortunately, this discrepancy between cyber security problems, sophistication, volume and variety on one hand and the human operator qualification to address them on another is expected to grow up even much more.
- Analysing and improving cyber security posture is not a human scale problem anymore. Cyber criminals and hackers have realised it and have been already widely utilising intelligent techniques and technologies.
- Cyber security professionals, in order not to lose in the competition, have to employ intelligent technologies too. Fortunately, they have had already gained a lot of experience in design and application of intelligence systems, in enhancing network security, recognising attacks, discovering security vulnerabilities and differentiating legitimate users from hackers.
- In cyber security practice most of the existing tools employ AI in one form or another however those applications remain more or less hidden.

Definition of AI

Various AI definitions do exist. Even ISO (the International Organization for Standardization) attempted to provide 2 definitions as:

1) An interdisciplinary field, usually regarded as a branch of computer science, dealing with models and systems for the performance of functions generally associated with human intelligence, such as reasoning and learning.
2) The capability of a functional unit to perform functions that are generally associated with human intelligence, such as reasoning and learning.

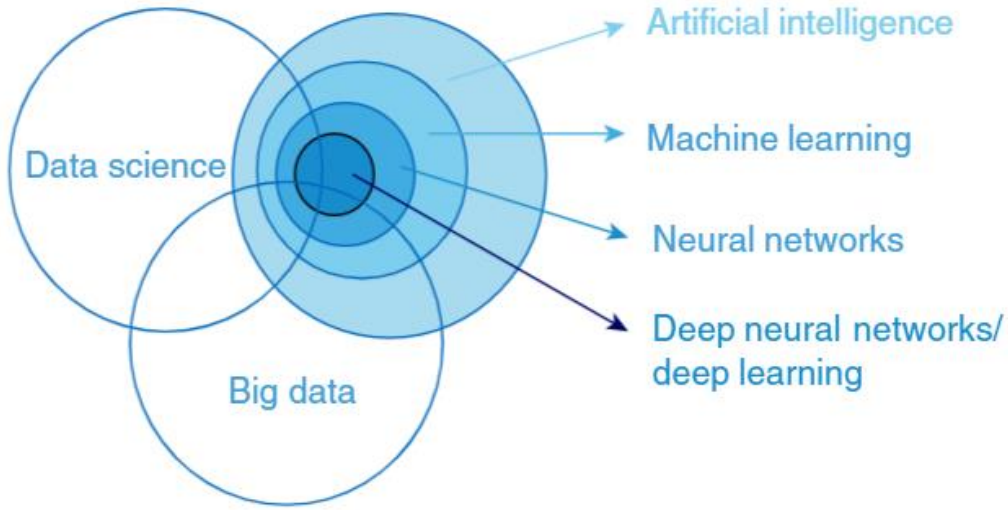


Table 1.2 Comparison between artificial intelligence, machine learning, and data science.

Domain	Artificial intelligence (AI)	Machine learning (ML)	Data science (DS)
Relationship between domains	Includes machine learning	Subset of AI	Intersects with AI and ML
Definition	An interdisciplinary field dealing with models and systems for the performance of functions generally associated with human intelligence, such as reasoning and learning	A subfield of AI that comprises the study of algorithms which have a capability to improve themselves automatically through experience to solve problems without external instructions, by using previously trained models	The field that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data
Major approaches	AI combines large amounts of data through iterative processing and intelligent algorithms to help computers learn automatically	ML uses efficient programs that can use data without being explicitly told to do so	DS works by sourcing, cleaning, and processing data to extract meaning out of it for analytical purposes
Major procedures	Knowledge generation Knowledge processing Learning	Classification Prediction	Data extraction Data visualization Data manipulation Data storage and maintenance
Popular tools	1. TensorFlow 2. ScikitLearn 3. Keras	1. Amazon Lex 2. IBM Watson Studio 3. Microsoft Azure ML Studio	1. SAS 2. Tableau 3. Apache Spark 4. MATLAB
Models and methods used	Artificial intelligence uses logic and decision trees	Machine learning uses intelligent and statistical models.	Data science uses mainly statistical methods to deal with structured and unstructured data.
Popular application examples	Chatbots and automatic assistants	Recommendation systems such as Spotify, and Facial recognition apps	Fraud detection and Healthcare analysis

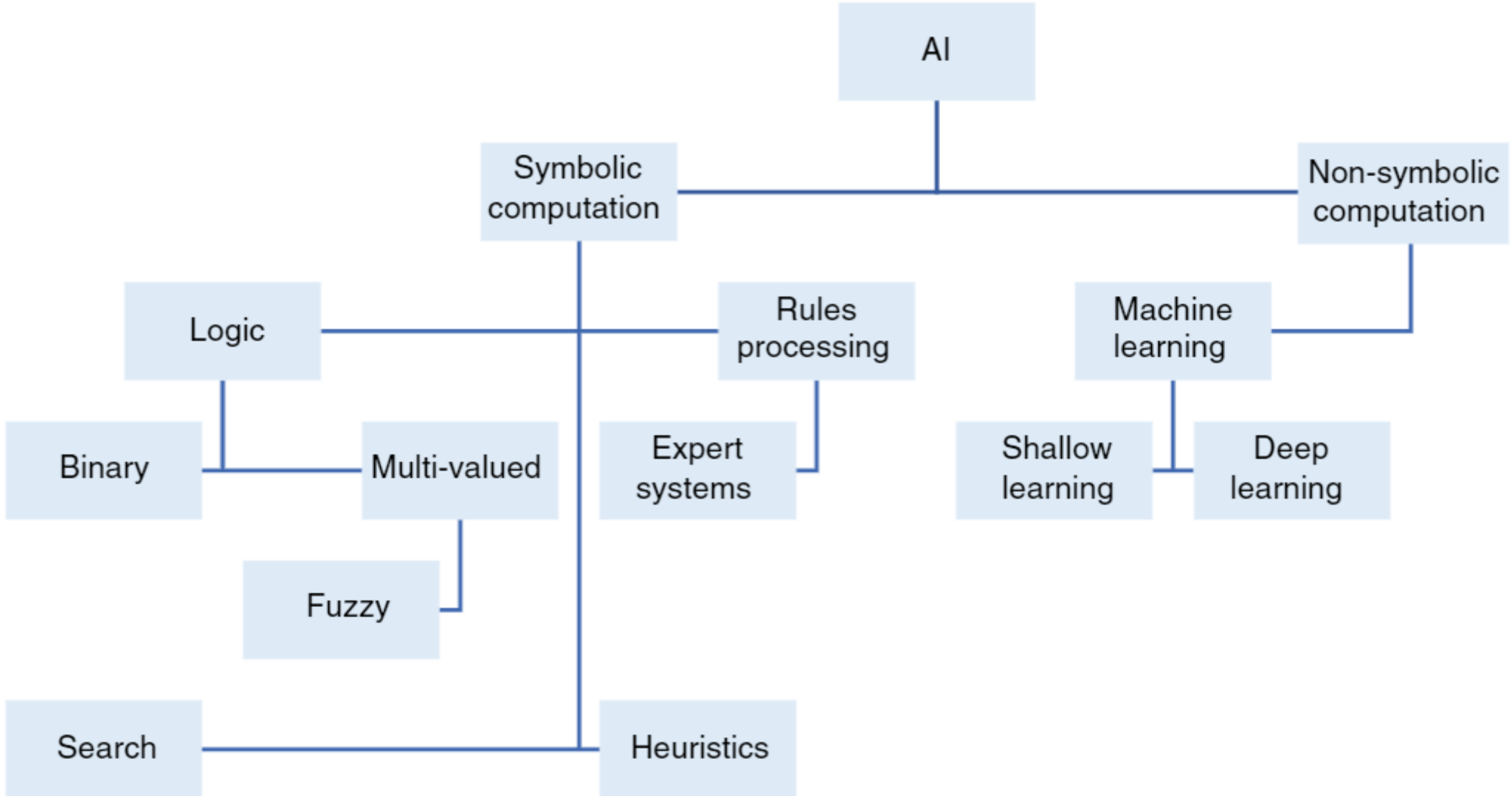
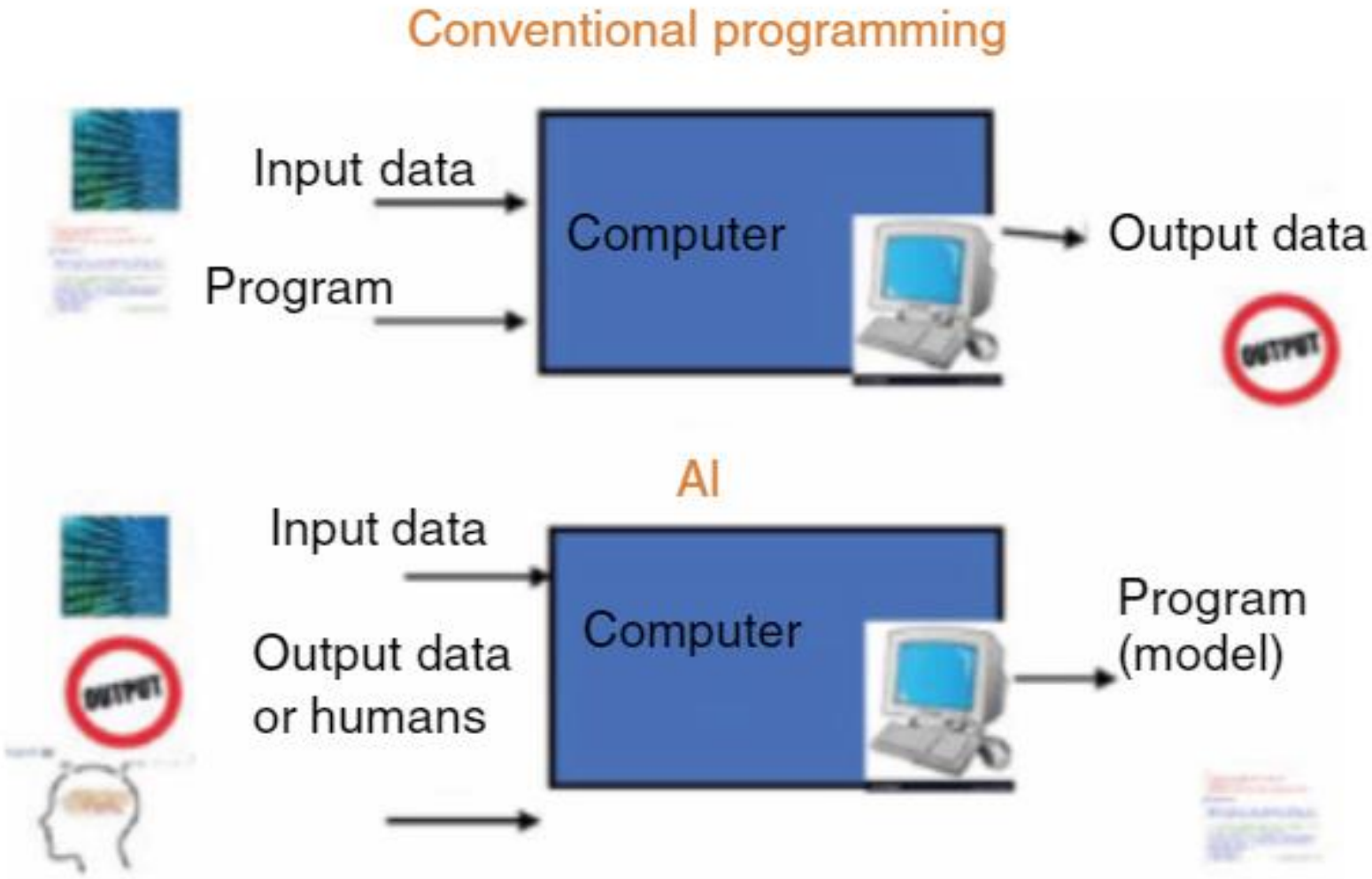


Figure 1.12 AI techniques.

Conventional vs AI Approaches to Coding

Programming without AI	Programming with AI
A computer program without AI can answer the **specific** questions it is meant to solve.	A computer program with AI can answer the **generic** questions it is meant to solve.
Modification in the program leads to change in its structure.	AI programs can absorb new modifications by putting highly independent pieces of information together. Hence, you can modify even a minor piece of information or program without affecting its structure.
Modification is not quick and easy. It may lead to affecting the program adversely.	Quick and easy program modification.


Conventional programming
Input data
Program
Computer
Output data
AI
Input data
Output data
or humans
Computer
Program
(model)

Rules Based vs Expert Systems

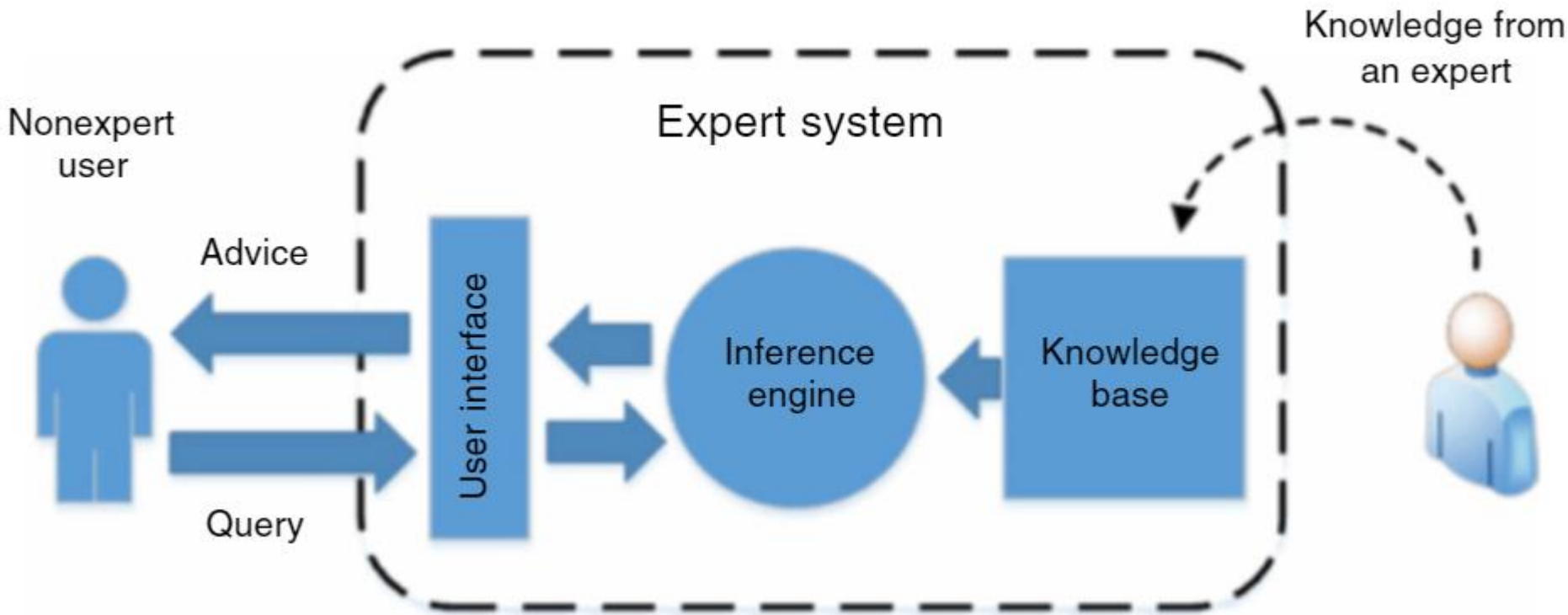
Rules-Based Systems:

A rules-based system is a type of software that uses a set of predefined rules to make decisions or perform actions. These rules are typically expressed in the form of "if-then" statements, where certain conditions trigger specific actions. Rules-based systems are widely used in various domains for decision support and automation.

Expert Systems:

An expert system is a type of artificial intelligence (AI) system designed to emulate the decision-making ability of a human expert in a specific domain. It uses a knowledge base of facts and heuristics to draw inferences and make decisions. Expert systems are particularly valuable in situations where human expertise is critical but may not be readily available.

An Expert System Composition and Operation



The basic structure of an expert system consists of the following parts.

- The knowledge base of rules.
- The database of facts,
- the inference engine,
- the interpreter,
- and the human computer interaction interface.
- The knowledge base is used to store an expert system expertise, including the database of facts and the rules employed to process them with the inference engine. It should be able to acquire new knowledge, expressing and storing knowledge in a way that the computer can accomplish.
- The inference engine matches the facts in the database with the rules from the knowledge base and derives new information which is placed back in the database.
- The explanation facilities are responsible for interpreting the results of the inference engine output. Including explaining the correctness and reason of the conclusion to the user. That is done through a specialised user interface.

- The knowledge base consists of rules, the term rule in AI which is the most commonly used type of knowledge representation can be defined as an IF-THEN structure that relates, given information or facts in the IF Part to some action in the THEN part.
- A rule provides some description of how to solve a problem.
- Rules are relatively easy to create an understand.
- Any rule consists of 2 parts. The IF Part called the **antecedent** and the THEN part called the **consequent**. It has the following generic format.

IF antecedent THEN consequent.

Rules can represent relations, recommendations, directives, strategies, and heuristics, for example:

- **Relation:** IF the fuel tank is empty THEN the car is dead.
- **Recommendation:** IF the season is autumn AND the sky is cloudy AND the forecast is drizzle THEN the advice is "take an umbrella."
- **Directive:** IF the car is dead AND the fuel tank is empty THEN the action is "refuel the car."
- **Heuristic:** IF the spill is liquid AND the spill pH < 6 AND the spill smell is vinegar THEN the "spill material" is "acetic acid."

ES apply heuristics to guide the reasoning and thus reduce the search space for a solution. A unique ES feature in comparison to conventional programming technologies is its explanation capability. It enables the ES to review its own reasoning and explain its decisions. ES employ symbolic reasoning when solving a problem. Symbols are used to represent different types of knowledge such as facts, concepts, and rules. An ES is built to perform at a human expert level in a *narrow, specialized domain* problem solving. Thus, the most important ES characteristic is its high-quality performance.

However, no matter how fast the system can solve a problem, a user will not be satisfied if the result is wrong. On the other hand, the speed of reaching a solution is very important. Even the most accurate decision or diagnosis may not be useful if it is too late to apply, for instance, in an emergency, when a patient dies or a nuclear power plant explodes before the decision is made. Table 1.5 lists the advantages and disadvantages of the expert system technology, among which the relatively low speed (in comparison against other computer technologies) plays the main role that results in severe limitations in the application domain space. Most industrial systems have less than a couple of hundreds of rules only as their processing might take a considerable time. This is a major reason why general problems solvers, which AI founders were dreaming about, have not been put in practice yet.

Table 1.5 Expert systems: good and bad.

Advantages	Disadvantages
Natural knowledge representation: An expert usually explains the problem-solving procedure with expressions like "In such-and-such situation, I do so-and-so." These expressions can be represented quite naturally as IF-THEN production rules.	**Opaque relations between rules:** Although the individual production rules are relatively simple and self-documented, their logical interactions within the large set of rules may be opaque. Rule-based systems make it difficult to observe how individual rules serve the overall strategy.
Uniform structure: Production rules have the uniform IF-THEN structure. Each rule is an independent piece of knowledge. The very syntax of production rules enables them to be self-documented.	**Ineffective search strategy:** The inference engine applies an exhaustive search through all the production rules during each cycle. Expert systems with a large set of rules (over 100 rules) can be slow, and thus large rule-based systems can be unsuitable for real-time applications.
Separation of knowledge from its processing: The structure of a rule-based expert system provides an effective separation of the knowledge base from the inference engine. This makes it possible to develop different applications using the same expert system shell.	**Inability to learn**. In general, rule-based expert systems do not have an ability to learn from the experience. Unlike a human expert, who knows when to "break the rules," an expert system cannot automatically modify its knowledge base, or adjust existing rules or add new ones. The knowledge engineer is still responsible for revising and maintaining the system.
Dealing with incomplete and uncertain knowledge: Most rule-based expert systems are capable of representing and reasoning with incomplete and uncertain knowledge.	

Feature	Rules-Based Systems	Expert Systems
Rule Representation	Uses explicit "if-then" rules, where conditions trigger specific actions.	Knowledge base contains facts, rules, and heuristics to emulate human expertise. Rules may also be in "if-then" format.
Inference Engine	Applies rules to data or inputs using a straightforward inference engine.	Utilizes an inference engine that processes the knowledge base, employing various reasoning methods like forward chaining or backward chaining.
Knowledge Base	Primarily consists of rules that define conditions and actions.	Knowledge base includes facts, relationships, and inference mechanisms, capturing expert knowledge in a specific domain.
Scalability	Can become complex as the number of rules increases. Managing and updating large rule sets may require careful organization.	Complexity is handled through the organization of knowledge in the knowledge base. The system can handle a large amount of expert knowledge.

Feature	Rules-Based Systems	Expert Systems
Transparency	Rules are explicit and easily understandable, providing transparency in decision-making.	Transparency in decision-making is maintained through explicit representation of expert knowledge in the knowledge base.
Learning and Adaptation	Learning mechanisms are typically limited; the system relies on predefined rules.	Can be designed with learning mechanisms to refine knowledge and improve performance over time. Learning may involve updating the knowledge base based on feedback or new data.
Complex Problem Solving	Suited for problems with well-defined conditions and actions.	Designed for complex problem-solving tasks within a specific domain. Can handle uncertainty, incomplete information, and ambiguous situations.
Applicability	Commonly used in business rules, workflow automation, and decision support systems.	Applied in areas such as medical diagnosis, financial analysis, troubleshooting, and other domains where specialized expertise is crucial.

Humans Vs Expert Systems Vs Conventional Programming Approaches

Human, Expert System, and Conventional Programming approaches are three different ways to solve problems or perform tasks, each with its own characteristics and applications. Here's a comparison of these approaches:

1. **Nature of Decision-Making:**
   - **Human:** Relies on human intuition, experience, and cognitive abilities to make decisions.
   - **Expert System:** Uses predefined rules and knowledge base to make decisions, mimicking human expertise.
   - **Conventional Programming:** Follows a set of predetermined instructions to make decisions.

2. **Knowledge Representation:**
   - **Human:** Knowledge is acquired through experience, learning, and intuition.
   - **Expert System:** Knowledge is explicitly represented in a knowledge base using rules, facts, and inference mechanisms.
   - **Conventional Programming:** Knowledge is encoded in the form of algorithms and data structures.

3. Flexibility and Adaptability:

- **Human:** Adaptable to new and unforeseen situations, can learn and adjust based on experience.
- **Expert System:** Relatively rigid, may struggle with unforeseen situations unless explicitly programmed for them.
- **Conventional Programming:** Lacks flexibility, requires code modification for adapting to new scenarios.

4. Problem-solving Approach:

- **Human:** Holistic and adaptable approach, able to handle complex and ambiguous situations.
- **Expert System:** Rule-based reasoning and pattern recognition, suited for well-defined problems within a specific domain.
- **Conventional Programming:** Algorithmic approach, suitable for problems with well-defined logic and clear step-by-step solutions.

5. Learning Capability:

- **Human:** Constantly learning and adapting through experience.
- **Expert System:** Limited learning ability, typically requires human intervention to update knowledge.
- **Conventional Programming:** Static and does not inherently learn from data or experience.

6. Domain of Application:

- **Human:** Universal applicability across a wide range of domains.
- **Expert System:** Specific to a well-defined domain where expertise can be codified.
- **Conventional Programming:** General-purpose and applicable to a wide variety of tasks.

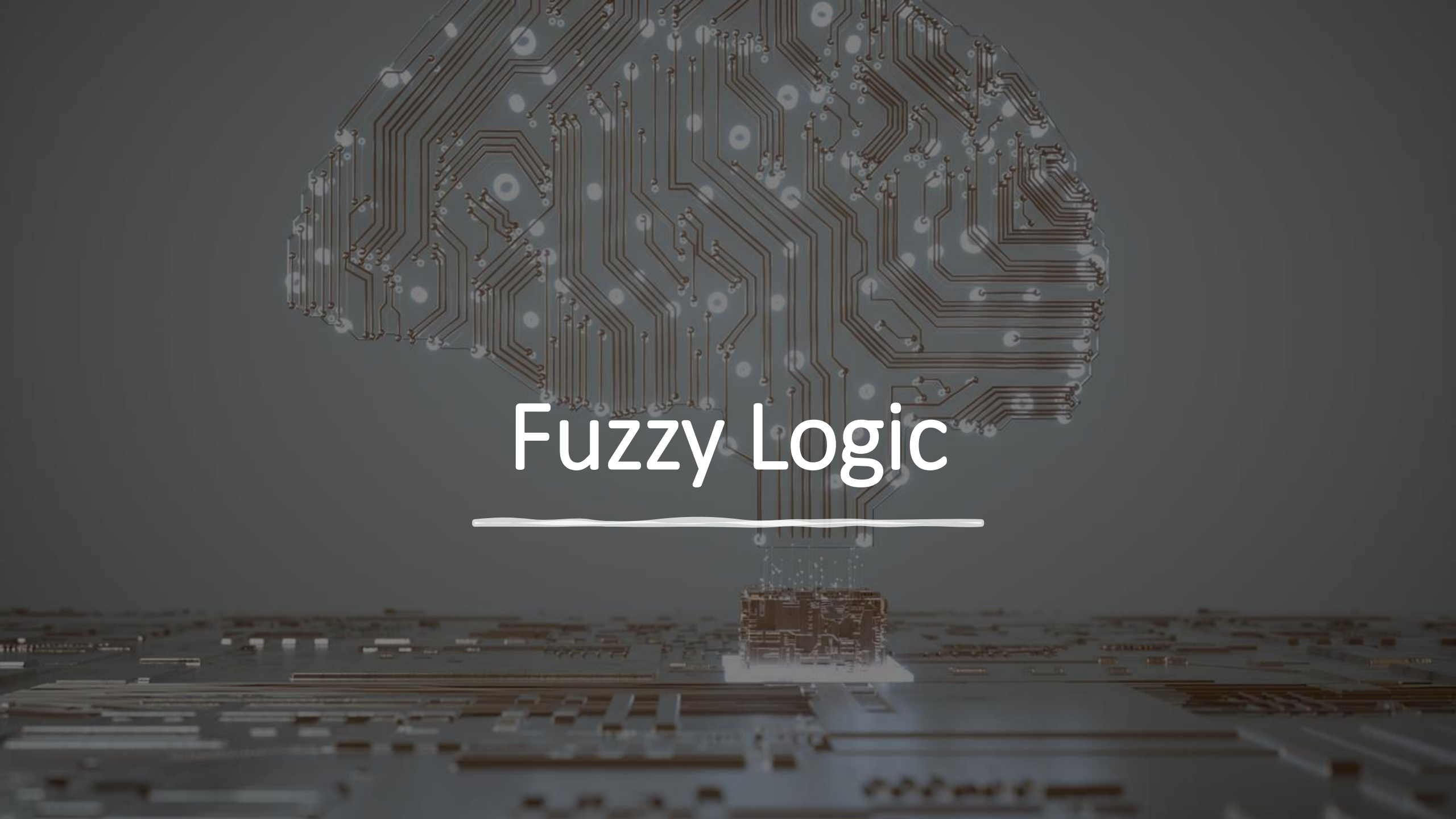
7. Development Time and Effort:

- **Human:** High development time, but adaptable and capable of handling diverse tasks.
- **Expert System:** Initial development time may be high, but once developed, it can provide consistent performance.
- **Conventional Programming:** Moderate to high development time, depending on the complexity of the problem.

8. Debugging and Maintenance:

- **Human:** Adaptable to changing conditions but may be prone to errors and biases.
- **Expert System:** Debugging and maintenance are relatively easier compared to conventional programming, as changes can often be made in the knowledge base.
- **Conventional Programming:** Debugging and maintenance can be complex, especially for large systems.

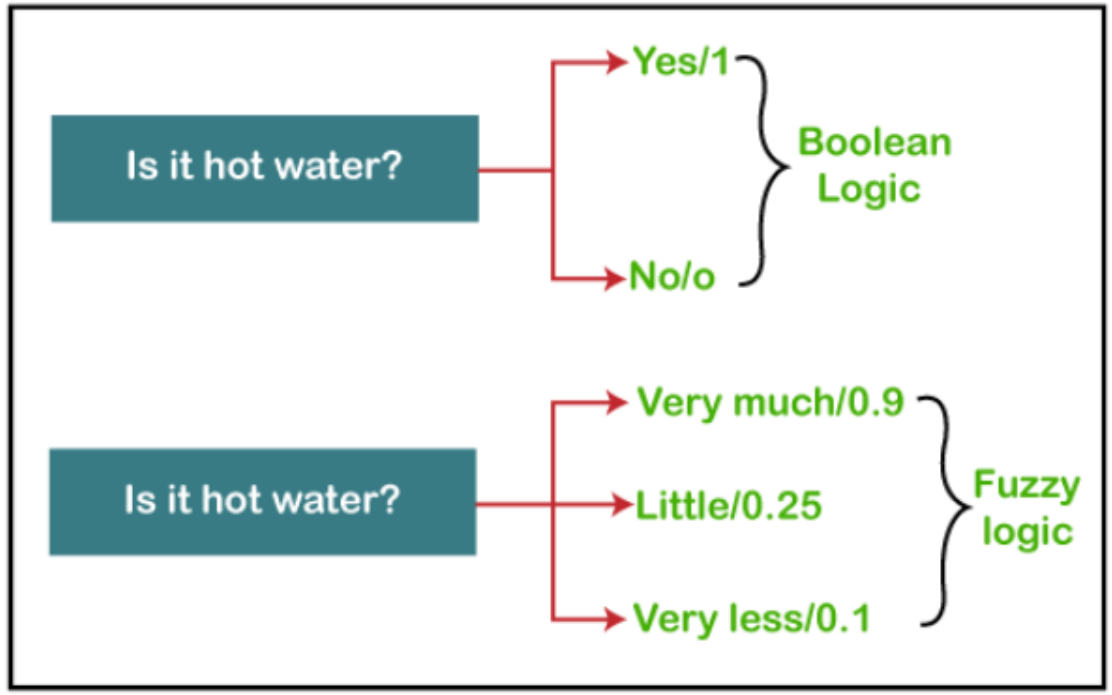
Human experts	*Expert systems*	*Conventional programs*
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a *narrow domain*.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a *clear separation of knowledge from its processing*.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	*Trace the rules fired* during a problem-solving session and *explain how* a particular conclusion was reached and *why* specific data were needed.	Do not explain how a particular result was obtained and why input data were needed.
Use inexact reasoning and can deal with incomplete, uncertain, and fuzzy information.	Permit *inexact reasoning* and can deal with incomplete, uncertain, and fuzzy data.	Work only on problems where data are complete and exact.
Can make mistakes when information is incomplete or fuzzy.	*Can make mistakes* when data are incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data are incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient, and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, *changes are easy* to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.



Fuzzy Logic

The **'Fuzzy'** word means the things that are not clear or are vague. Sometimes, we cannot decide in real life that the given problem or statement is either true or false. At that time, this concept provides many values between the true and false and gives the flexibility to find the best solution to that problem.

Example of Fuzzy Logic as comparing to Boolean Logic



- Fuzzy logic contains multiple logical values and these values are the truth values of a variable or problem between 0 and 1.
- This concept was introduced by **Lofti Zadeh** in **1965** based on the **Fuzzy Set Theory**. This concept provides the possibilities which are not given by computers, but similar to the range of possibilities generated by humans.
- In the Boolean system, only two possibilities (0 and 1) exist, where 1 denotes the absolute truth value and 0 denotes the absolute false value. But in the fuzzy system, there are multiple possibilities present between the 0 and 1, which are partially false and partially true.
- The Fuzzy logic can be implemented in systems such as micro-controllers, workstation-based or large network-based systems for achieving the definite output. It can also be implemented in both hardware or software.

Characteristics of Fuzzy Logic

Following are the characteristics of fuzzy logic:

1. This concept is flexible and we can easily understand and implement it.
2. It is used for helping the minimization of the logics created by the human.
3. It is the best method for finding the solution of those problems which are suitable for approximate or uncertain reasoning.
4. It always offers two values, which denote the two possible solutions for a problem and statement.
5. It allows users to build or create the functions which are non-linear of arbitrary complexity.
6. In fuzzy logic, everything is a matter of degree.
7. In the Fuzzy logic, any system which is logical can be easily fuzzified.
8. It is based on natural language processing.
9. It is also used by the quantitative analysts for improving their algorithm's execution.
10. It also allows users to integrate with the programming.

Architecture of a Fuzzy Logic System

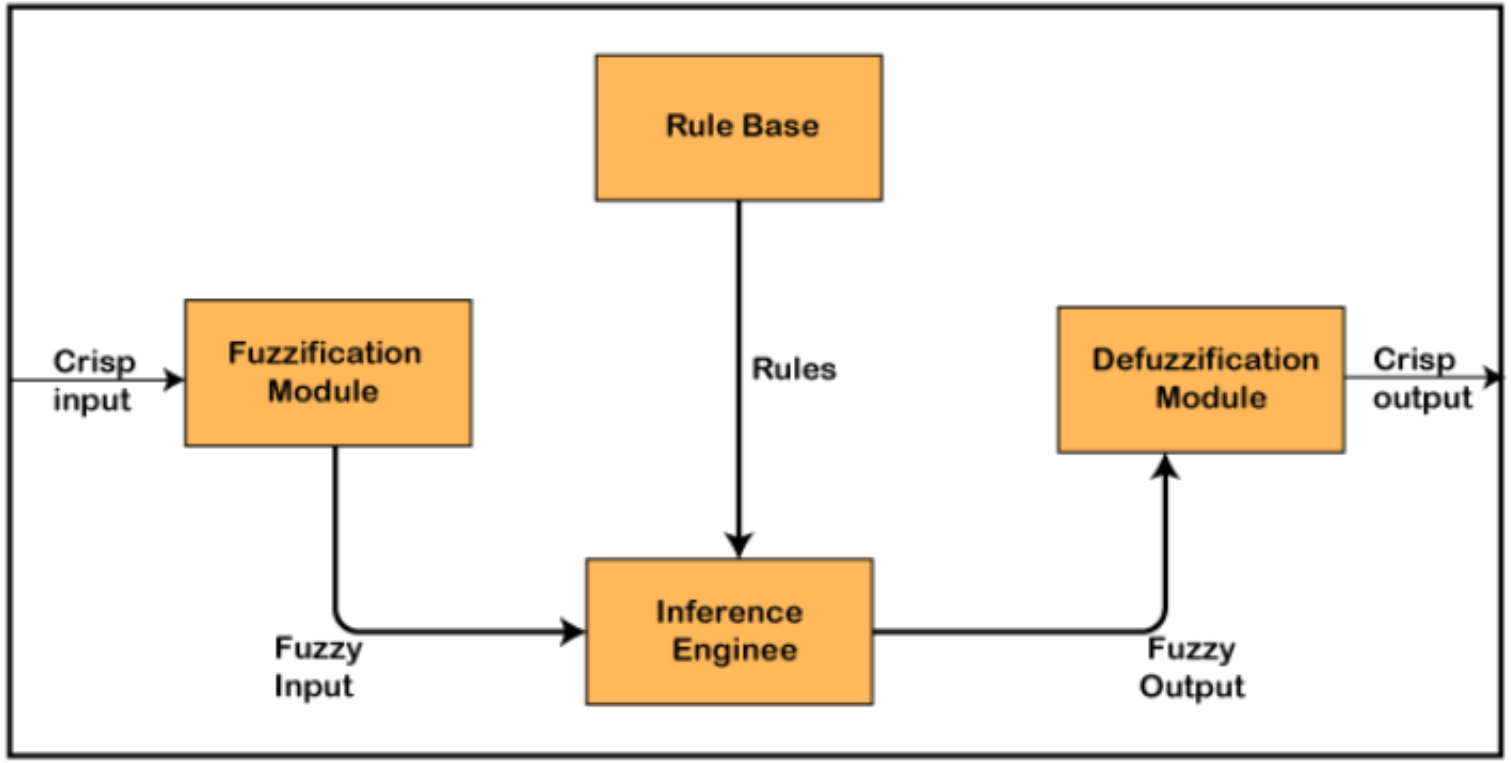
In the architecture of the **Fuzzy Logic** system, each component plays an important role. The architecture consists of the different four components which are given below.

1.Rule Base

2.Fuzzification

3.Inference Engine

4.Defuzzification



1. Rule Base

- Rule Base is a component used for storing the set of rules and the If-Then conditions given by the experts are used for controlling the decision-making systems. There are so many updates that come in the Fuzzy theory recently, which offers effective methods for designing and tuning of fuzzy controllers. These updates or developments decreases the number of fuzzy set of rules.

2. Fuzzification

- **Fuzzification** is a module or component for transforming the system inputs, i.e., it converts the crisp number into fuzzy steps. The crisp numbers are those inputs which are measured by the sensors and then fuzzification passed them into the control systems for further processing. This component divides the input signals into following five states in any Fuzzy Logic system:
- Large Positive (LP)
- Medium Positive (MP)
- Small (S)
- Medium Negative (MN)
- Large negative (LN)

3. Inference Engine

- This component is a main component in any Fuzzy Logic system (FLS), because all the information is processed in the Inference Engine. It allows users to find the matching degree between the current fuzzy input and the rules. After the matching degree, this system determines which rule is to be added according to the given input field. When all rules are fired, then they are combined for developing the control actions.

4. Defuzzification

- **Defuzzification** is a module or component, which takes the fuzzy set inputs generated by the **Inference Engine**, and then transforms them into a crisp value. It is the last step in the process of a fuzzy logic system. The crisp value is a type of value which is acceptable by the user. Various techniques are present to do this, but the user has to select the best one for reducing the errors.

?	Why do we need fuzzy rules instead of conventional ones?
!	First off, fuzzy rules do not replace conventional rules, they add up to them. Conventional rules require an expert to provide numerical values for certain parameters, which they might not know or hesitant to release. For example, as a kid, did you ask your mom at what distance from the car in front of you and what speed of your car should you hit the brake? Do you recall the answer? Probably, it was like: "When the car is close to the obstacle and the speed is fast, brake!"

Many experts find that fuzzy rules provide a convenient way to express their domain knowledge with the help of linguistic variables. This approach was very popular in the design of pioneer fuzzy systems (FS) and controllers. In application design, fuzzy logic exploits its tolerance to imprecision. The advantage of using fuzzy rules systems may include the following characteristics of fuzzy logic:

- This concept is flexible and commonly is easy to understand and implement.
- It may help to extract the human expert's knowledge and optimize the operation with it.
- It provides a very good technique for finding the solution of those problems, which are suitable for approximate or uncertain reasoning.
- It includes mathematical provisions for NLP and the methods for the quantitative analysis of variables expressed in natural language.

Typically, the fuzzy rules system takes numerical values as inputs and fuzzifies them on the first step (see Figure 1.14 for an example). Fuzzification involves the comparison of the crisp input values against the input membership functions and the calculation of the fuzzy input value. Sometimes, inputs might be given as fuzzy or linguistic variables, in which case this step could be skipped. Then, fuzzy inputs are compared against fuzzy rules conditions and the applicability degree for each rule is calculated. On the next step, fuzzy rules are inferred and their outputs are produced. The outputs might be fuzzy in the case of Mamdani-type processing or crisp in the case of Takagi–Sugeno rules processing method application. In the first case, the fuzzy outputs are defizzified by producing a crisp output value. In the second case, the output is calculated as an algebraic function of fuzzy input values.

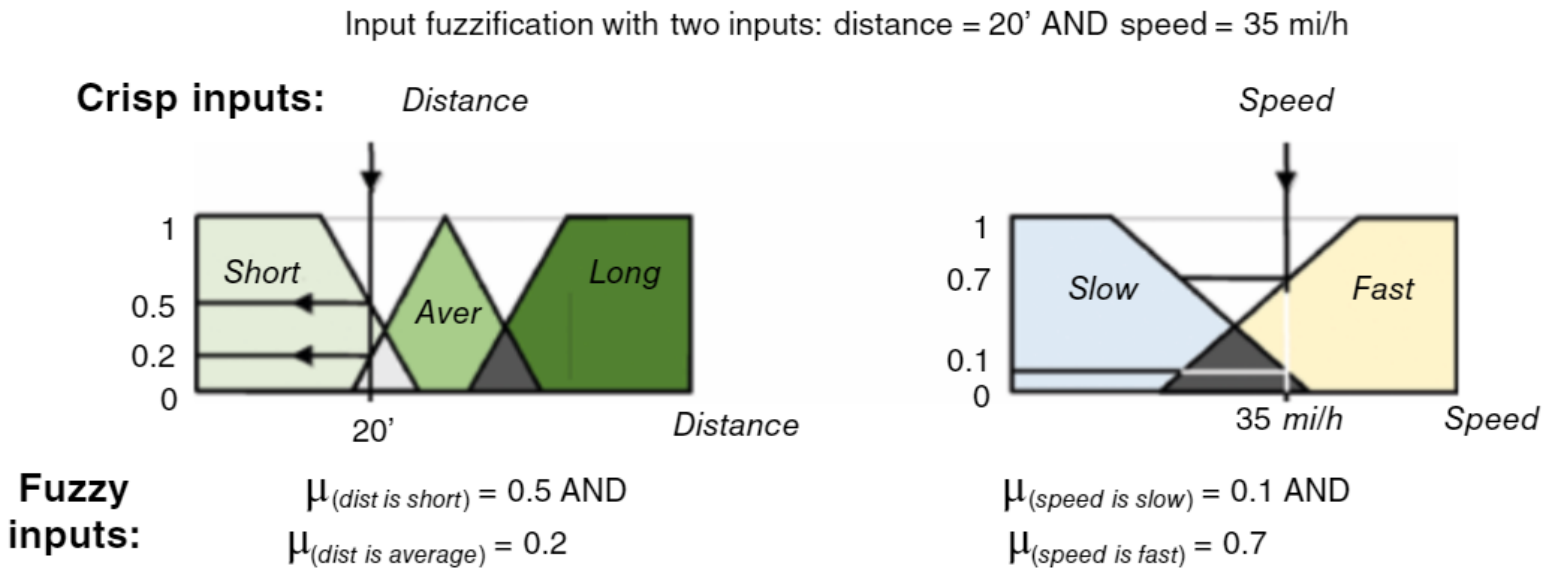


Figure 1.14 Fuzzification of crisp inputs.

Example 1.8 Neuro-Fuzzy Hybrid System (Negnevitsky and Kelareva 2001)

This example describes the design of a neuro-fuzzy system that has been used to generate a model for predicting air quality in Launceston, Tasmania, Australia. Using the input variables in Table 1.6, with the multilayered structure derived below, a prediction system can be built using the hybrid model, with results that can be interpreted by a human expert to aid the prediction of air quality.

A neuro-fuzzy system model is designed applying a multilayer neural network. In this approach, a neuro-fuzzy system is composed from a multilayer neural network with different layers representing input and output layers, and three hidden layers representing fuzzy membership functions and rules. Each layer in a neuro-fuzzy system is associated with a particular procedure in the process of fuzzy inferencing. This process of model generation includes creating a multilayer structure.

Layer 1 is the input layer; neurons in this layer transmit external crisp signals directly to the next layer. The neuro-fuzzy system in Figure 1.15 has seven input neurons ($N = 7$). Layer 2 is the fuzzification layer. Here, neurons represent fuzzy sets used as the antecedents of fuzzy rules. Fuzzification neurons within this layer receive crisp input and determine the degree, to which this input belongs to the neuron's fuzzy set. The activation function of a membership neuron is set to a triangular function.

Layer 3 is the fuzzy rule layer, where each neuron represents a single fuzzy rule. A fuzzy rule neuron receives its inputs from the fuzzification neurons representing fuzzy sets in the rule antecedent. To perform the antecedent evaluation, the product operator is used. Layer 4 is the fuzzy output layer, where neurons represent fuzzy sets as consequents of fuzzy rules. An output membership neuron receives inputs from the corresponding fuzzy rule neurons and combines them by using the probabilistic OR operation. The output membership layer is represented by two neurons. Layer five is the defuzzification layer; it consists of a single output neuron. The neuro-fuzzy system applies the sum–product composition method to produce a crisp output.

Table 1.6 Input variables for air quality control neuro-fuzzy system.

Variable	Description	Range
x_1	3 pm surface air temp (°C)	4.6–20.2
x_2	3 pm surface wind speed (m/s)	0–13.9
x_3	3 pm cloud cover (oktas)	0–8
x_4	3 pm MSL pressure (hPa)	976.9–1037.5
x_5	9 am MSL pressure (hPa)	981.2–1039.7
x_6	9 pm 850 hPa wind speed (m/s)	1–30.9
x_7	Dateterm	0.01–3.3

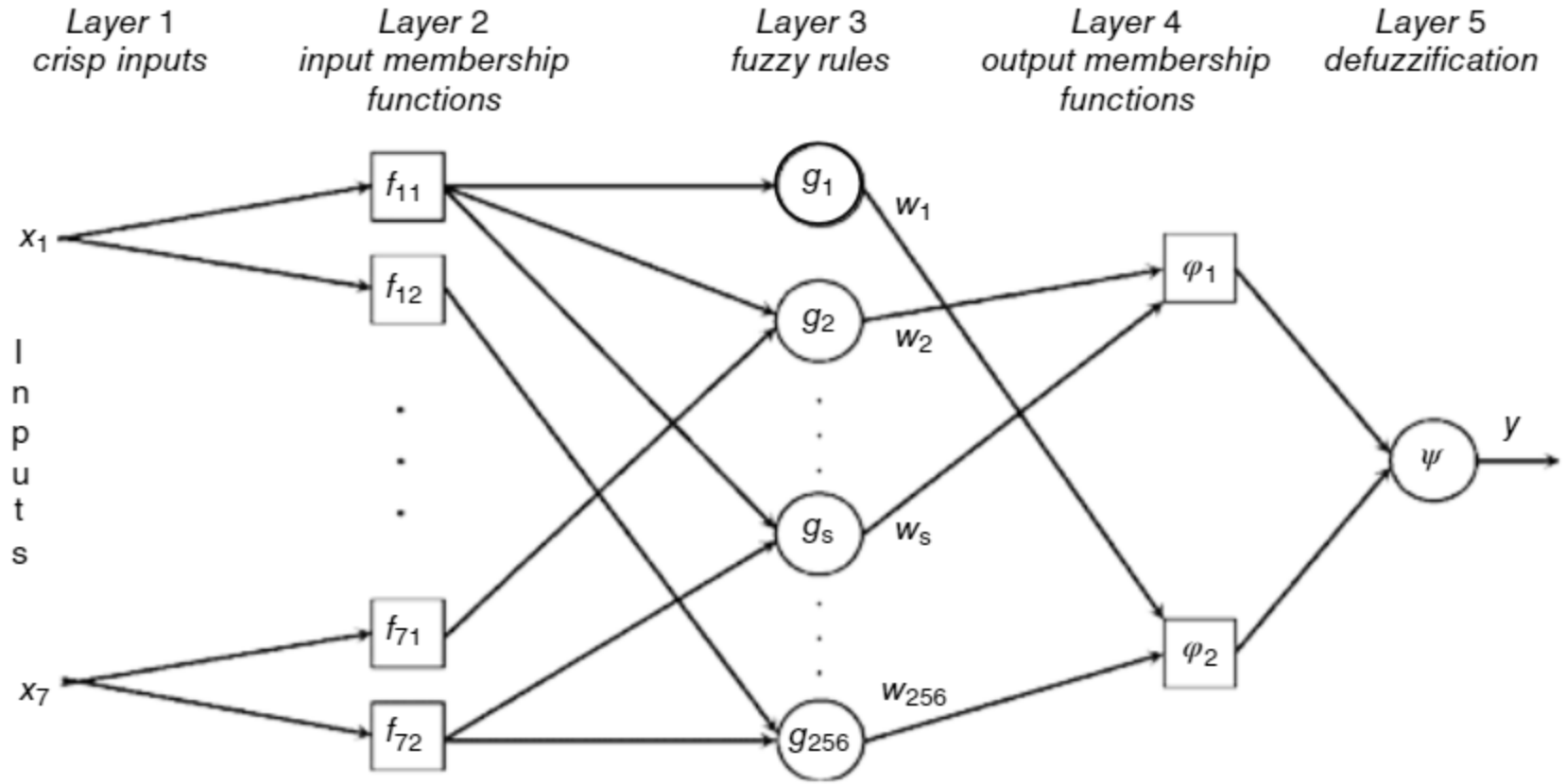


Figure 1.15 Typical architecture of the neuro-fuzzy system.

Weights between layer 3 and layer 4 reflect the importance of the corresponding fuzzy rules. The weights close to one indicate rules, whose role is critical to the system performance, while the weights close to zero indicate rules that are less important or even questionable. FS should not include false and redundant rules. Thus, one of the key goals of training a neuro-fuzzy system is to remove such rules. It can be done by setting the respective weights to zero when they fall below a specified threshold.

Machine Learning and Data Science

Contribution of machine learning and data science to cybersecurity.

The appeal and pervasiveness of ML has been increasing over time. Existing methods have been improved, and their ability to understand and address real-life problems is highly appreciated. These achievements have led to the ML methods adoption in numerous domains, especially in computer vision, medical analysis, and of course, cybersecurity. In some scenarios, ML techniques represent the best choice over traditional rule-based algorithms and even human operators. This trend is also affecting the cybersecurity field with multiple ML-based applications put into service.

?	What is ML?
!	ML is a subfield of AI that comprises the study of algorithms, which have a capability to improve themselves automatically through experience to solve problems without external instructions, by using previously trained models.
?	How is learning done in ML?
!	In ML, the model of a decision-making or a control process is learned from the available data. This learning process could be performed with one of three generic approaches: 1) Supervised and semi-supervised learning 2) Unsupervised learning 3) Reinforcement learning

Commonly, ML is applied to solve one of the following problems: classification, regression/prediction, or clustering.

Supervised and semi-supervised learning algorithms develop a mathematical model from the input data and known desired outputs. In mathematical meaning, each training data piece is represented by an array or a vector and the whole group by a matrix mapping inputs to the desired output values.

?	If we already know the output values, what else do we need to learn?
!	We know a SAMPLE of output values only. Commonly, the known number of samples is pretty small in comparison to the dimension of all possible output values.

The algorithm can develop an objective function through an iterative optimization in order to predict the possible output from another input, for which an output value is not given. ML models are used for classification and regression. ML provides effective tools to automate classification tasks. A classifier can be seen as the function f: X→Y, mapping any object from the feature space X into a corresponding class from Y. The subfield of supervised learning studies effective techniques to automatically generate classifiers starting from a set of labeled data. Therefore, to fruitfully use supervised learning, one must take the steps described in Algorithm 1.1.

The model M that has been learned through training is now employed to solve a classification problem. Algorithm of getting the regression model is similar to Algorithm 1.1. In supervised learning, the model is built up in training. Over this training period, the model CANNOT be used to solve problems, for which it is being developed. We may start exploiting it for problem solving only after the training is complete.

Algorithm 1.1 Supervised Learning Approach

1) Collect data representing the set of objects of interest O, for example, data from packet headers representing traffic.
2) Define the set of classes Y. The number of classes could be two in the case of a binary classification. For example, an anomaly detection aims to discriminate the class of packets representing normal traffic (class 0) and the class of packets representing any attack (class 1). Or in another example, when the goal is not only to detect an attack but to identify it too, the number of classes could be $n+1$ when there could be n various attack classes, which one wants to identify.
3) Define the feature space X by specifying the salient aspects that look useful to assign the objects in O to their correct class in Y. For example, one could decide to employ the origin address, the origin port, the destination address and port, and the protocol-type data to differentiate between attacks and normal traffic.
4) Build a training set D of labeled pairs (x^*,y^*), where each x^* is the set of features of an object $o \in O$ encoding in X and y^* is its class.
5) Employ the ML technique that will use the training dataset D to build up the model M that will map the set of feature data X into a class set Y. On this stage, only pairs (x^*, y^*) that belong to the training set D will be used.
6) Use model M to find a class for any object O through mapping its feature set x into the class y. At this stage, x is not required to belong to the training set D.

?	How to get those features?
!	This is a very good question! Until recently, feature extraction had been performed either manually or semi-manually, although multiple techniques were developed and employed for help. However, new approaches of DL allow to automate this process, so the corresponding step becomes internal and hidden from a user.

As you see, after learning is complete, the learnt model could be used to find the labels or to predict the value for inputs composed of new unknown data.

?	Looks good! Supervised learning seems to be absolutely marvelous.
!	It is not perfect. Supervised learning requires a teacher or some sort of supervision to direct training.
?	There is no supervisor in your algorithm, is it?
!	Training dataset plays this role. It has to be provided, otherwise supervised learning is not possible. Also, it has to be big enough to represent all the patterns that are hidden in the data. Unfortunately, collecting those data representing relationships between the model inputs and outputs and then labeling them could be time and effort consuming, and expensive.

In other words, the initial problem we are trying to solve has to be already solved for a reasonable number of cases and these solutions have to be accessible in a supervised ML approach. If a sufficient dataset is not available for training, one should try other ML approaches.

Unsupervised learning algorithms are mainly employed for clustering purposes. They take a set of data consisting only of inputs and then they attempt to cluster the data objects based on the similarities or dissimilarities in them, without the need for the data to be previously classified, labeled, or categorized. The generic way of solving the problem in the unsupervised approach is provided in Algorithm 1.2.

As one can see in Table 1.7, this approach does not require direct supervision or a training dataset. The output could be produced straightaway as soon as you feed your inputs to the model, so one does not have to wait until the model is trained. The model training and its application in solving a particular problem are executed at the same time.

Algorithm 1.2 Unsupervised Learning Approach

1) Define the features space X by specifying the salient aspects that look useful to assign the objects in O to their correct class in Y – this step is similar to Algorithm 1.1.
2) Choose the ML technique that allows an unsupervised learning implementation.
3) Based on the chosen technique, build up the initial version M_0 of the model M that will map the set of features data X into a class set Y. The model can be generated as a random structure within certain limitations depending on the technique. Or some knowledge and transformational learning approaches could be used to inherit the model from previous results.
4) Feed the current model M_i with the input data object x representing object o and make the model produce the output y. This output could be used as a problem solution.
5) Based on input and output data pairs (x,y), produce the next model version M_{i+1} with the unsupervised learning algorithm.
6) Loop back to step 5.

Table 1.7 Comparison between supervised and unsupervised ML technique approaches.

Features	Supervised ML techniques	Unsupervised ML techniques
Process phases	The whole process is divided into two phases: training and execution	The whole process is not divided into phases
Output solution generated	The solution is not generated in the training but only after the first phase is complete	The solution is generated from the beginning of the process
Provided data	Algorithms are trained using labeled data.	Algorithms do not need training data
Algorithms used examples	Support vector machine, Neural network, Linear and logistics regression, random forest, and Classification trees.	Cluster algorithms, K-means, Hierarchical clustering
Computational complexity	Supervised learning is a simpler method.	Unsupervised learning is generally more computationally complex
Accuracy of results	Highly accurate and trustworthy method.	Less accurate and trustworthy method.
Real-time learning	Learning takes place offline before execution.	Learning takes place in real time, simultaneously with model execution.
Number of classes	Commonly, the number of classes is known.	Commonly, the number of classes is not known.
Main drawback	Requires training dataset, have to wait until training is complete to get the solution	The solution quality could be low

?	Can one guarantee that the derived solution will be correct?
!	No, but one cannot guarantee that the solution provided by supervised training will be correct either as in supervised training the solution quality depends on multiple factors: inclusion of various patterns representation into the training set, the training set size, its distribution, etc.

?	What is the difference then?
!	Generally, in supervised learning, the solution is expected to get better with training. So, it is considered more reliable as, at least, it provides some training direction given by the supervisor or the training dataset.
?	Will unsupervised learning make a solution better too?
!	Unsupervised learning may move the solution in the wrong direction as generally it has no constraints. With unsupervised learning, it is even hard to compare solutions as no comparison base or criteria are given.

Outlier detection and clustering are typical examples of the problems solved with unsupervised learning techniques. They may use unlabeled data and might be capable to detect unknown types of attacks, for an instance.

Reinforcement learning algorithms are mainly associated with dynamic programming techniques and are mostly used in AI development (i.e. autonomous cars, AI used in games). Generally, these are the most advanced ML algorithms aimed toward maximizing the cumulative reward from the actions that algorithm performs (for example, winning a round in a game as a team and not just gaining the best possible score for each individual AI player). Reinforcement learning differs from the supervised learning as the training data used in supervised learning clearly determines the direction how the model is trained with the correct answer, whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task to maximize the given function. In the absence of a training dataset, it is bound to learn from its experience. As Liang et al. (2019) points out, reinforcement learning is considered to be positioned somewhere between supervised and unsupervised learning, since the input of reinforcement learning has no label information that is given in supervised learning, but instead is associated with a reward value, such that each execution improves the decision-making of the overall model, typically by maximizing the rewards. This can be represented by the perception–action–learning loop. The two major reinforcement methods are policy search and value function approximation.

- **Policy search:** This is the search for an optimal policy using gradient-based or gradient-free methods. For example, Google's Alpha Go is based on policy search, and can learn without any human intervention or interaction but still achieves superiority.
- **Value function approximation:** This method estimates the expected rewards of actions and attempts to reach an optimized learning process and results. The key component is the state-action value function, known as the quality function.

ML In Cybersecurity

Contribution of machine learning and data science to cybersecurity.

As typical for AI fields, ML involves a large variety of algorithms and paradigms in continuous evolution, presenting weak boundaries and cross relationships. Furthermore, different views and applications may lead to different classifications. Below, two classification approaches are given, one (Figure 1.16) presents a traditional taxonomy, based on an algorithm structure and basic operational principles. Another one (Figure 1.17) builds up the classification on the learning approach employed and attempts to consider the new methods latest development trends. Please note that both classification schemes are oriented toward ML applications in the cybersecurity domain and do not even attempt to build up the ultimate taxonomy that can cover all cases and satisfy all AI experts.

- **Symbolists** – These algorithms essentially focus on an inverse deduction. Instead of starting with the premise and looking for conclusions, symbolists start with some premises and conclusions, and try to fill the gaps in between. Among the symbolists, the most notorious are decision trees and random forests.

A decision tree (DT) is a tree-structure resembling a flowchart, where every node represents a test to an attribute, each branch represents the possible outcomes of that test, and the leaves represent the class labels. Each DT presents a conditional classifier: the tree is visited from the top and, at each node, a given condition is checked against one or more features of the analyzed data. The paths from root to leaves represent the decision rules. These methods are efficient for large datasets and excel at multiclass problems, but deeper trees might lead to overfitting.

A random forest (RF) is an ensemble learning method that builds a large group of independent decision trees, and outputs the mode of the label predictions of all the trees. This method has higher computing costs, but tends to reduce overfitting of the data, which happens when a model adapts too strictly to a particular set of data, having poor capabilities for generalization.

- **Connectionists** - This approach involves creating artificial processing elements (PE) or neurons and connecting them in neural networks. Connectionist algorithms revolve around the usage of neural networks.

ANN propose a generic methodology that would allow to build up a network of PEs connecting the model inputs to its outputs. Connections are assigned certain weights. Through the training process that consists of adjusting the weights of the connections, the model learns the correct structure in which its outputs present the correct values in response to each input.

Autoencoders are a type of neural network that aim to reconstruct data from the input layer into the output layer with a minimal amount of distortion. As the objective function is calculated with a comparison between the inputs and outputs, autoencoders represent an unsupervised learning algorithm, since they do not require class labels to be trained.

- **Evolutionaries** attempt to build up and use the strategy of evolution that should drive learning to produce the better solutions, commonly in a more efficient way.

Genetic algorithms (GA) represent a set of evolutionary algorithms, which take an inspiration from genetic evolution theories by Charles Darwin. The algorithms typically start with a set of individuals, and through processes of mutation and crossovers between multiple individuals, new populations are generated. As the algorithms progress, the most fit individuals have a higher chance of reproducing, leading to a more fitting population on each iteration. At the end, the member of the final generation is chosen as a solution.

- **Bayesians** focus on applying probabilistic inference, such as the Bayes theorem assuming that the outcomes probabilities could be estimated based on known "a priori" distributions.

Naive Bayes (NB) is a ML algorithm that consists of applying the Bayes theorem in order to find a distribution of conditional probabilities among class labels, with the assumption of independence between features. It has the advantage of being highly scalable.

Hidden Markov Models (HMM) build up as a set of states producing outputs with different probabilities with the goal to find out the sequence of states that results in the observed outputs. HMM are effective for understanding the temporal behavior of the observations, and for calculating the likelihood of a given sequence of events. Although HMM can be trained on labeled or unlabeled datasets in general, in computer security, they have mostly been used with labeled datasets.

Analogizers exploit the concept that close elements are more strongly related, essentially matching related pieces of data. Among this category, the most popular ones are k-nearest-neighbors (KNN), SVM, and K-means algorithms.

KNN is a classification algorithm that uses a distance function (for example, Euclidean distance) in order to determine to which class to assign the new element by finding the k closest elements in the feature space. The final label will be determined based on the distribution of classes among those neighbors.

K-means clustering uses a distance function too to distribute all data pieces between k clusters defined by their centroid position in the feature space. This is typically an unsupervised algorithm that redistributes data pieces among clusters when the cluster position changes from the previous distribution.

SVM is a binary classification algorithm that creates a hyper plane that separates the data into two classes with the objective to maximize the gap perpendicular to the plane, allowing better generalization. Through the usage of kernel functions (for example, Gaussian), it is possible to generate hyper planes in higher dimensions, when the classes' data are not linearly separable in an original space.

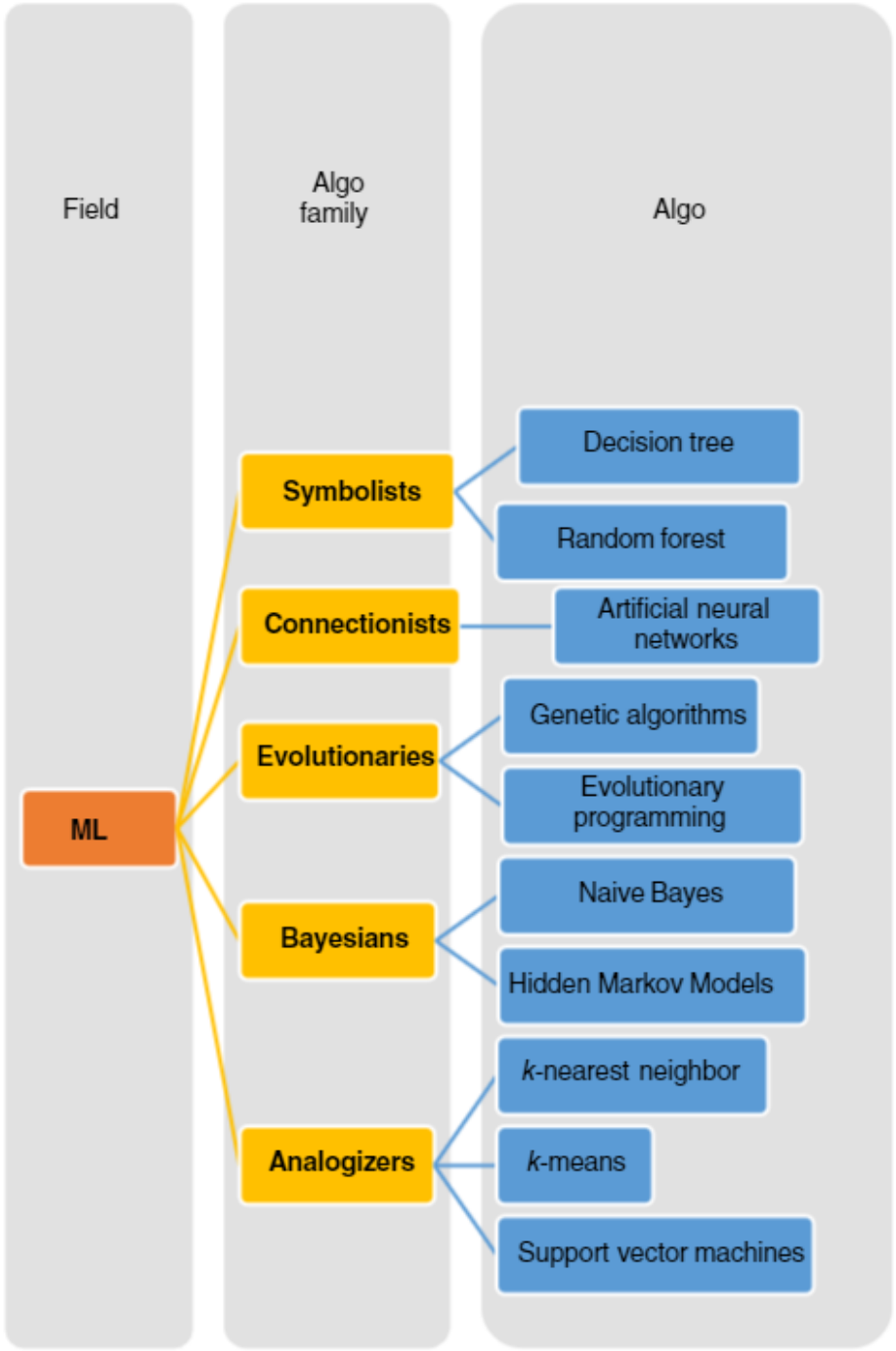


Fig 1.16: ML Algorithm Families and Classification

Another classification proposed by Apruzzese et al. (2018) builds up the algorithms' taxonomy on differentiating between learning approaches. At the first level (see Figure 1.17), they discriminate between more traditional Shallow Learning (SL), in opposition to the more recent DL. SL commonly separates the process of feature extraction from learning itself. Feature extraction is some kind of data dimension reduction or compression that attempts to find out which data attributes are the most important for solving a particular application problem. Commonly, in the SL approach, it is done manually or semi-manually with an assistance of a domain expert, who can perform the critical task of identifying the relevant data characteristics before executing the SL algorithm. DL relies on a multilayered representation of the input-to-output data connections and can perform feature selection autonomously through a process

defined as representation learning. DL algorithms require a large amount of data to understand the data perfectly, so they do not perform as well when the data volumes are small. As the DL algorithm requires many matrix operations, DL relies more on high-performance machines with GPUs than do shallow ML algorithms. Despite using more powerful hardware, commonly a DL algorithm takes a longer time for training because there are many parameters to learn. While the shallow ML approach usually breaks down the problem into multiple subproblems and solves the subproblems, ultimately obtaining the final result, DL advocates direct end-to-end problem solving.

SL and DL approaches can be further characterized by distinguishing between *supervised* and *unsupervised* algorithms. At the next layer, we include some examples of the popular categories of ML algorithms in addition to ones that were presented at the previous classification.

A) **Shallow Learning (SL)**

1) *Supervised learning algorithm examples*

- **Logistic regression** is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression estimates the parameters of a logistic model (a form of binary regression).

2) *Unsupervised SL Algorithms*

- **Clustering** builds up clusters or groups of data pieces that possess similar characteristics. Well-known approaches include k-means and hierarchical clustering. Clustering methods have a limited scalability, but they represent a flexible solution that is typically used as a preliminary phase before adopting a supervised algorithm or for anomaly detection purposes.
- **Association** algorithms aim at identifying unknown patterns between data, making them suitable for prediction purposes. However, they tend to produce an excessive output of not necessarily valid rules; hence, they must be combined with accurate inspections by a human expert.

B) **Deep Learning (DL)**

All DL algorithms in this classification scheme are based on deep neural networks (DNN), which are large ANN organized in many layers capable of autonomous representation learning. Further classification is given in Section 1.8.

Artificial Neural Networks

As ANN represents the most popular ML technique, especially in the field of DL, let us spend a little more time on analyzing it in greater detail.

?	Why ANN is winning the competition? What makes it so good?
!	There could be various reasons for different developers. I believe that most of them are based on facts that ANN is a generic technique that could be employed and has been proved to be successful in multiple application domains with a good base of software and hardware tools available to support an application design and development. On the other hand, it is built on a solid mathematical foundation that has been progressed over a long time with a reasonable expertise available for a future development.

1.8.1 What Is an ANN?

There are many answers to this question. Many definitions of neural networks assume that you already know the basics of such networks, or that you are at least mathematically literate and comfortable with calculus and differential equations. Although mathematical models of computation are valuable, it is easy for not specialists to be intimidated by them. Other definitions tend to be jargon, and still others end up being esoteric or vague. Often, they talk about a new computing architecture inspired by biological models. So, let me give you not one but a few definitions, describing different aspects of the net (Reznik 1999).

The ANN is:

1) A new form of computing, inspired by biological models.
2) A mathematical model, composed of a large number of PEs with specially organized interconnections between them.
3) A computing system, made up of a number of simple, highly interconnected PEs, which processes information by its dynamic state response to external inputs.

?	Which one to apply?
!	All of them. The first one tends to consider an origin of the ANN model. The second one considers ANN as a mathematical model of the object. The third one refers to a structure and a technological implementation.
?	I guess the name was inherited from its biological prototype.
!	You are right! Most of the textbooks on this subject include a description of a biological neuron and consider a human brain as a network of such neurons.

To differentiate the technology from the nature, a corresponding computing model is called an ANN. The ANN is supposed to consist of artificial neurons or PE. These artificial neurons bear only a modest resemblance to their natural cousins.

1.8.2 ANN Architecture

PE manages several basic functions:

1) It evaluates the input signals, determining the strength of each one.
2) It calculates the total value of the output based on inputs and sends it to other PEs.

A PE may have multiple input signals with all of them coming into the PE at the same time.

?	How does a PE calculate it outputs?
!	It depends on the chosen ANN model.

Let us consider a simple, from the computational point of view, case, where the output is calculated as just a weighted sum of the input signals (Figure 1.18a). If we denote the PE's *i*-th input as *xi* and the output as *Y*, then we can write the mapping from the inputs to the output performed by the PE in this model as

$$Y = f(x_1, x_2, \ldots, x_n) = w_1 * x_1 + w_2 * x_2 + \ldots + w_n * x_n$$

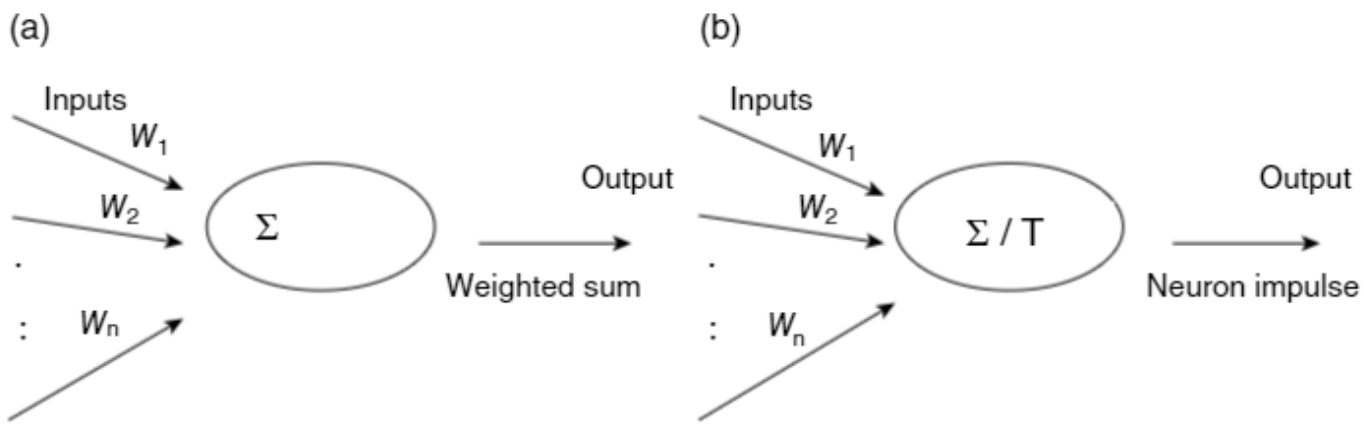


Figure 1.18 Processing element (neuron) with an output determined as (a) weighted sum of the inputs and (b) comparison result of the weighted sum of the inputs against the threshold value *T*.

This function creates the linear model. However, the linear function does not model "firing" behavior of natural neurons. In order to simulate it, another model called perceptron was proposed early. It adds up comparing the weighted sum against the introduced threshold value (Figure 1.18b). As a result of the comparison, a special impulse (an excitation signal) is generated if the weighted sum of the inputs is higher than the threshold value (the neuron is fired).

New effects rise up when PEs are interconnected into a network and some local memory is attached to a PE, so they can store results of previous computations and modify the weights used as we go along. This ability to change the weights allows a PE to modify its behavior in response to its inputs, *or* ***to learn***. For example, suppose a network identifies a dog as "a cat." On successive iterations, connection weights that respond correctly to dog images are strengthened, those that respond to other images, such as cat images, are weakened until the wrong output falls below the threshold level. It is more complicated than just changing the weights for a dog recognition; the weights have to be adjusted so that all images are correctly identified.

?	*How can we connect PEs into a network?*
!	Various ANN models might have different connectivity patterns and topologies.

In the most popular model, called multilayer perceptron (MLP), see Figure 1.19, neurons compose a layer and layers are connected between each other creating an ANN with certain connectivity organization rules to follow up. Suppose we take one processing element and combine it with other neurons to make a layer of these nodes. Inputs could be connected to many nodes with various weights, resulting in a series of outputs, one per node. Carrying the design yet another step, we can interconnect several layers. The layer that receives the inputs is called the input layer. It typically performs no function other than buffering of the input signal. The network outputs are generated from the *output layer*. Any other layers in the middle are called hidden layers because they are internal to the network and have no direct contact with the external environment. Sometimes, they are likened to a "black box" within a network system. But just because they are not immediately visible does not mean one cannot examine what goes on in those layers. There may be from zero to several hidden layers. The connections are multiplied by the weights associated with that particular node with which they interconnect. Note that there are many more connections than nodes. The network is said to be fully connected if every output from one layer is passed along to every node in the next layer.

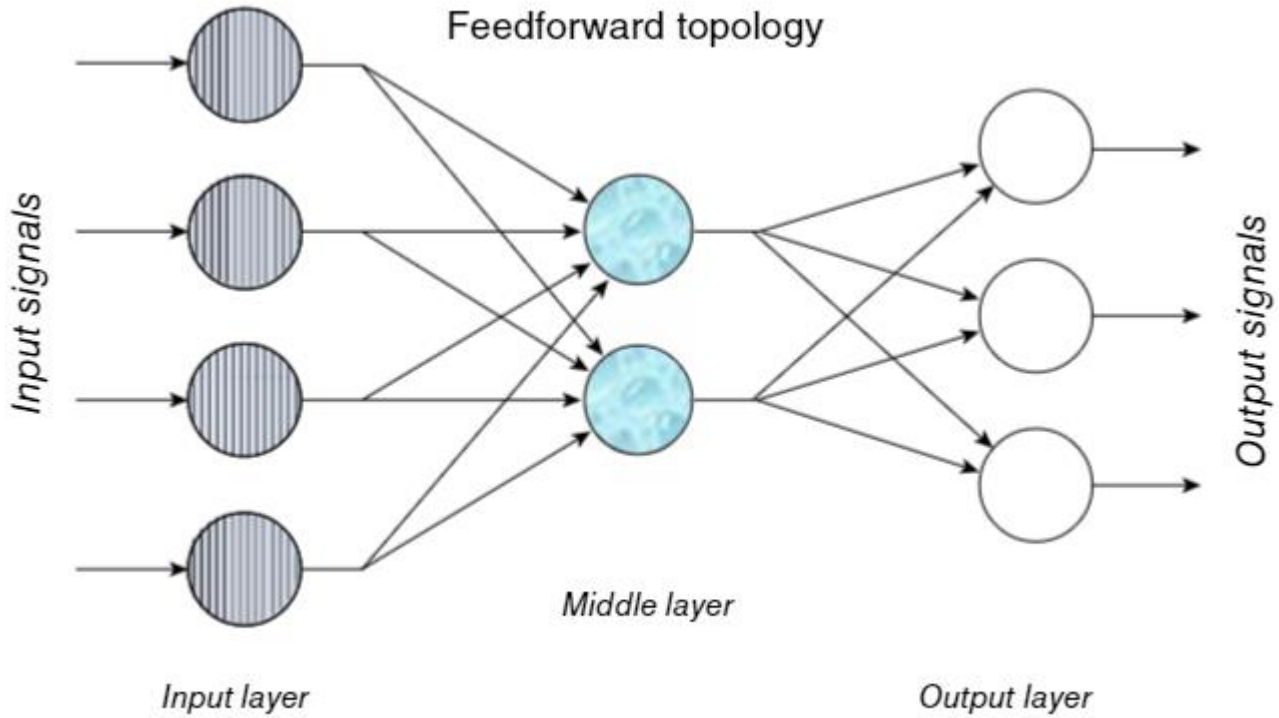


Figure 1.19 Multilayer perceptron topology.

Since each PE is a nonlinear block, such an arrangement of PEs results in a highly nonlinear network. Also, as one can see in this model all connections are feedforward going from left (inputs) to right (outputs). There are no backward loops, nor connections between neurons belonging to the same layer in this model.

ANN learning or training is achieved through the process of changing weights. Each neuron output is connected to the neuron input in the following layer, and a training algorithm is employed to update the input weights and optimize the logistic fit to the incoming data. As a part of a neural network layer, multiple parallel neurons initiated with different weights learn on the same input data simultaneously. In the application of multiple layers of multiple nodes, each node learns from all the outputs of the previous layers, stepwise reducing the approximation of the original input data to provide an output representation set.

1.8.3 ANN Classification

?	*Can the nodes be connected by any way? Does any order exist?*
!	It depends on the type of the particular network.

Actually, a lot of different ANN models have been proposed. Some of them are given in Figure 1.20, modified from Reznik (1997). This list is definitely not complete. Some of the topologies, which are not included, are presented in this section. On the basis of the connection types, ANN can be divided into feedforward and recurrent. If the output of each node propagates from the input side (left) to the output side (right) unanimously, then the ANN is called feedforward (for example, see Figure 1.19). If there exist any link between nodes directed from the right to the left, this ANN has a recurrent type (Figure 1.21).

The life of any supervised ANN includes two phases: learning and application. Before one starts applying an ANN, it should be trained in a supervised mode. The training procedures are determined by a learning regime and can be quite different from each other. Generally, the goal of a neural network is to nudge the weights in the direction that will minimize the difference. Unsupervised ANN can be employed straightaway with no prior training required.

Various algorithms have been developed for training supervised ANNs. The most popular one is the back-propagation algorithm. When a training input–output example is presented to the system,

the back-propagation algorithm computes the system output and compares it with the desired output of the training example. The error is propagated backward through the network from the output layer to the input layer. The connection weights are modified as the error is propagated from the output layer neurons backward to the input layer.

Unsupervised learning (i.e. without a "teacher"):

Feedback Nets:

Additive Grossberg (AG)
Shunting Grossberg (SG)
Binary Adaptive Resonance Theory (ART1)
Analog Adaptive Resonance Theory (ART2, ART2a)
Discrete Hopfield (DH)
Continuous Hopfield (CH)
Discrete Bidirectional Associative Memory (BAM)
Temporal Associative Memory (TAM)
Adaptive Bidirectional Associative Memory (ABAM)
Kohonen Self-organizing Map (SOM)
Kohonen Topology-preserving Map (TPM)
Recurrent Neural Networks (RNN)
Long Short Term Memory (LSTM)

Feedforward-only Nets:

Learning Matrix (LM)
Driver-Reinforcement Learning (DR)
Linear Associative Memory (LAM)
Optimal Linear Associative Memory (OLAM)
Sparse Distributed Associative Memory (SDM)
Fuzzy Associative Memory (FAM)
Counter-propagation (CPN)
Autoencoders (AE)
Stacked Autoencoders (SAE)
Deep Belief Neural Networks (DBNN)

Supervised learning (i.e. with a "teacher"):

Feedback Nets:

Brain-State-in-a-Box (BSB)
Fuzzy Cognitive Map (FCM)
Boltzmann Machine (BM)
Mean Field Annealing (MFT)
Recurrent Cascade Correlation (RCC)
Learning Vector Quantization (LVQ)

Feedforward-only Nets:

Perceptron
Multilayer Perceptron (MLP)
Modified Time-Based Multilayer Perceptron (MTBMLP)
Radial Basis Function (RBF)
Adaline, Madaline
Backpropagation (BP)
Cauchy Machine (CM)
Adaptive Heuristic Critic (AHC)
Time Delay Neural Network (TDNN)
Associative Reward Penalty (ARP)
Avalanche Matched Filter (AMF)
Backpercolation (Perc)
Artmap
Adaptive Logic Network (ALN)
Cascade Correlation (CasCor)

Supervised and unsupervised learning

Convolutional Neural Networks (CNN)
Convolutional Transpose Neural Networks (CTNN)

1.8.3.1 Supervised Learning Topologies

Fully connected feedforward deep neural networks (**FNN**) are classical ANN models, where every neuron is connected to all the neurons in the previous layer. This topology does not include any feedback loop. Commonly, the architecture consists of an input layer, multiple hidden layers, and an output layer with a tuned set of hyper-parameters. At the execution stage, an information flows in one direction only: feedforward from the inputs to the outputs. They provide a flexible and general-purpose solution for classification and regression problems. Shallow FNN commonly have not more than three to five layers. DNN have a larger number of layers that results in a high resource consumption, especially during the training phase.

Convolutional feedforward deep neural networks (CNN) are a variant of DNN, where each neuron receives its input only from a subset of neurons of the previous layer. This characteristic makes CNN effective at analyzing spatial data, but their performance decreases when applied to nonspatial data. CNN have a lower computation cost than FNN. As Sirinam (2019) describes, CNNs have become very popular, especially in image classification, after Krizhevsky et al. (2012) won the Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Convolution is embedded by the use of the weighted average operation to smoothen noisy data. During the execution, the CNN performs a number of convolutions to yield multiple numbers of linear activation functions. In CNN architecture (see Figure 1.22), layers are divided into two groups with the first one responsible for feature extraction and the second one for classification itself. This topology allows performing feature extraction automatically. In feature extraction, the input is first fed into a convolutional layer, which consists of a set of filters. Each input type is convolved through a specialized set of filters, essentially by taking the dot product of the two vectors, to get an intermediate set of values. Depending on the number of features to be extracted, the number of filters could vary. The output of the activation function is then fed into a pooling layer. The pooling layer progressively reduces the spatial size of the representation from the feature map to further reduce the number of parameters and required computation. The most common approach used in pooling is Max Pooling that simply selects the maximum value in a spatial neighborhood within a particular region of the feature map to build a representation of the data. This method has the advantage of being invariant to small transformations, distortions, and translations of the input, since the largest signals in each neighborhood are retained.

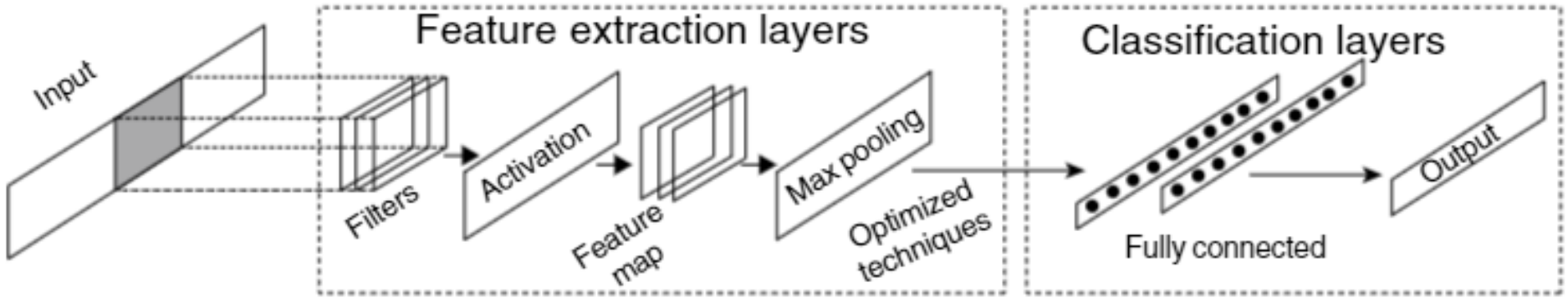


Figure 1.22 A basic architecture of convolutional neural networks (CNN). *Source*: Courtesy of P. Sirinam (2019).

The final part of the feature extraction component (Optimized techniques in Figure 1.22) mainly consists of a stochastic dropout function and Batch Normalization that help improve a classifier performance and prevent overfitting. The CNN then passes the output from the convolutional and pooling layers, which represents high-level features of the input, into the Classification component. In this component, a set of fully connected layers use the features to classify the input. During training, the loss value of classification is applied to update not only weights in the classification component but also the filters' values in feature extraction.

Recurrent Deep Neural Networks (RNN) architecture (Figure 1.21) includes the feedback loop that connects its output to the inputs. The topological structure makes the model independent from the input dimension and allows for processing of any length inputs. This architecture is commonly used to analyze and predict time series outcomes and various signals. RNNs are trained to predict the future development based on the previous history. Long Short Term Memory (LSTM) is a special type of RNN having memory cells that maintain information in memory for a longer period, which also overcomes the standard RNN problem of vanishing gradient. LSTM has inputs, outputs, and a forget gate making it a powerful architecture to deliver better results that comes at an expense of a high computational cost. The LSTM model is commonly trained by using the back-propagation technique.

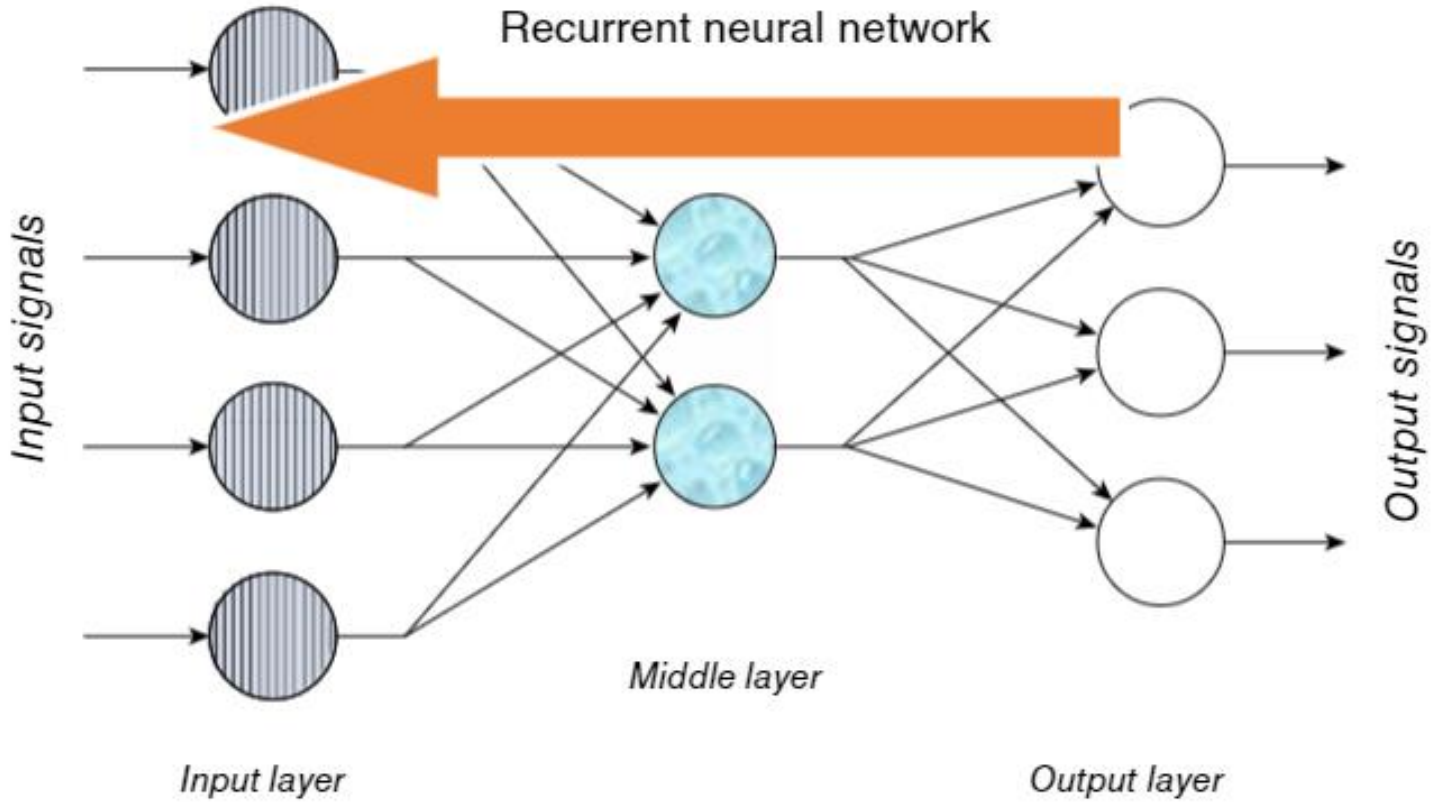


Figure 1.21 Recurrent neural network topology (see the feedback loop arrow added up).

1.8.3.2 Unsupervised Learning Topologies

- **Deep Belief Networks (DBN)** are modeled through a composition of *Restricted Boltzmann Machines* (RBM), a class of neural networks with no output layer. DBN can be successfully used for pretraining tasks because they excel in the task of feature extraction. They do require a training phase, but with unlabeled datasets.
- **Stacked Autoencoders (SAE)** are composed of multiple *Autoencoders*, a class of neural networks where the number of input and output neurons is the same. SAE are good at pretraining.

As Sirinam (2019) pointed out, Vincent et al. introduced SAEs in 2010 to improve classification performance in recognizing visual data. SAE leverages the concept of an autoencoder (AE), a simple neural network with input, hidden, and output layers (Figure 1.23). In AE, the input data is first encoded, passing it through a layer of neurons to a more condensed representation (the hidden layer). The AE then performs decoding, in which it attempts to reconstruct the original input from the hidden layer while minimizing the error.

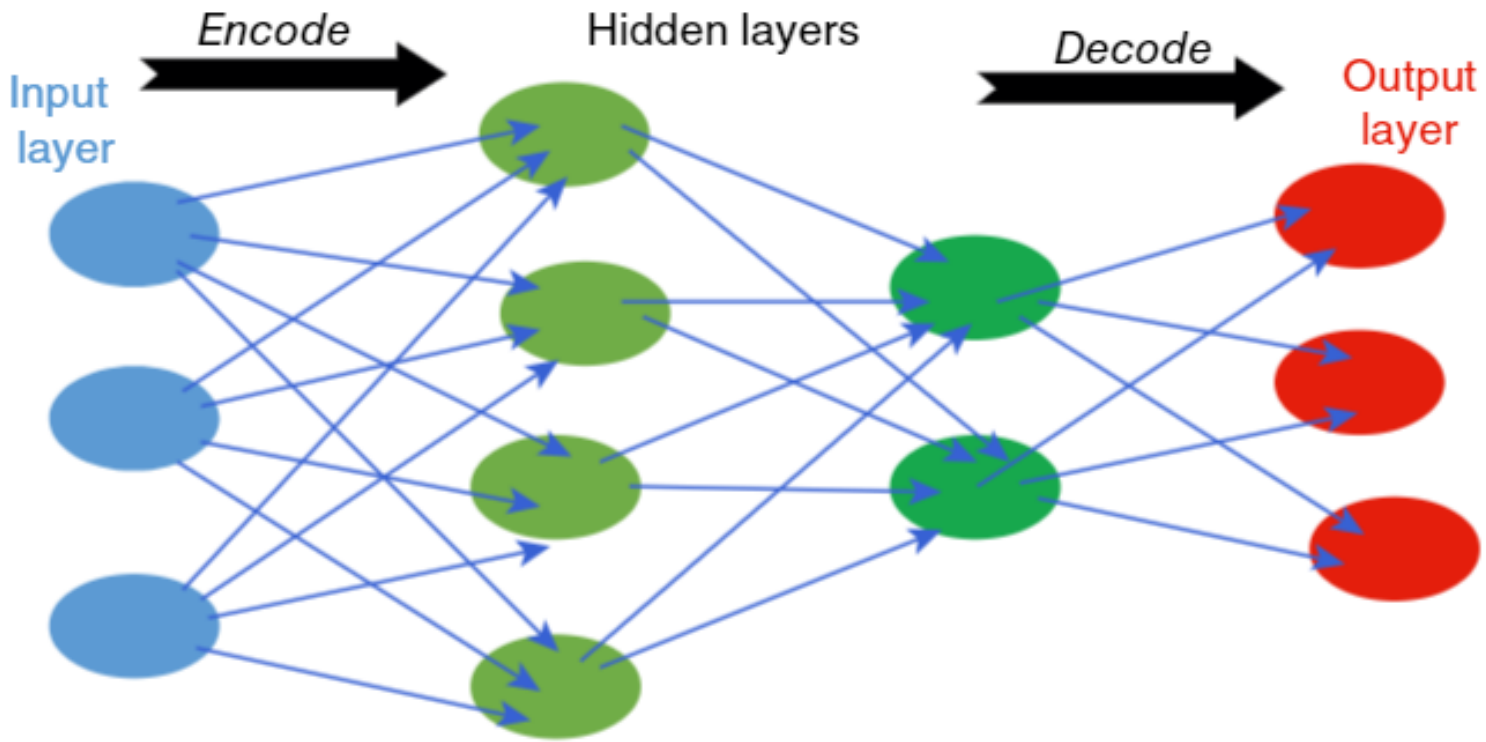


Figure 1.23 Autoencoder structure.

Genetic Algorithms

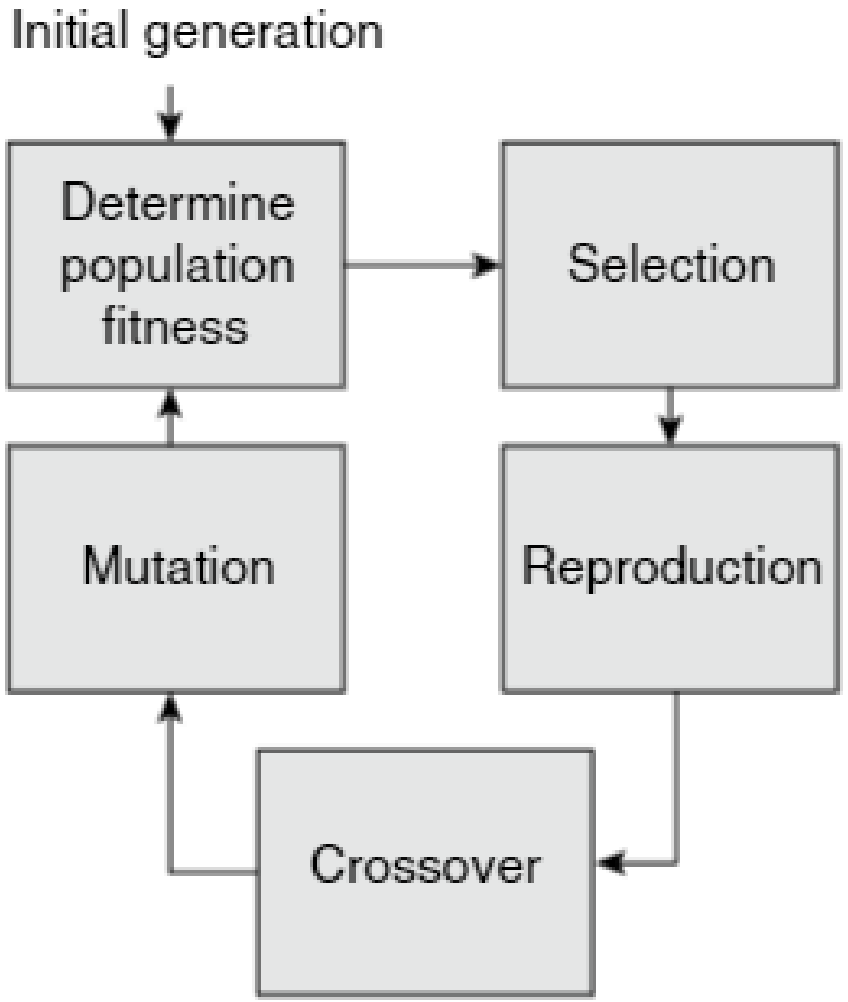


Figure 1.24 Genetic algorithm operations.

GA method imitates natural evolution processes. It includes a number of operations called *reproduction, crossover, and mutation* (Figure 1.24). A conventional GA has four main features: Population size, Reproduction, Crossover, and Mutation. *Reproduction* is a process, in which a new generation of population is formed by the stochastic selection of individuals from an existing population, based on their fitness. This process results in individuals with higher fitness values obtaining more copies in the next generation, while low fitness individuals receive less; for this reason, it is sometimes referred to as a survival of the fittest test. *Crossover* represents, perhaps, the most dominant operator in most GAs, and is responsible for producing new trial solutions "on-line." Under this operation, two or more individuals are selected to produce new offspring by exchanging portions of their structures. These offspring may then replace the existing trials. *Mutation* is a localized, bitwise operator, which is applied with a very low probability, typically 0.001 per bit or less. Its role is to alter the value of a random position in a gene

An offshoot of GA is genetic programming (GP). The aim of GP is to automatically generate correct computer programing codes from a population of randomly generated computer programs. In an era of escalating software development costs, it is reasonable to look for automatic methods of generating correct computer programs. GP's performance on an initial set of test problems has been encouraging.

Algorithm 1.3 Generic Genetic Algorithm Procedure

Generate an initial random population of trials, $P(0) = A_n(0)$, $n = 1, ..., N$, where N is the population size;

For successive sample instances t:

Evaluate the performance of each trial, $f(A_n(t)), t = 0, 1, 2, ...$;

Select one or more trials to produce offspring by taking a sample of $P(t)$ using the probability distribution $m(A_n(t)) = m(A_n(t))/\sum m(A_i(t))$ – selection algorithms may vary depending on the implementation;

Apply one or more of the genetic operators to the selected trials in order to produce new offspring, $A^\circ_m(T)$, $m = 1, ..., M$, where M is the number of offspring which is usually equal to the number of selected parent trials;

Form the next generation of population, $P(t + 1)$ by selecting $A_i(t) \in P(t)$, $i = 1, ..., M$ to be replaced by the offspring, $A_j(t)$, $j = 1, ..., M$. The criterion for selecting which trials should be replaced may be random, on the basis of the least fit or some other fitness basis;

Terminate the GA process after a prespecified number of generations is complete or on the basis of a criterion, which determines convergence of the population.