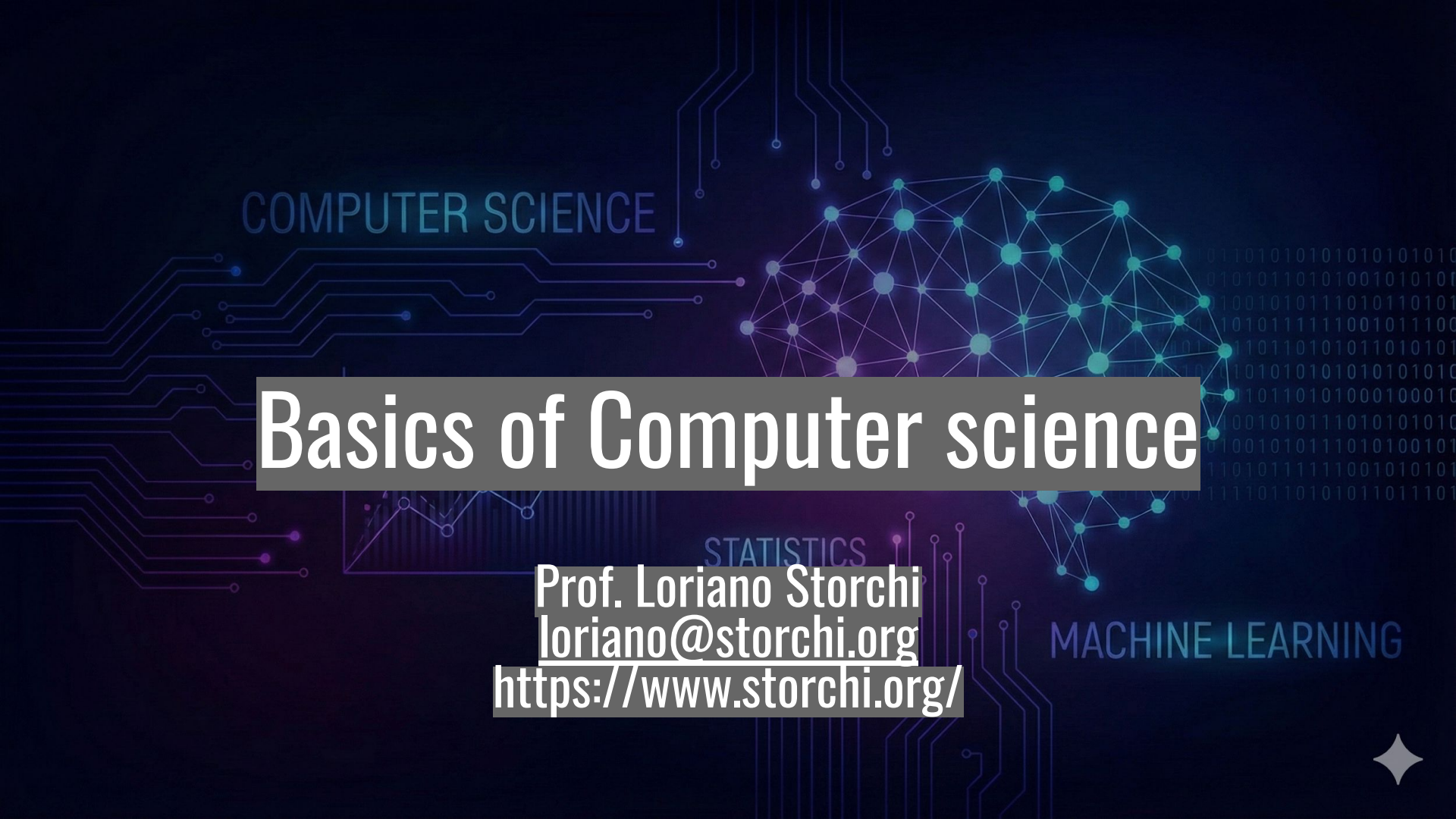



# Basics of Computer science

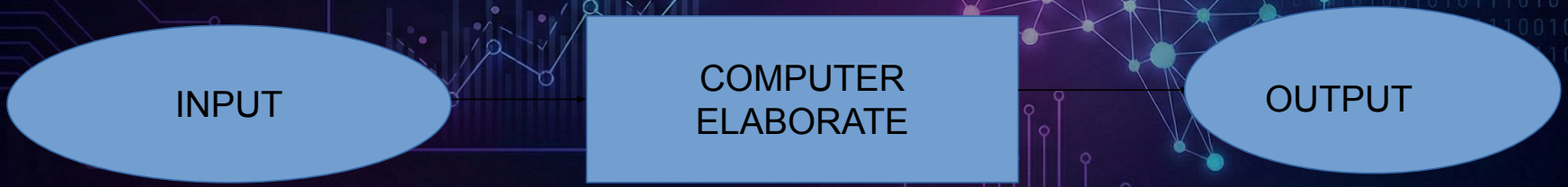
Prof. Loriano Storchi
loriano@storchi.org
https://www.storchi.org/

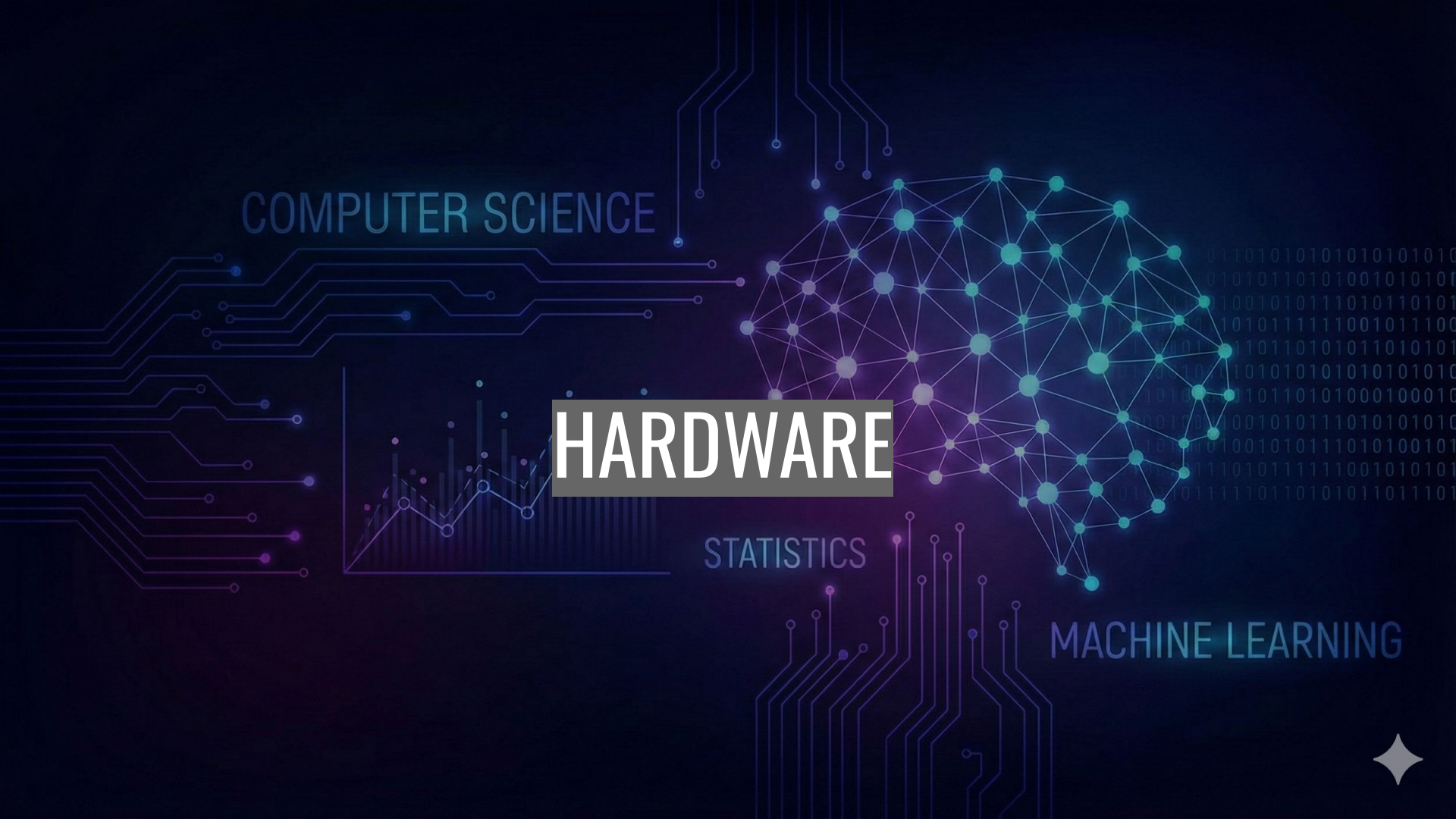
# Contents


- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption

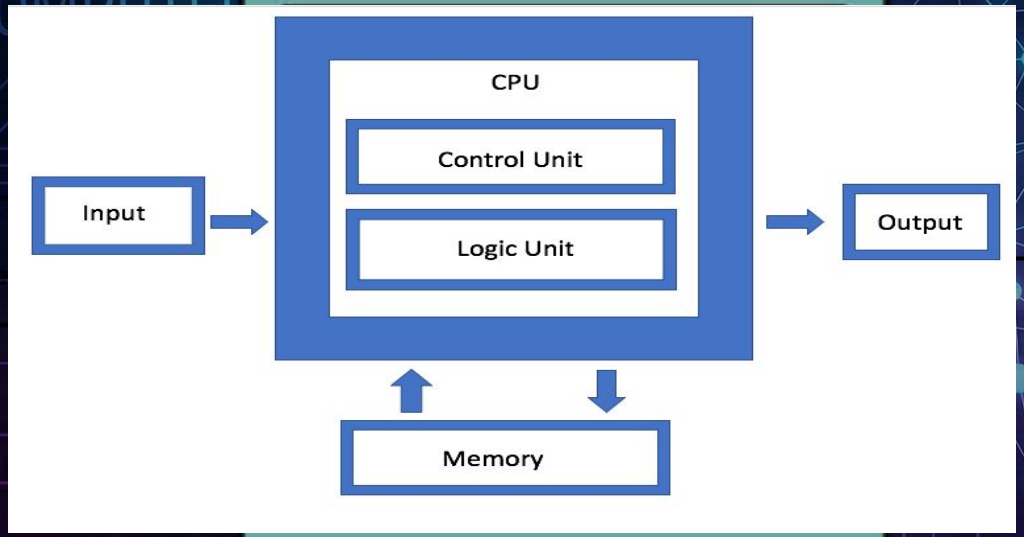
# Informatics

We can use different definitions such as: computer science is the science of representing and processing information, paraphrasing the study of algorithms that describe and transform information.




COMPUTER SCIENCE
HARDWARE
STATISTICS
MACHINE LEARNING

# Von Neumann machine (Zuse)




SYSTEM BUS, connection (Harvard architecture separation between data memory and program memory)

# Von Neumann machine (Zuse)

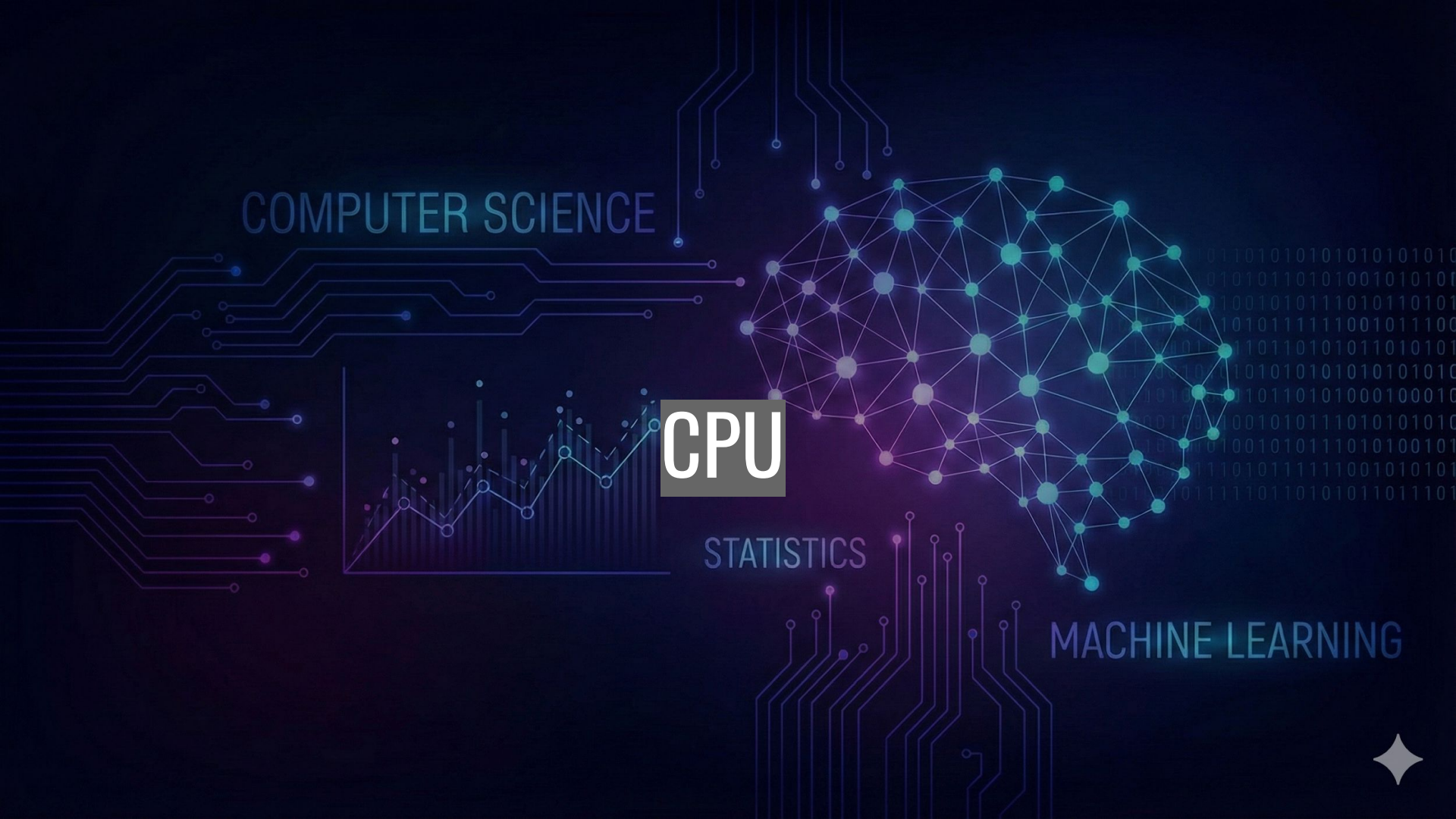
The architecture described by Von Neumann (Zuse) is therefore composed of:

- The CPU is therefore a central processing unit
- A memory device that is used to store data which can then be identified by its address.
- The various Input Output (I/O) devices that are used to interact with the external user or other systems
- An interconnection line that links the various subsystems, the bus

# Von Neumann machine (Zuse)

A computer constructed in this way is an extremely flexible machine. The hardware provides basic functionality, while the software will then specialize the machine to perform specific tasks.

Below we will present the various basic components of the calculator with a “modern” perspective


COMPUTER SCIENCE
CPU
STATISTICS
MACHINE LEARNING

# CPU

CPU (Central Processing Unit) coordinates and manages all the various hardware devices to acquire, interpret, and execute program instructions.

**Today, it consists of a single chip** and, like any other chip, **communicates with the outside world via pins** . Using these pins, it receives signals and sends electrical signals (information consisting of sequences of bits).

# CPU - CU

The Control Unit (CU) is a core component of the CPU that directs the operation of the processor.

- Think of it as the conductor of an orchestra. It doesn't play the instruments (the ALU does the calculation), but it tells every other component when to play and what to play.
- It manages the flow of data between the CPU, memory, and input/output devices.
- Without the CU, the powerful ALU and fast Registers would be silent and inactive.

# CPU - ALU

**ALU** (Arithmetic Logic Unit) performs logical and arithmetic operations.

- Inputs: Operands (A and B) and an Opcode (Operation Code).
- Outputs: The Result (Y) and Status Flags.
- Role: It acts as the "calculator" of the CPU. Without it, the computer cannot process data, only move it.

It is often represented by a "V" shape symbol in architectural diagrams.

# CPU - ALU

## Core Operations

PUTER SCIENCE

### ARITHMETIC

These operations handle numerical calculations. The complexity depends on the ALU design.

- **ADD / SUB:** Basic integer addition and subtraction.
- **INC / DEC:** Increment or decrement by 1.
- **MUL / DIV:** Often handled by specialized hardware in modern CPUs, but basic ALUs may use repeated addition/subtraction.

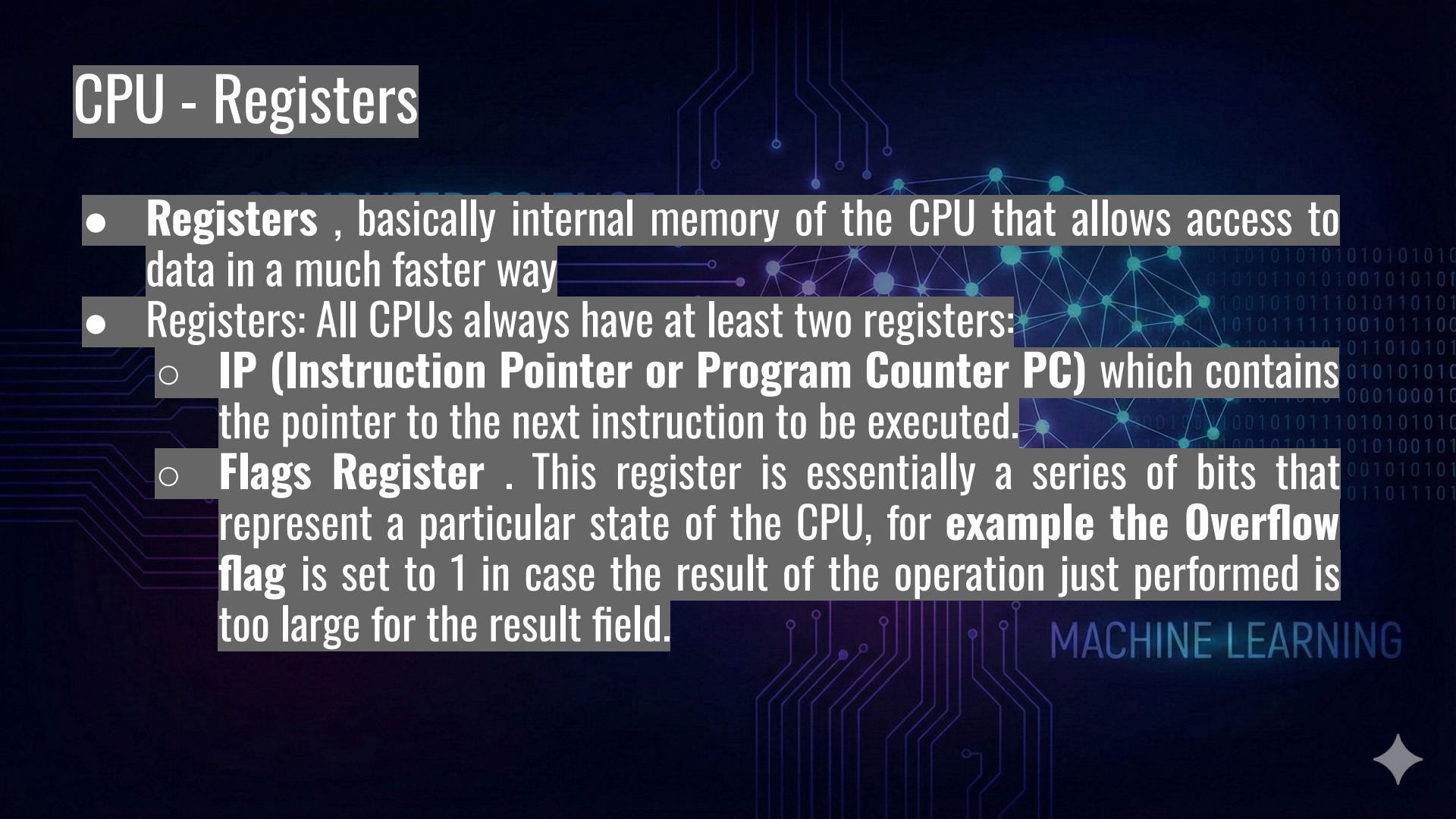
### LOGIC

These operations treat data as individual bits rather than numbers, essential for decision making.

- **AND:** Used for masking (clearing specific bits).
- **OR:** Used for setting specific bits.
- **XOR:** Used for toggling bits or comparing values.
- **NOT:** Inverts all bits (1s Complement).

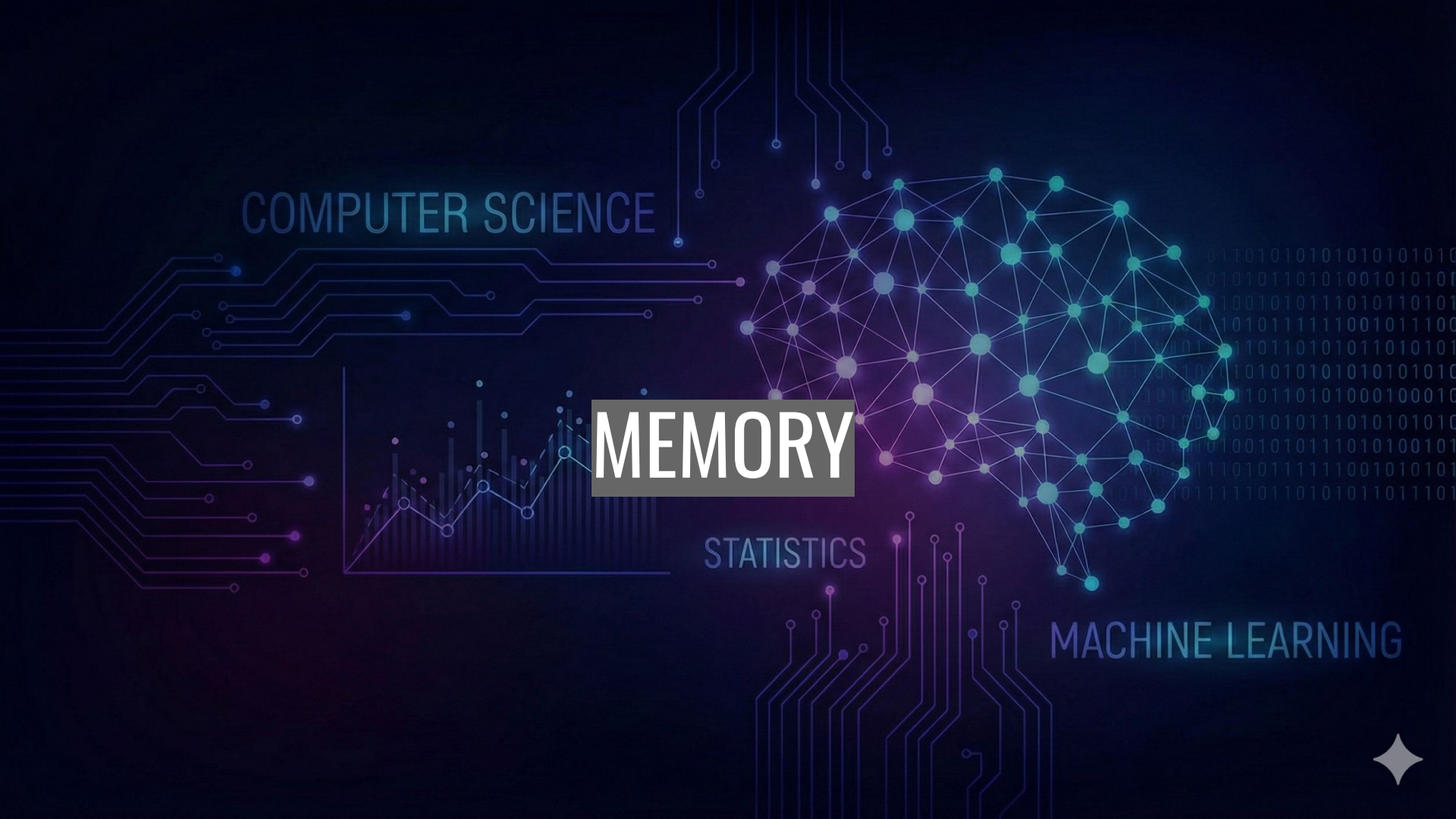
# CPU - Registers

- **Registers** , basically internal memory of the CPU that allows access to data in a much faster way
- Registers: All CPUs always have at least two registers:
  - **IP (Instruction Pointer or Program Counter PC)** which contains the pointer to the next instruction to be executed.
  - **Flags Register** . This register is essentially a series of bits that represent a particular state of the CPU, for **example the Overflow flag** is set to 1 in case the result of the operation just performed is too large for the result field.



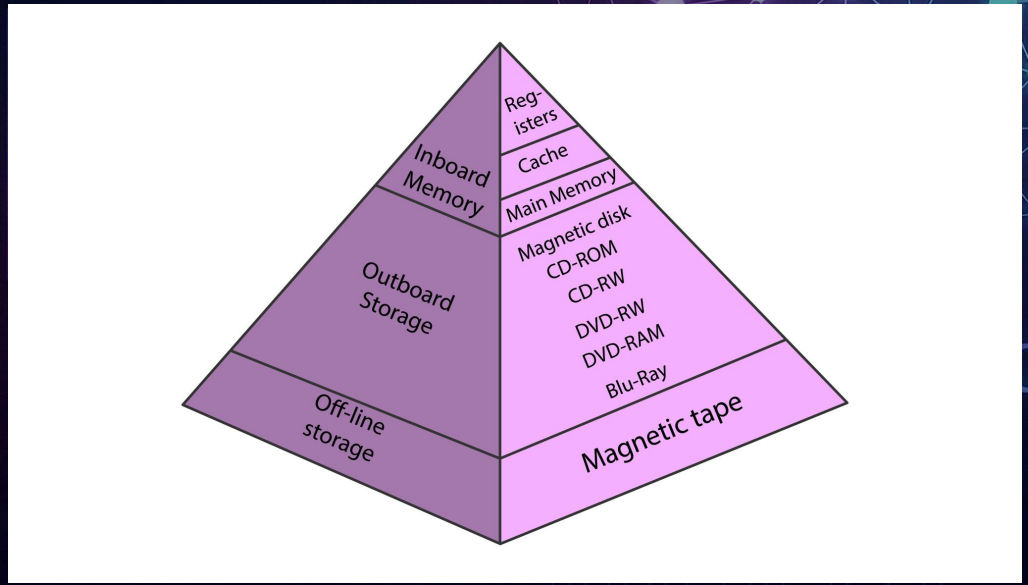
# CPU - CLOCK

- **CLOCK** : marks the time intervals in which the CPU's internal devices operate. It determines their **speed expressed as the number of intervals per unit of time** .
- **The state of the CPU changes every time a pulse is sent.** So the execution time of a given operation is measured in number of clock cycles.
- An important part of the CPU is the series of “circuits” that serve to propagate this impulse between all the components of the CPU.
- No CPU can operate faster than the time it takes for the clock signal to travel the longest path of this signal distribution “circuit,” **the critical path .**


COMPUTER SCIENCE
MEMORY
STATISTICS
MACHINE LEARNING

# Memory hierarchy

The memory hierarchy in current processors is composed of several levels, each one characterized by data access speeds inversely proportional to their size: the larger these areas, the longer it takes to retrieve the data contained within them.



# Registers (The CPU's "Hands")

- Location: Inside the CPU itself.
- Speed: The absolute fastest (instant access).
- Capacity: Extremely small (only holds a few bits of data, like the specific numbers currently being added).
- Role: These are not usually considered "memory" in the storage sense; they are the actual working parts of the processor.

# Cache

**Cache: Generally, one to three levels of cache memory** are installed. Cache memory has different access times and sizes, depending on whether it is on-chip or off-chip, and on the technology used to build the memory cells. (Cat /proc/cpuinfo to see cache size)

The use and benefit of a cache is based on the principle of locality, meaning that a program tends to reuse recently used data and instructions. A consequence is a rule of thumb that typically states that 90% of the total time is spent executing 10% of the instructions. To be more precise, using a cache is advantageous when a code exploits both spatial and temporal locality of the data.

# Cache

**Temporal Locality**

"If you used it recently, you'll likely use it again soon."

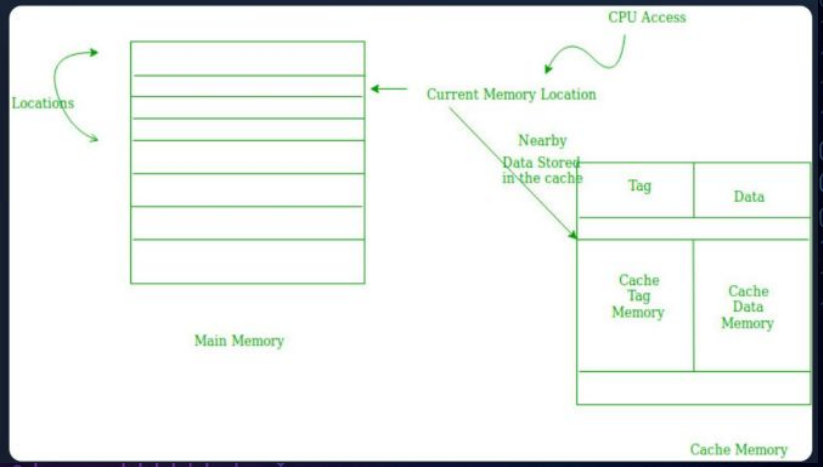
Impact: The cache keeps recently accessed data (variables, instructions) ready for quick reuse.

Example: A counter variable inside a for loop.

**Spatial Locality**

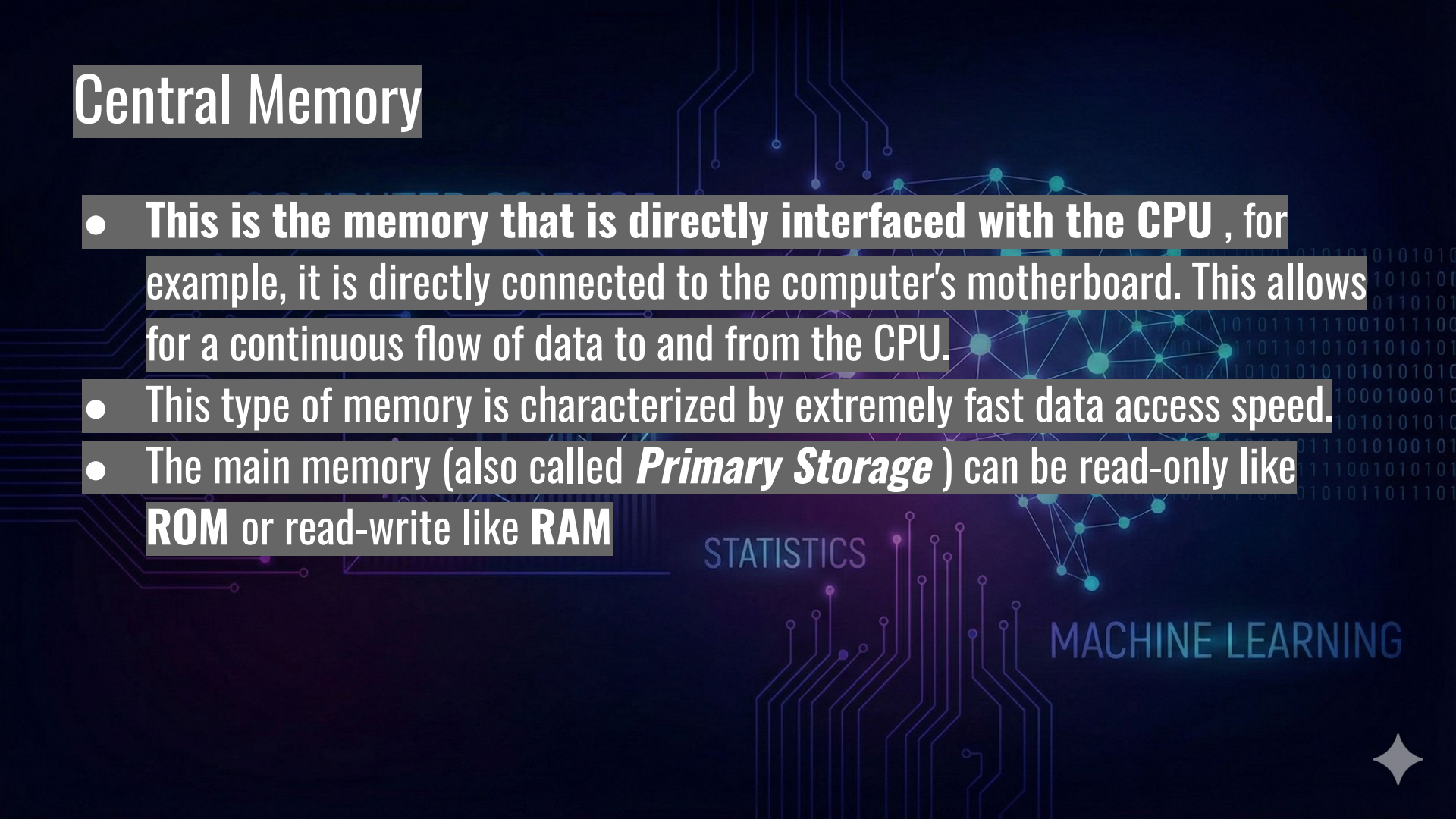
"If you used this address, you'll likely use its neighbors."

Impact: When fetching data, the cache pulls in the entire "block" or "line" of surrounding memory.



# Central Memory

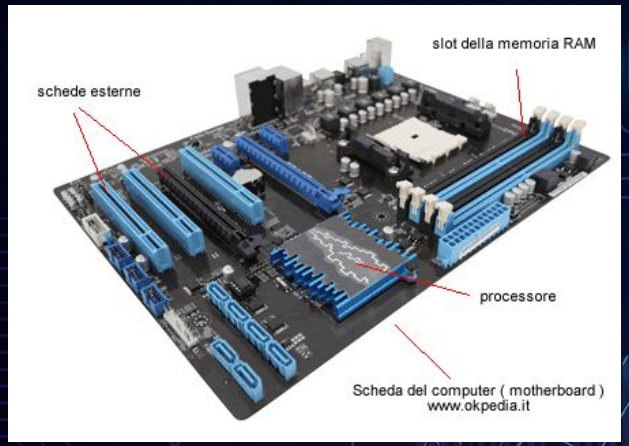
- **This is the memory that is directly interfaced with the CPU** , for example, it is directly connected to the computer's motherboard. This allows for a continuous flow of data to and from the CPU.
- This type of memory is characterized by extremely fast data access speed.
- The main memory (also called ***Primary Storage*** ) can be read-only like **ROM** or read-write like **RAM**



# Central Memory - RAM

- **RAM (Random Access Memory)** is divided into memory cells, and **each cell is assigned an address** that is used to read and write the data it contains.
- RAM **also contains the instructions (opcodes)** that will be executed and the data on which these instructions will operate.
- Characteristics:
  - **Volatile** : The content is lost when the computer is turned off.
  - **Fast** : Approximately 100 clock cycles. Fast, therefore expensive.
  - **Small size** compared to mass memory, order of a few GiB

# Central Memory - RAM

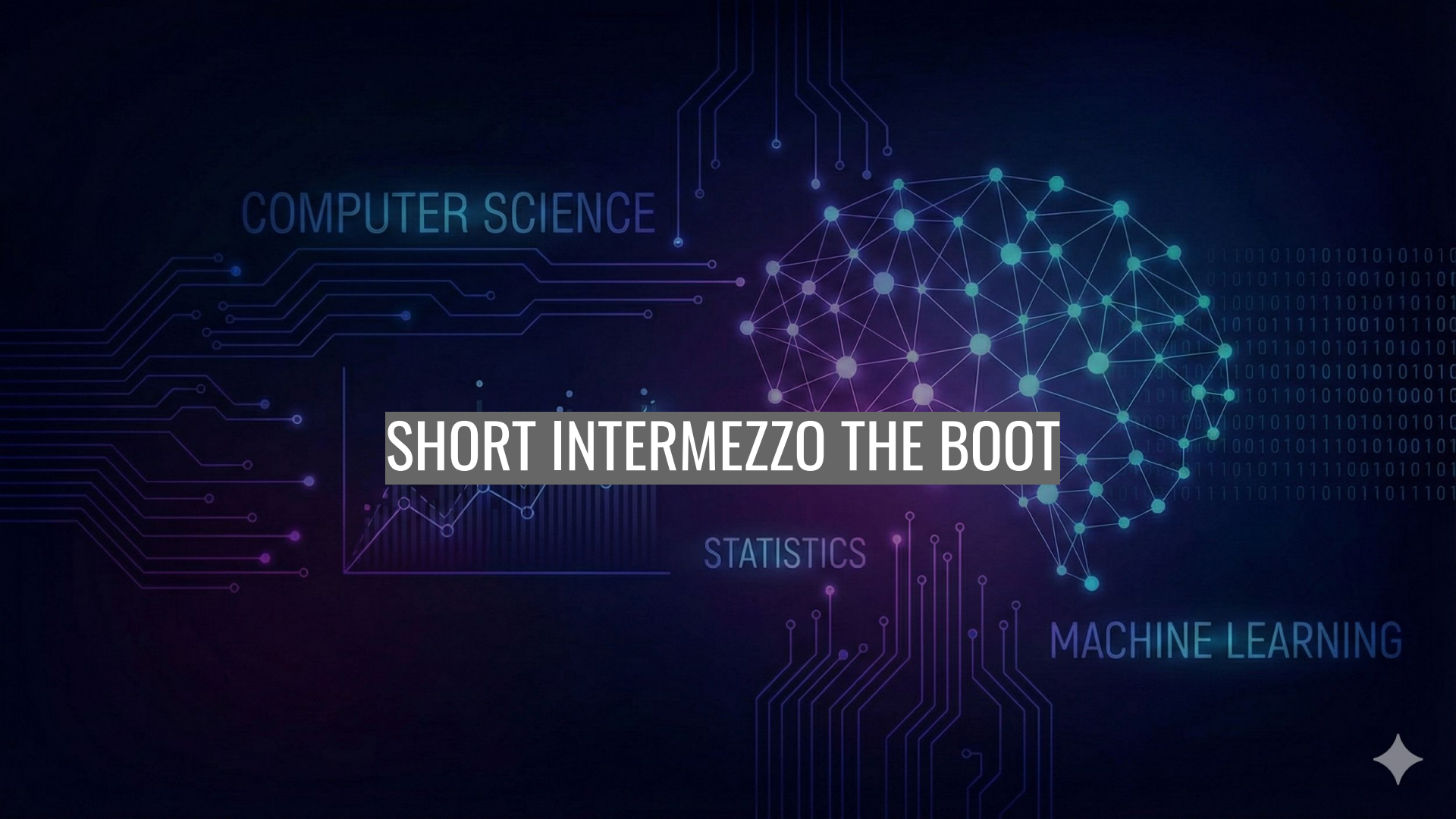



There are several RAM technologies, the most commonly used one nowadays is **DRAM (Dynamic RAM)**
In practice, each bit is stored in a capacitor ( **bit 1 or 0 depends on the charge of the capacitor** ). Each capacitor must be periodically recharged, otherwise there would be a loss of charge and therefore of information. There are then several technologically different variants of DRAM.

**SRAM (Static RAM)** is a memory technology that doesn't require continuous refresh, but can retain information for very long periods of time. It offers **low power consumption and short access times, but is expensive to build. It's generally used for cache memory.**

# Central Memory - ROM

- **ROM (Read Only Memory)** read-only memory with a higher access speed than mass storage
- **Allows you to store data permanently** . This type of memory is used to store data and code necessary for the computer startup procedure and other programs/procedures ( **Firmware** ).
- **BIOS Basic Input/Output System** , in modern systems replaced by **UEFI Unified Extensible Firmware Interface**
- **EPROM Erasable Programmable Read Only Memory** , or erasable programmable read-only memory. They can be erased and rewritten a generally limited number of times.

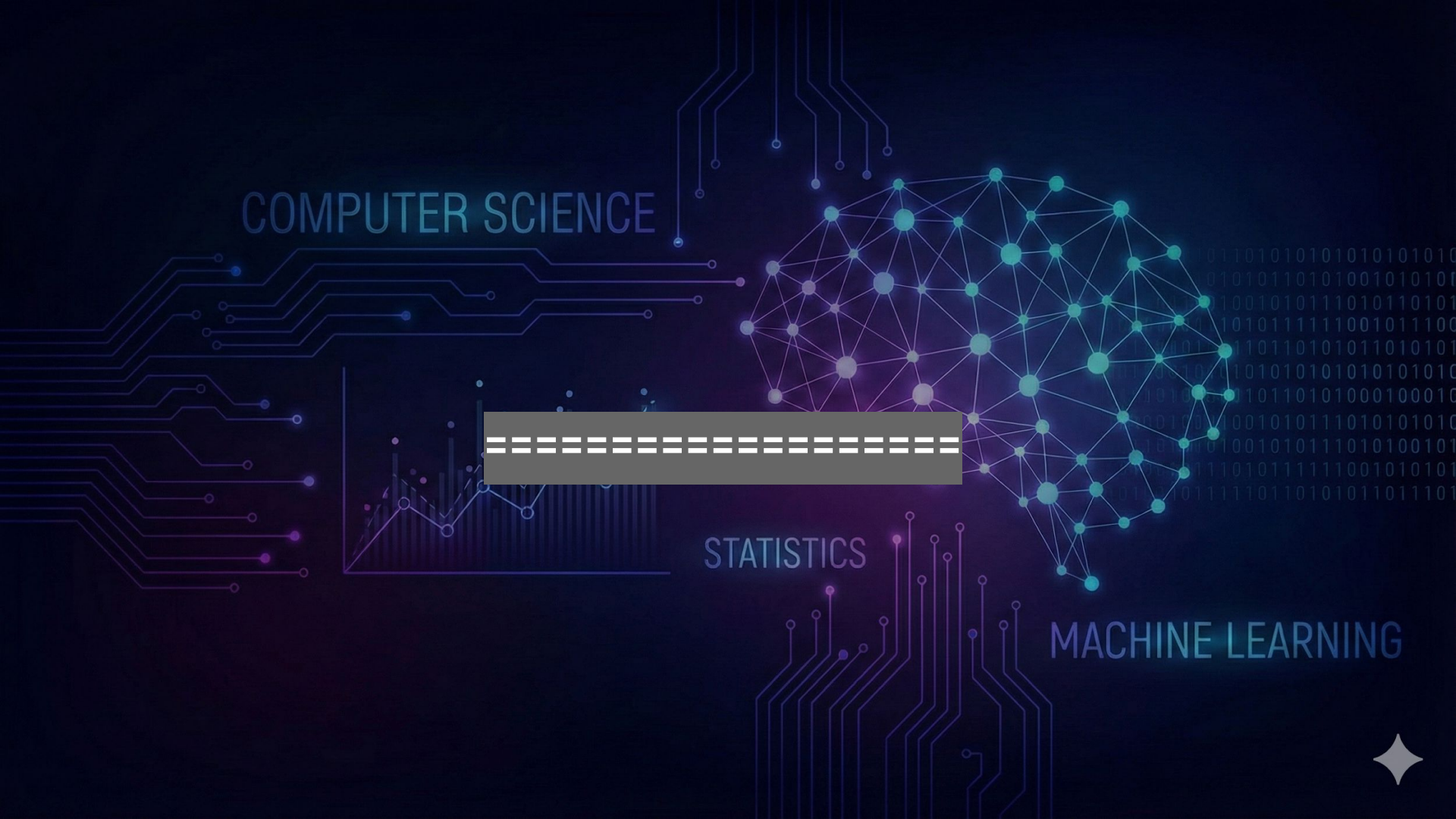

COMPUTER SCIENCE
SHORT INTERMEZZO THE BOOT
STATISTICS
MACHINE LEARNING

# BOOTUP process

- **Power-on** : Obviously the first stage of the process is the power-on, which is usually initiated by the user.
  - **wake on LAN "** setting has been added , which makes it easy to power on over the network, so no physical presence or action from the user is required.
  - As soon as the CPU is powered it executes the code that is located in the **ROM** (read-only memory) on the motherboard.

# BOOTUP process

- **POST** : The system then runs a procedure called POST ( **Power-On Self Test** ), which ensures that all the hardware is operational and ready for use. This includes checking **the memory and hard drives and...** Once POST is complete, the system looks for the first device in the boot order list.
- At this point, after **loading the BIOS or UEFI**, the system searches for an active device in the boot device list. When it finds an available device, the **BIOS/UEFI** provides information about basic communications with peripherals and communications on the motherboard itself.


COMPUTER SCIENCE
==========
STATISTICS
MACHINE LEARNING

# Mass Storage – Secondary Storage

- **Unlike main memory, it is not directly accessible from the CPU, but the CPU communicates with a controller (I/O bus)**
- Data is transferred from secondary memory to an area in main memory and read from there directly by the CPU.
- Physically these devices are connected to the motherboard with a high-speed cable or via cables connected to external interfaces.

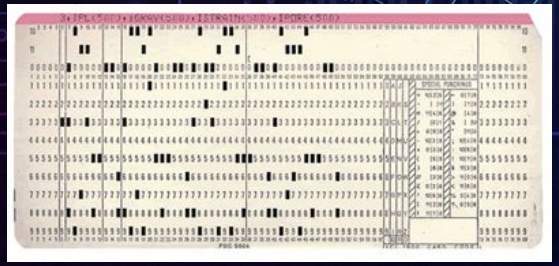
They can be made with magnetic, optical or solid-state technology

# Mass storage

- Consisting of Hard Drives, CDs, DVDs, Solid State Drives, Tapes
  - **Non-volatile** , so data remains stored when the computer restarts
  - **Slow compared to RAM** (> 10000 cycles), obviously very variable depending on the type of support
  - **Averagely cheaper** or at least cheaper than DRAM or SRAM
  - **Large in size** (now hundreds of GiB or a few TiB)
  - **They can be either read-only or read-write** (think of HARD-Disks and DVD-ROMs)

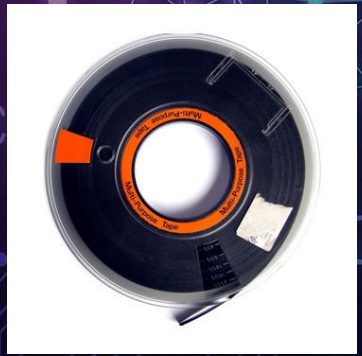
# Mass Storage – The Early Days

- Punched cards are the first secondary memory in the history of computers. Data is stored on cardboard cards and recorded by punching holes, and then read by the computer.
- Punched tapes similar to punched cards in type



# Mass storage – Magnetic tape

- In this case the data is stored on a tape in magnetic form. These were historically used in mainframes (large computers capable of performing very fast processing and storing large amounts of data) in the 70s and 80s, but **are also used today for data backup.**

# Mass Storage – Hard Disk

- **Composed of several magnetic plates** placed one on top of the other and rotating
- Data is stored on both sides of the individual platters and is organized into **tracks and sectors.**
- The **heads** are used to read and write data.
- The rotation speed of the platters ( **rpm** ) is one of the determining factors in the reading and writing speed.
- **SSD solid state drives** lower latencies less susceptible to failures and mechanical shocks

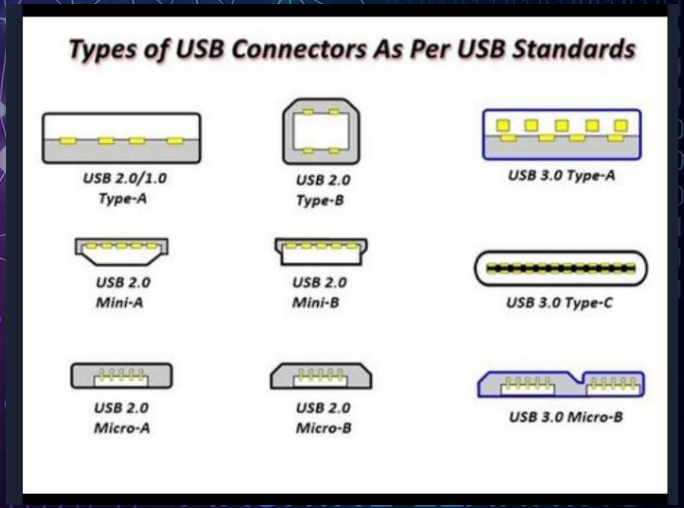
# Mass Storage – Removable

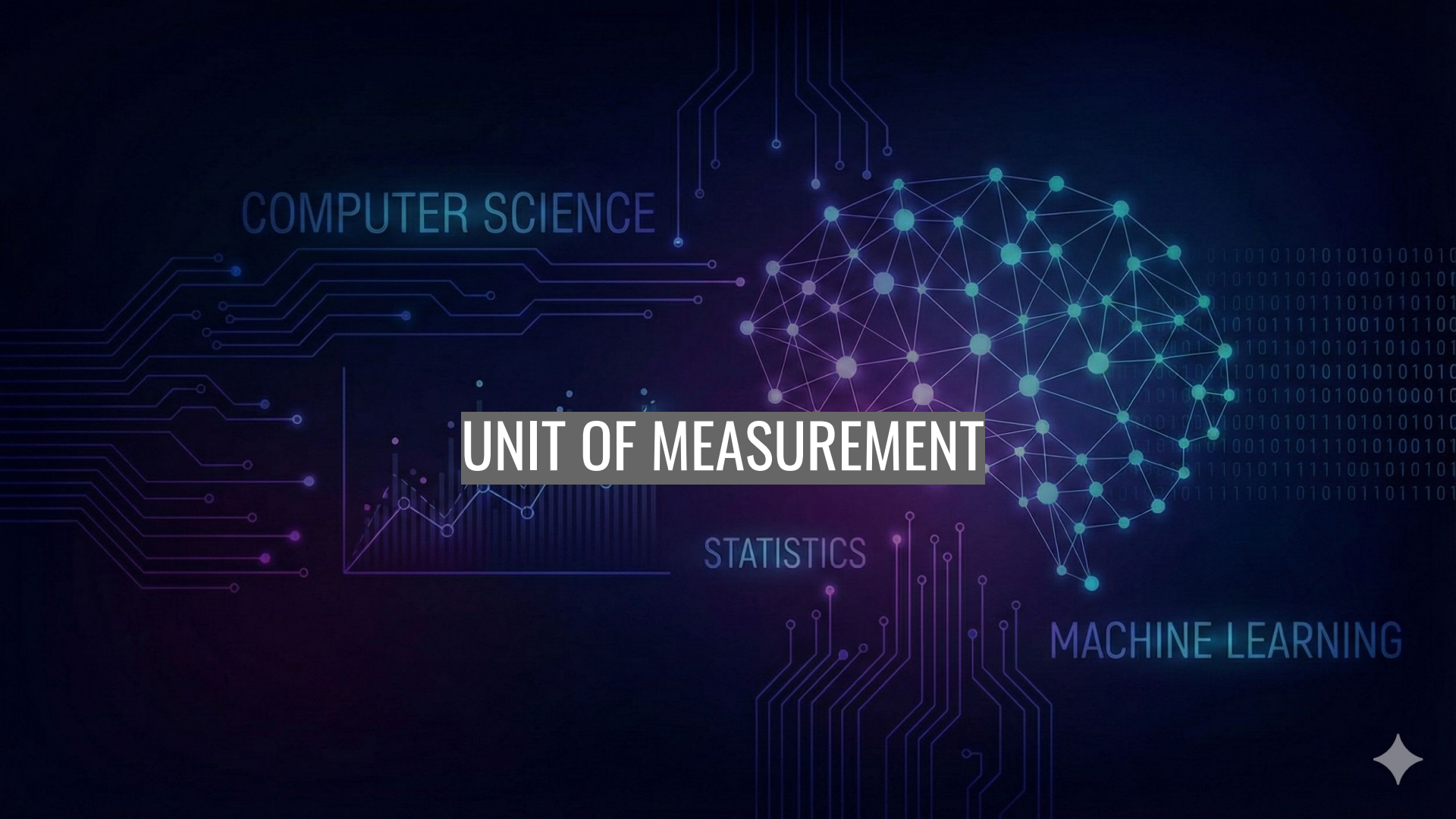
- It can be removed and separated from the computer
- **CDs and DVDs** , both read-only and read-write, data is stored and read using a laser light (the basic idea is simple reflection or no reflection to identify bits as 1 or 0)
- **USB flash drive**
- **Foppy disks** no longer used

# USB

| GENERATION | PORT COLOR | MAX SPEED | MAX POWER |
|---|---|---|---|
| **USB 2.0**<br>High Speed | ● Black / White | 480 Mbps | 2.5 W (500mA) |
| **USB 3.0**<br>SuperSpeed | ● Blue | 5 Gbps | 4.5 W (900mA) |
| **USB 3.1 / Type-C**<br>SuperSpeed+ / PD | ● Teal / Any | 10 Gbps+ | 100 W (Power Delivery) |

**Backwards Compatibility:** USB 3.0 ports (Blue) can accept USB 2.0 devices, but speeds will drop to 480 Mbps.




COMPUTER SCIENCE
UNIT OF MEASUREMENT
STATISTICS
MACHINE LEARNING

# Unit of measurement of information

- **BIT** = is the unit of measurement of information (from the English "binary digit"), defined as **the minimum quantity of information needed to distinguish between two equally likely events** . (Wikipedia)
- **BYTE** = 8 BITS (historically, characters were represented by 8 BITS, which is why **1 Byte remains the minimum addressable memory unit to this day** )
- “KiloByte” better “kibibyte” KiB = $2^{10}$ Byte = 1024 Byte
- “MegaByte” better “mebibyte” MiB = 1024 * 1024 Byte
- “GigaByte” better “gibibyte” GiB = 1024 * 1024 * 1024 Byte
- “TeraByte” better “tebibyte” TiB = 1024 * 1024 * 1024 * 1024 Byte

# Computer performance

- Clearly, increasing the performance of a computer means decreasing the time it takes to perform an operation.
- $T_{clock}$ is the clock period of the machine ( **frequency increase** )
- $CPI_i$ is instead the number of clock "shots" needed to execute the given instruction i ( **reduction of complexity for the single instruction** )
- $N_i$ is finally the number of instructions of type i (for example sums, jumps, etc.)

$$T_{esecuzione} = T_{clock} \sum_{i=0}^{n} N_i CPI_i$$

# MIPS

**MIPS** is an abbreviation for **Mega Instructions Per Second** , and indicates the number of general-purpose instructions a CPU executes in one second. It's a more general-purpose unit of measurement used to measure the performance of a computer than the FLOPS, which we'll see shortly.

$$\textbf{MIPS} = (\textbf{Clock Frequency}) / (10^{6}\ \textbf{CPI})$$

This type of measurement does not take into account, for example, optimizations due to the presence of the cache and the percentages of the different instructions within real programs, and not only.

# FLOPS

**FLOPS** is an abbreviation for **Floating Point Operations Per Second**, and indicates the number of floating-point operations a CPU performs in one second. It is a unit of measurement used to measure a computer's performance, particularly in scientific computing.

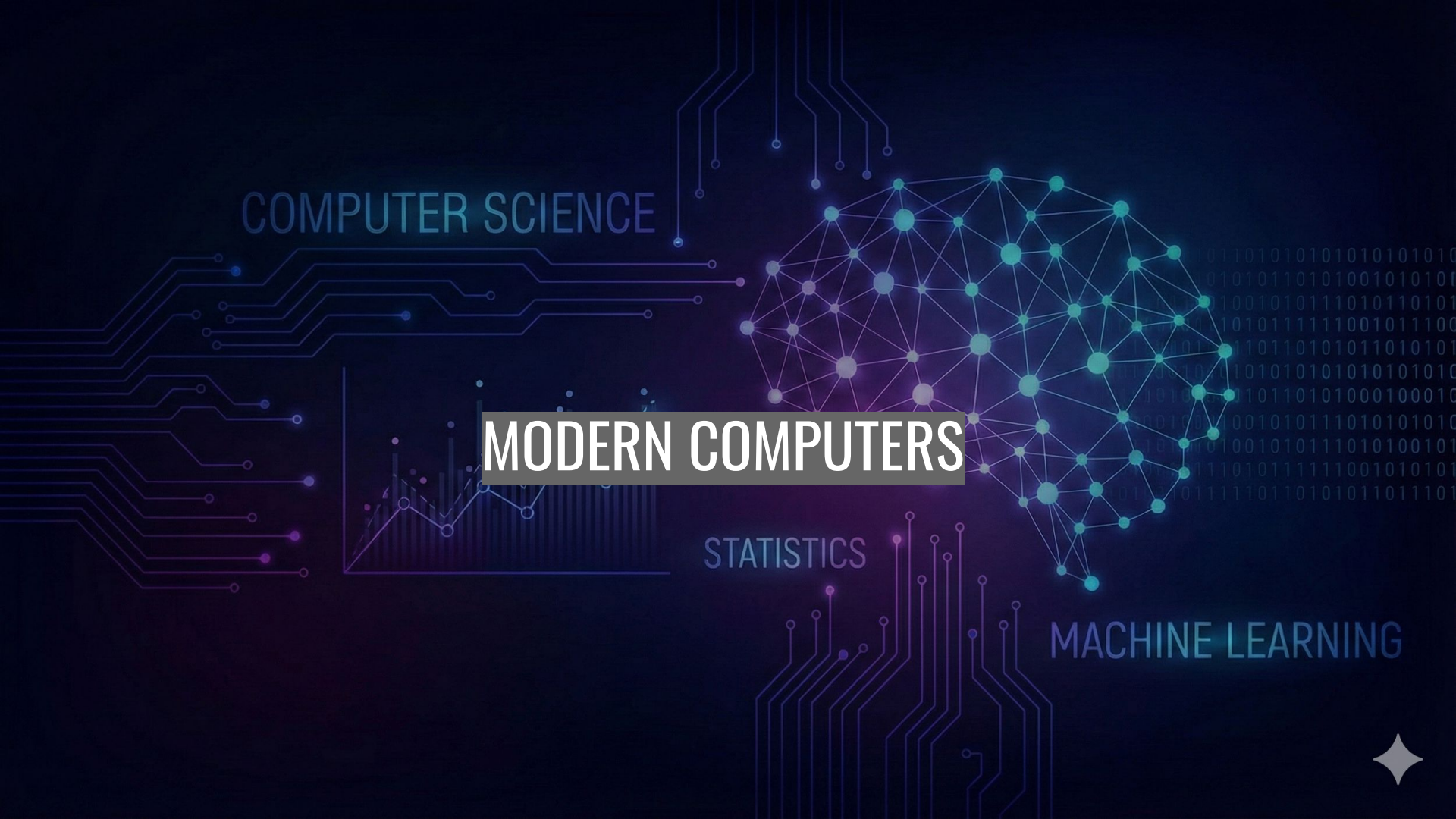
For example, in the case of a classic matrix product, $2*N^3$ operations are performed, so I can evaluate the FLOPS exactly by measuring the time needed to perform this multiplication and obtain:

$$[flops] = 2*N^3 / time$$

# SPEC

**Standard Performance Evaluation Corporation** is a non-profit organization that produces and maintains a standardized set of computer benchmarks (i.e., a set of test programs that are representative of real-world computer applications).

There are different sets of SPECs that are specific for example to different intended uses of the computer


COMPUTER SCIENCE
MODERN COMPUTERS
STATISTICS
MACHINE LEARNING

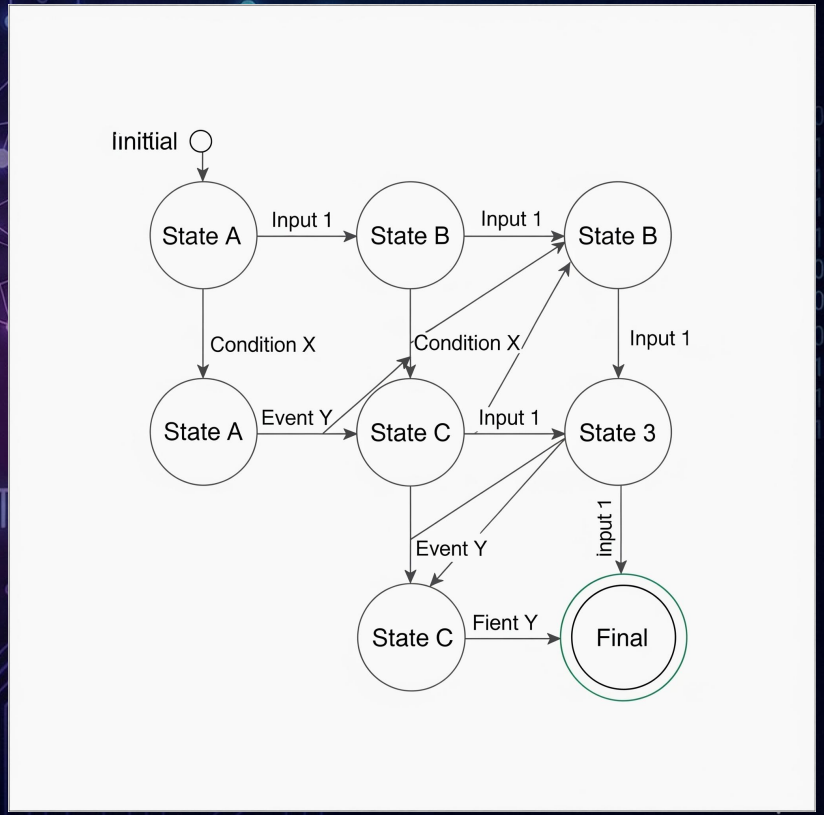
# FLOPs

| Prefix | Scale | Notation | Typical Device |
| --- | --- | --- | --- |
| GigaFLOPS | Billion | $10^{9}$ | Standard Laptop |
| TeraFLOPS | Trillion | $10^{12}$ | PS5 / Gaming PC |
| PetaFLOPS | Quadrillion | $10^{15}$ | Supercomputers (2010s) |
| ExaFLOPS | Quintillion | $10^{18}$ | Frontier (Current) |
| ZettaFLOPS | Sextillion | $10^{21}$ | Future AI Clusters |

# Finite State Machine

## The Rigid Specialist

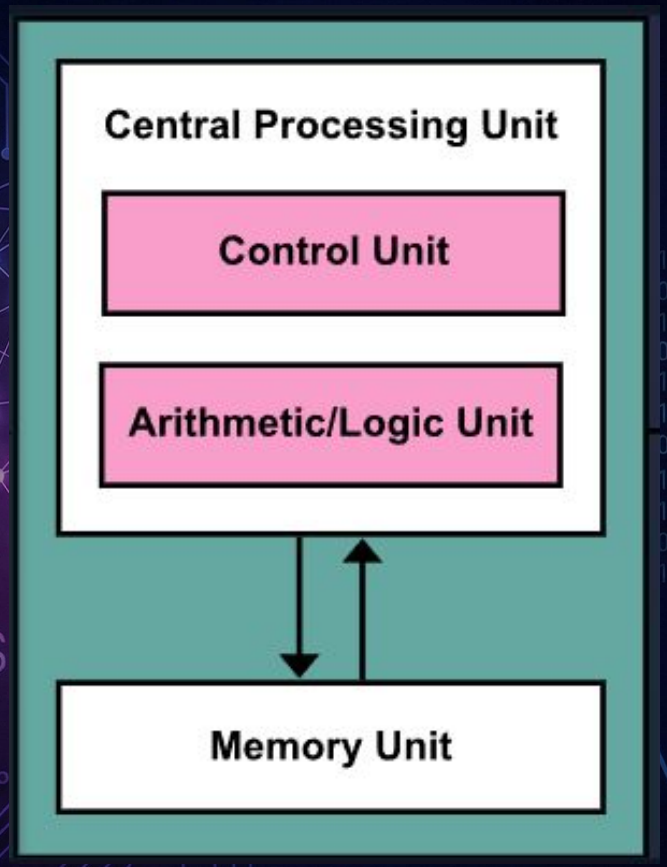
- An FSM is a computation model that can be in exactly one of a finite number of states at any given time.
- Logic: Transitions between states are triggered by specific inputs (Conditions).
- Storage: Does not store a "program" in memory. The logic is usually hardwired or fixed in the circuit structure.
- **Example: A vending machine. It waits for coins (Input), transitions to "Credit" (State), and dispenses a soda (Action). It cannot "learn" to play chess.**



# Von Neumann Machine

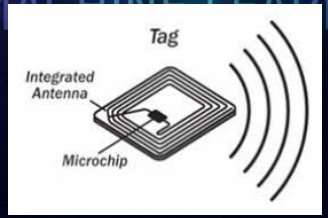
## The Flexible Generalist

- The Von Neumann architecture describes a computer where program instructions and data are held in the same memory storage.
- Stored Program: The machine can change its behavior by loading a different software program into memory.
- Cycle: It uses a Central Processing Unit (CPU) to endlessly Fetch, Decode, and Execute instructions.
- **Flexibility: The same hardware can calculate taxes, play video games, or browse the web.**



# RFID

- **RFID Radio-frequency identification** cost a few cents and can reach **a few MIPS**
  - Most standard RFID tags are **Finite State Machines** (**FSM**), not Von Neumann machines. However, advanced "Smart" RFID tags (like those in passports or credit cards) can be considered full computers that follow the **Von Neumann** (or Harvard) architecture.
- They use electromagnetic fields to automatically identify and track objects using an ID or other information.
- **Passive** if they use energy emitted by the RFID reader. Also battery-powered.
- **Active** with battery they periodically send a signal



# Embedded systems and SmartPhone Tablets

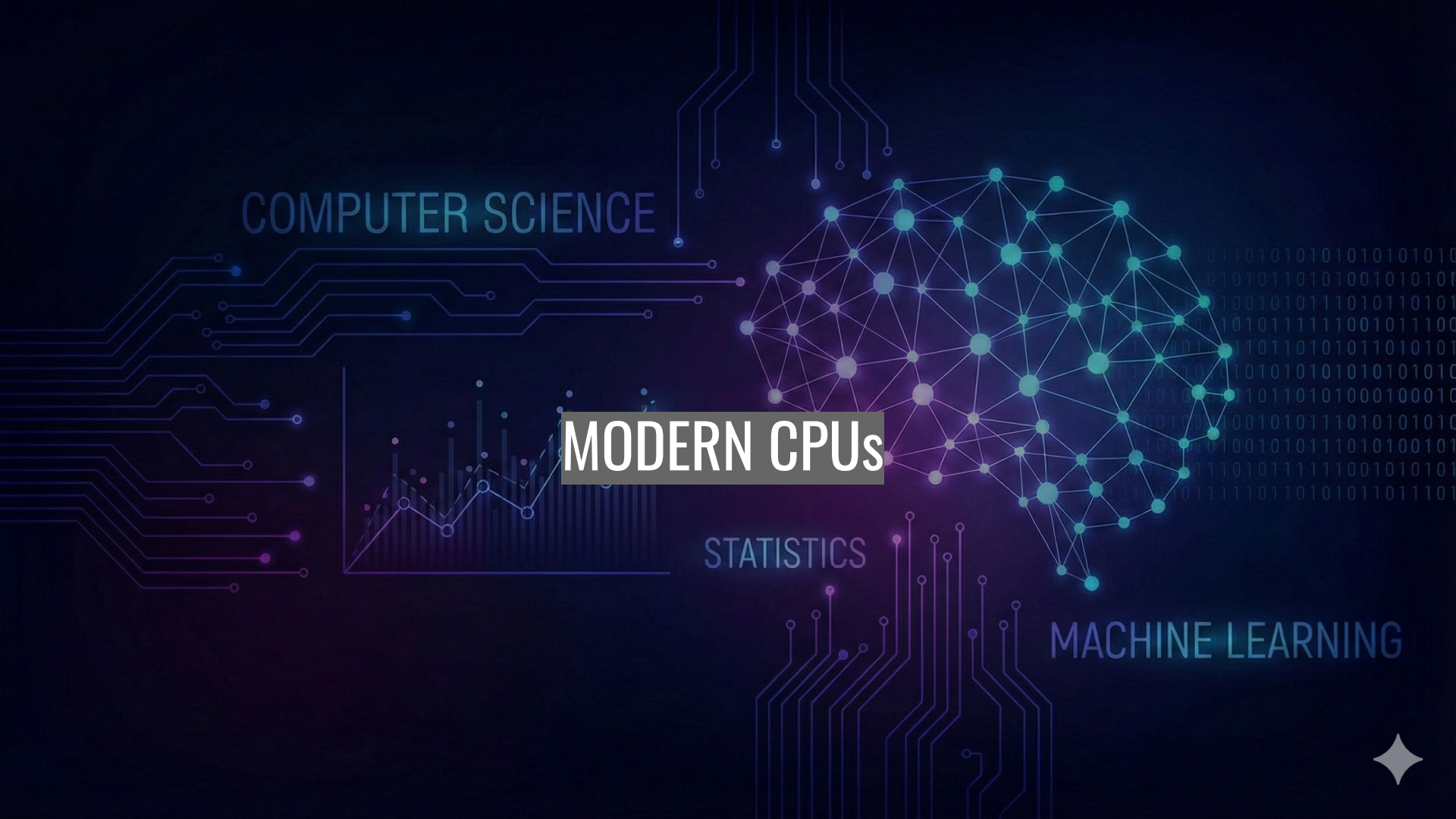
- **Embedded systems that** are very popular today are calculators designed for watches, cars, household appliances, medical devices, and audio/video players (Android Box). These calculators cost a few dozen euros and offer performance in the order of **a few hundred MIPS ( often equipped with a Linux OS )**.
- **Smartphones and Tablets,** on the other hand, are systems with significantly higher computing power. **Modern high-end smartphones (like iPhone 15 Pro or Galaxy S24) have neural engines and GPUs that perform in the Teraflops range, not just GFLOPS**

# Game Consoles, PCs and Workstations

- **Game consoles** are systems with overall performances that can reach around **10 to 12 TFLOPS (10,000 to 12,000 GFLOPS).also considering the GPUs**
- **PCs considering Desktops and Laptops** cover a wide range of possibilities starting from a few hundred euros up to a few thousand with performances that therefore **go from Hundreds to 80,000 GFLOPS considering the GPU**
- There are **Servers and Workstations** that are designed for **High Performance Computing** and centralization of service. **For €20,000 today, you can expect 100+ TFLOPS for graphics/AI (Workstation), but perhaps only 10-20 TFLOPS for high-precision scientific simulation (Server).**

# HPC

- In order to increase the performance of the computing resource in general it is necessary to couple together numerous computers **interconnected with high-performance networks (parallel computing)**
- For example, **workstation clusters**
- **Supercomputers As of 2024, the top supercomputers (like Frontier) have broken the Exaflop barrier (1000 PFLOPS)**
- Obviously, production costs and maintenance costs increase in the same way (even just in terms of absorbed power).


COMPUTER SCIENCE
MODERN CPUs
STATISTICS
MACHINE LEARNING

# CISC vs RISC

The speed of execution of a single instruction is one of the determining factors of the CPU's performance. Two different perspectives:

- **CISC (Complex Instruction Set)** in this case the basic idea is that the basic instruction set of a CPU must be as rich as possible, even if **each single instruction actually requires more clock cycles to be executed.**
- **RISC (Reduced Instruction Set)** in this case **each instruction is executed in a single clock cycle** . Obviously, more RISC instructions will be needed to execute the same single instruction as a CISC.
- CISC architecture was the predominant one on the market in the 1970s and 1980s. Today there is a trend in favor of the RISC type CPU.

# Improve Performance

**Increasing the performance of a CPU** is always a compromise between costs and consumption, for example some basic strategies can be adopted:

- **Reduce the number of cycles** required to execute a single instruction (trivial example of multiplication)
- **Increase the frequency** (clock) with obvious physical limits ( **for example the speed of light** )
- **Parallelism, for example, performing multiple operations in parallel** (at the same time)
  - **At the instruction level, multiple instructions** are executed in parallel by the same CPU, for example using **pipelines or superscalar processors.**
  - **Core-level parallelism, meaning more cores per CPU**

# Improve Performance – Increase Clock Frequency

Until 2000, the increase in CPU performance largely coincided with the increase in clock frequency. **We have reached about 4 GHz.** We have reached the physical limits ( **1 GHz and therefore in one ns the distance that an electrical impulse can travel, imagining it travels at the speed of light in a vacuum, is approximately 33 cm** ).

- High frequencies create **noise** and **increase the heat to be dissipated**
- Delays in signal propagation,
- **Bus skew** signals traveling on different lines travel at different speeds

# Improve Performance – Increase Clock Frequency and Miniaturization

**The Speed of Light Limit** (Signal Delay): This is the most fundamental reason. Electricity travels very fast, but it is not instantaneous. Inside copper or gold wires, signals travel at roughly 15 to 20 centimeters per nanosecond (slower than light in a vacuum).

- The Scenario: Imagine a CPU running at 4 GHz.
  - The Timing: It has a cycle time of 0.25 nanoseconds.
  - The Distance: In 0.25 nanoseconds, a signal can physically travel only about 4 to 5 centimeters.
  - If a CPU were large (e.g., the size of a dinner plate), the clock cycle would end before the electrical signal could even travel from one side of the chip to the other. The data wouldn't arrive in time for the next calculation.

To increase frequency (reduce the time), you must reduce the distance. You have to pack the components closer together so the signal arrives instantly.

# Improve Performance – Increase Clock Frequency and Miniaturization

**The "Bucket" Problem  transistor acts like a tiny capacitor**—think of it as a bucket of water.

- To switch a transistor from "0" to "1" (Off to On), you have to fill that bucket with electrons.
- To switch back to "0", you have to empty it.
- The problem: Large Transistors = Large Buckets. It takes longer to fill them up. If you try to switch them too fast (high frequency), they never get fully full or fully empty, and the data becomes corrupted.
- Small Transistors = Small Buckets. You can fill and empty a thimble much faster than a bathtub.

Miniaturization makes the "gate capacitance" smaller, allowing the transistor to switch states billions of times per second without lagging.

# Improve Performance – Increase Clock Frequency and Miniaturization

**Power and Heat (Power Density)**

**The formula for the power consumed by a CPU is roughly:**

$$P = C * V^2 * f$$

**(Power = Capacitance × Voltage squared × Frequency)**

If you increase the Frequency (f), the Power (P) shoots up, generating massive heat.

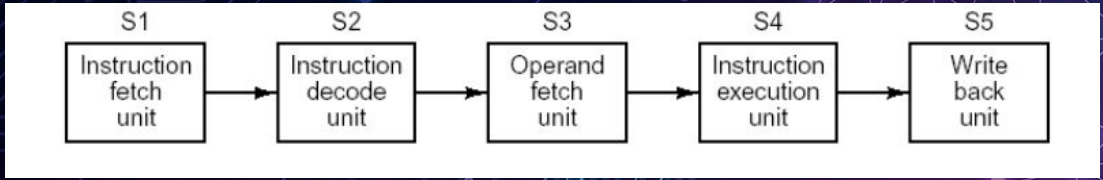
To stop the CPU from melting, you must lower the other variables.

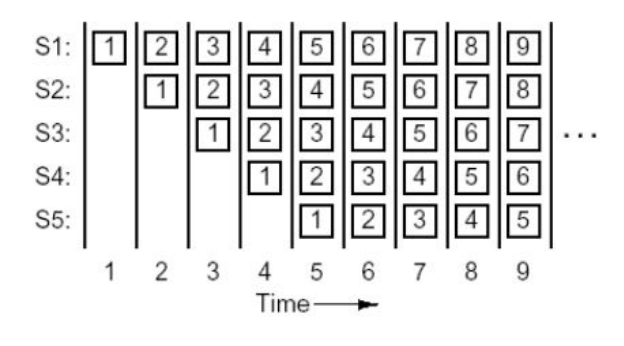
- **Miniaturization reduces Capacitance (C).**
- **Miniaturization allows you to lower the Voltage (V).**

**By making things smaller, you require less energy to move the electrons, which compensates for the heat generated by the increased speed.**

# Improving Performance – Pipeline

**Each single instruction is divided into multiple stages (or phases) and each phase is handled by a dedicated piece of CPU (hardware):**

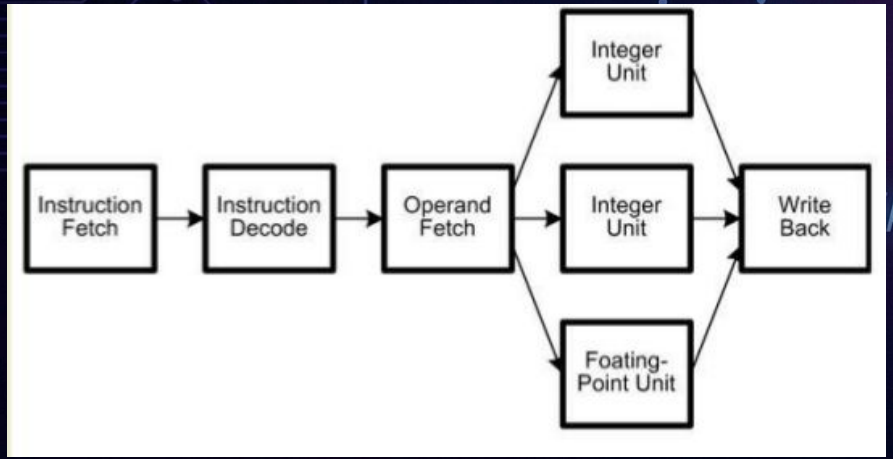



Clearly, **the operations must be independent** . In this case, after an initial time ( **latency** ) to load the pipeline, there will be n operations executed in parallel. Multiple pipelines can also be used, and therefore multiple instructions are read at a time and executed.

# Improving Performance – Superscalars

Different instructions process their operands simultaneously on different hardware units, **in practice there are several functional units of the same type, for example there are several ALUs**

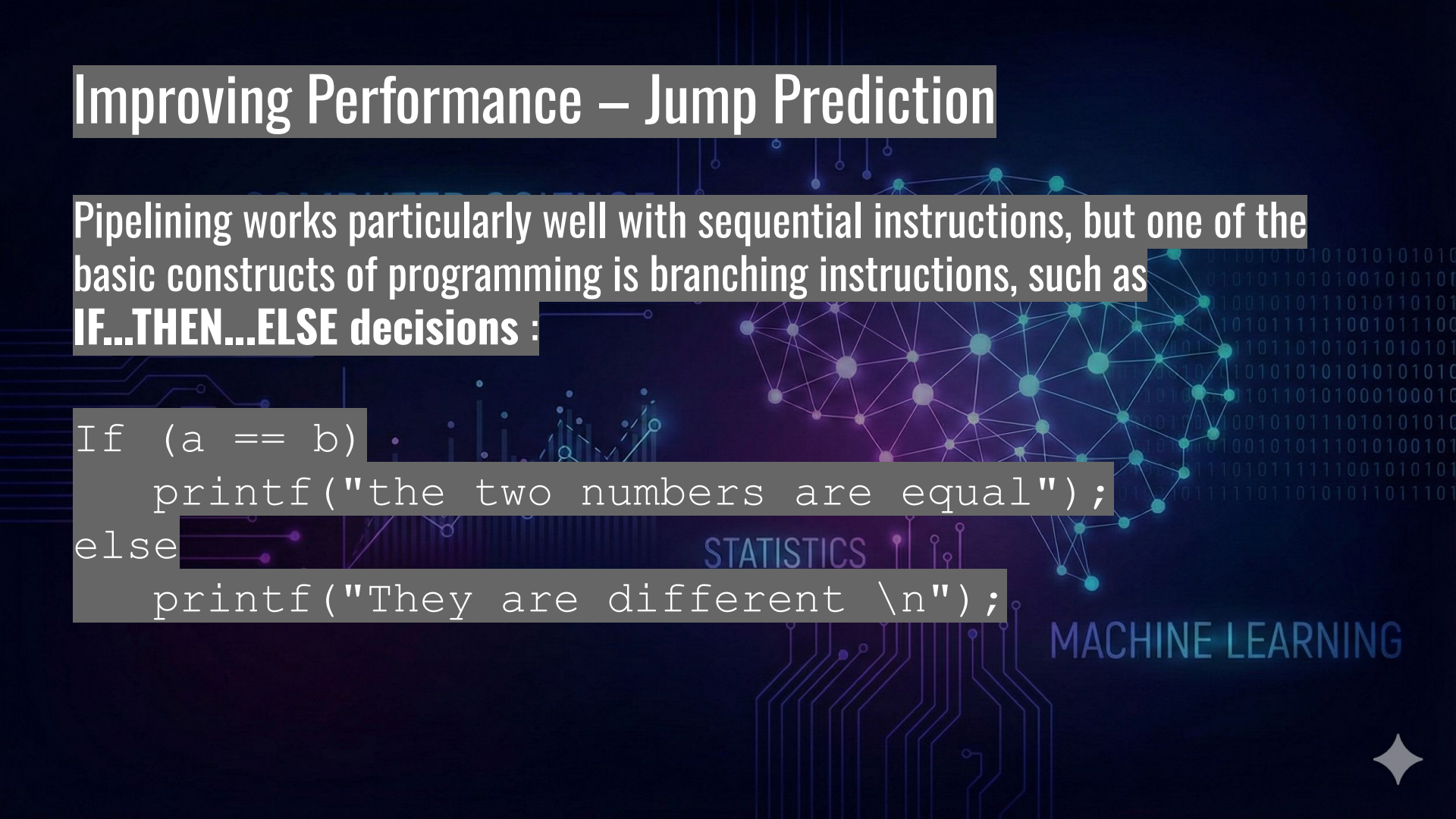



Multiple Execution Units: Unlike a scalar processor, which has a single pipeline and can (at best) complete one instruction per cycle, a superscalar processor has multiple execution units (e.g., two integer ALUs, two Floating Point Units, and a Load/Store unit).

# Improving Performance – Jump Prediction

Pipelining works particularly well with sequential instructions, but one of the basic constructs of programming is branching instructions, such as **IF...THEN...ELSE decisions** :

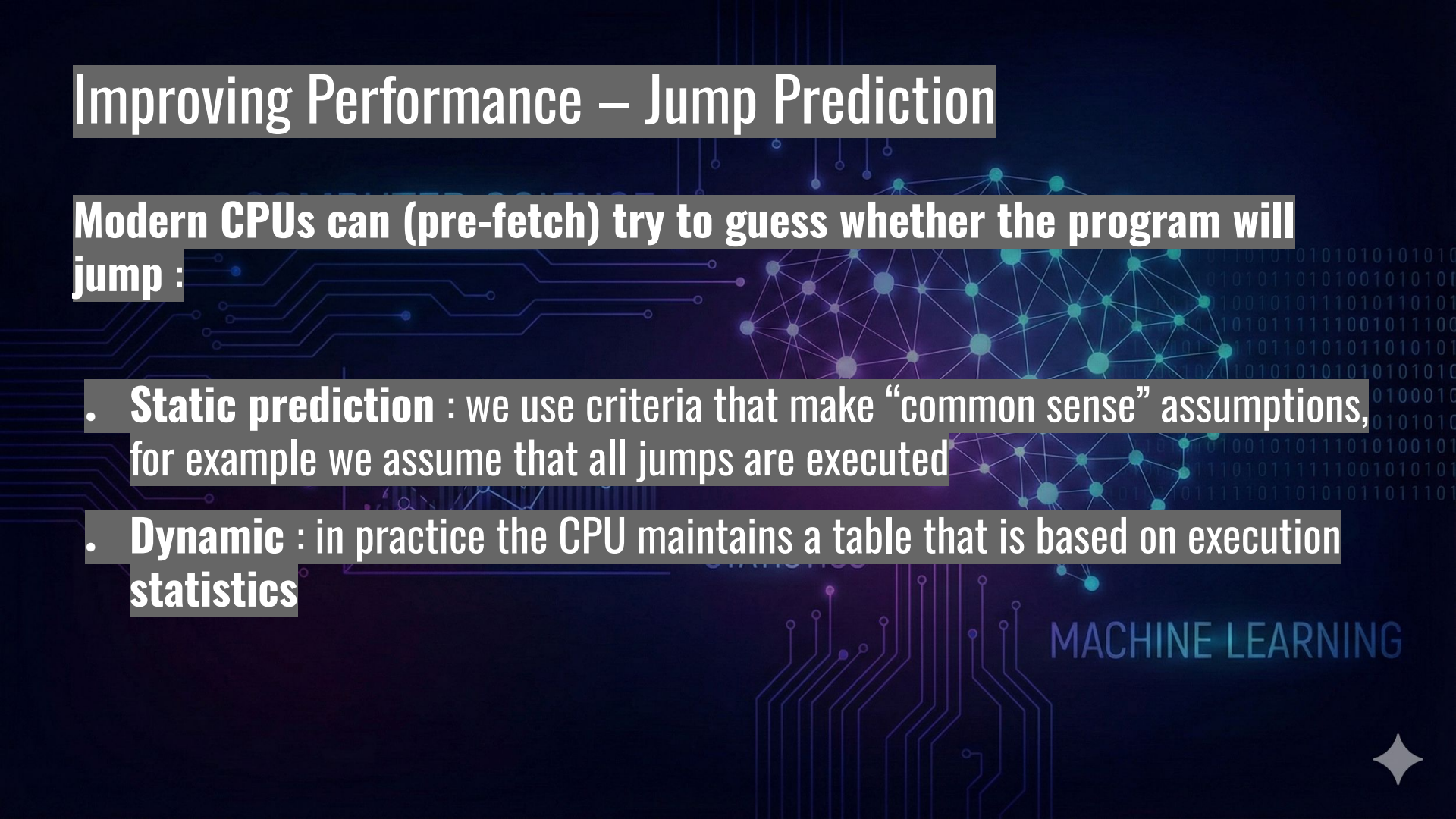
```
If (a == b)
   printf("the two numbers are equal");
else
   printf("They are different \n");
```



# Improving Performance – Jump Prediction

**Modern CPUs can (pre-fetch) try to guess whether the program will jump :**

- **Static prediction** : we use criteria that make “common sense” assumptions, for example we assume that all jumps are executed
- **Dynamic** : in practice the CPU maintains a table that is based on execution **statistics**



# Improving Performance – Out-of-order and Speculative Executions

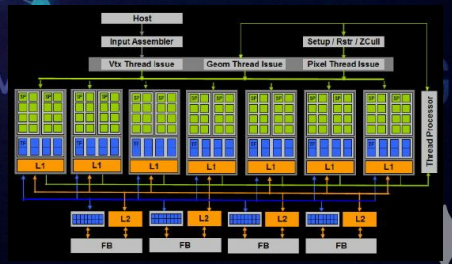
CPU design is significantly simpler if all instructions are executed sequentially, but as we've seen, we can't make this assumption upfront. There are dependencies. For example, executing a given instruction requires that the result of the previous instruction be known.

- **Out-of-order execution** : Modern CPUs can **temporarily skip some instructions to increase performance, putting them on hold** to follow other instructions that do not introduce dependencies.
- **Speculative execution** : running particularly heavy parts of the code before being sure they are actually needed.

# CPU - Final Thoughts

Modern CPUs:

- **Multi-core** : in practice, there are multiple independent processors in the same chip, **each with its own cache memory, for example, interconnected with each other** . This allows for increased "theoretical" performance without increasing the frequency (even **Hyper-threading** ).
- **GPUs** (graphics processors) are increasingly used today to accelerate computing.

# CPU - Final Thoughts

**Embedded systems, IoT, and Mobile CPUs (including Raspberry Pi):**

- ARM refers to a prominent family of RISC (32-bit and 64-bit) processor architectures developed by Arm Holdings (a British company). Uniquely, Arm does not manufacture chips directly; instead, it designs the architecture and licenses it to other companies. These are then produced as System-on-Chips (SoC) by manufacturers such as Apple, Qualcomm, Samsung, NVIDIA, STMicroelectronics, and Broadcom (used in Raspberry Pi).
- These RISC CPUs are designed to optimize the balance between high performance and energy efficiency. While originally dominant in mobile and embedded devices, the architecture has now scaled up to power laptops and supercomputers.
    - Comparison: A modern efficient ARM core (e.g., a Cortex-A55 or similar found in smartphones and IoT devices) typically consumes between 100 mW to 500 mW under normal load. In contrast, a high-performance desktop x86 CPU (like a modern Intel Core i9 or AMD Ryzen 9) can consume over 200 W at peak performance.

# Why RISC Consumes Less Power Than CISC

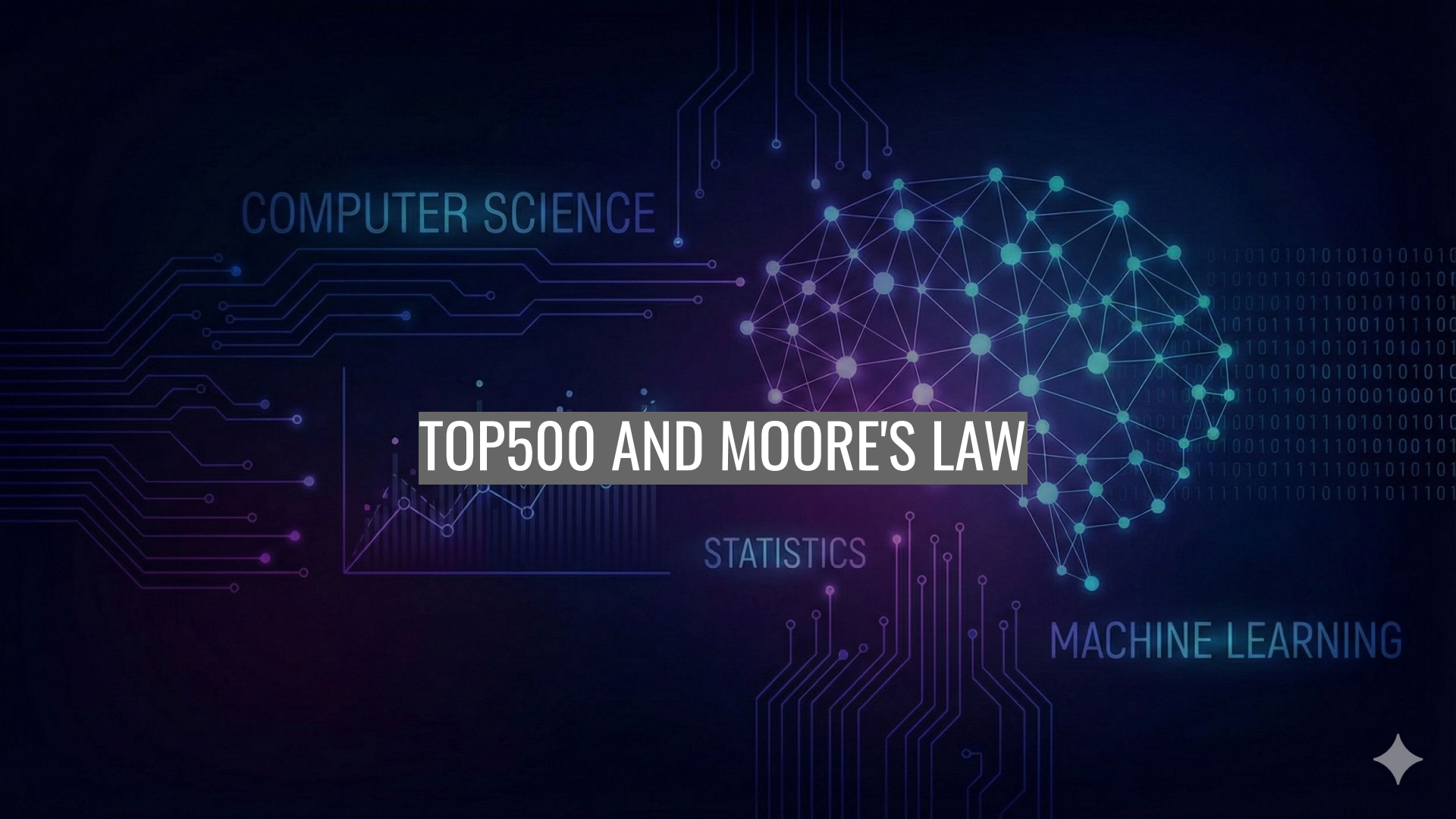
**The Decoding Penalty (The Main Culprit). This is the single biggest factor.**

- **CISC (e.g., Intel x86): Instructions are like complex sentences**. They have variable lengths (some are 1 byte long, others are 15 bytes long) and can perform many tasks at once (e.g., "Go to memory, get a number, add it to this register, and save it back").
  - **The Cost: The CPU needs a massive, power-hungry circuit called a Decoder just to figure out where one instruction ends and the next begins**. It takes significant electrical energy to "translate" these complex instructions into signals the chip can understand.
- **RISC (e.g., ARM): Instructions are like simple** commands. They have a fixed length (usually 32 bits).
  - The Advantage: **The hardware knows exactly how big every instruction is**. The decoder is tiny, simple, and essentially "hardwired." It uses very little power because it doesn't have to perform complex analysis on the code stream.

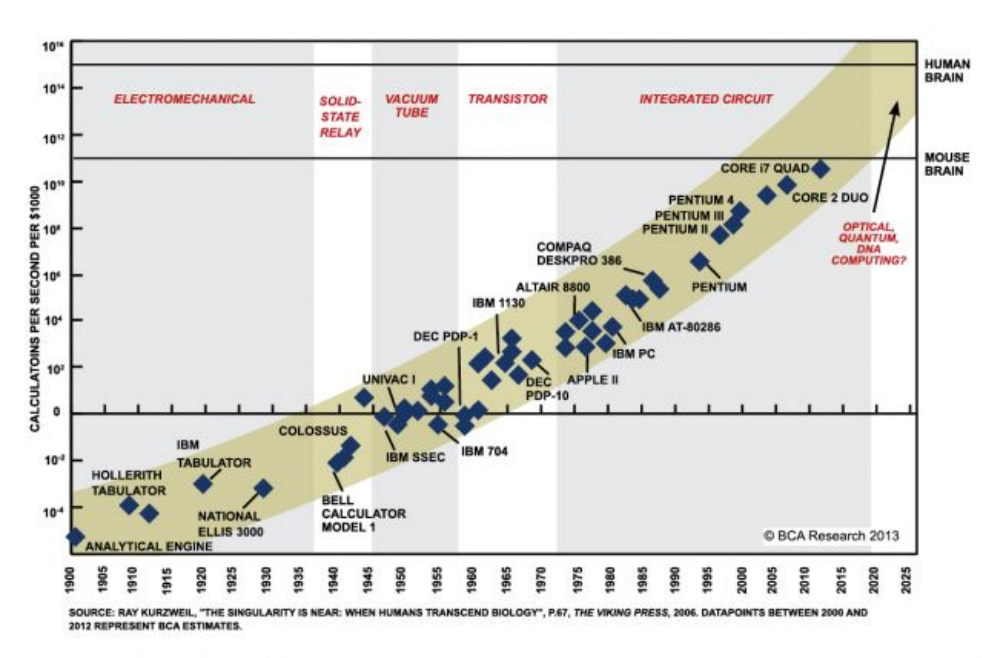
# Why RISC Consumes Less Power Than CISC

**Today, the line is somewhat blurred.**

- **Modern CISC processors (like Intel Core i9) actually cheat: they use a hardware translator to turn CISC instructions into RISC-like "micro-operations" internally. However, that translation step itself still burns power constantly.**
- **Modern RISC processors (like Apple M3 or Snapdragon) are becoming more complex to get faster, but they still benefit from the fundamental efficiency of their instruction set, which allows them to run cooler and use less battery than their CISC counterparts.**


COMPUTER SCIENCE
TOP500 AND MOORE'S LAW
STATISTICS
MACHINE LEARNING

# Moore's Law




**Moore's first law initially stated that microcircuit complexity (e.g., transistors per chip) doubles every 18 months. Though accurate for decades, the doubling period has slowed** (now 2.5–3 years). Current performance gains prioritize specialized architectures (like AI accelerators), 3D stacking (chiplets), and efficient design over solely transistor scaling.

# Moore's Law

Is it still true today? In the 2020s, Moore's Law is slowing down or, according to some (like Nvidia's CEO), is "dead."

- Physical Limits: We are approaching the size of atoms. Transistors are now measured in nanometers (e.g., 3nm, 2nm). At this scale, quantum tunneling and heat dissipation become massive problems.
- Economic Limits (Moore's Second Law): While we can still pack more transistors, the cost to build the factories (fabs) to do so is doubling. A modern fab costs over $20 billion, making it harder to keep the "cheaper and faster" promise of the original law.

# TOP500

- Difference between **sustained performance and peak performance**

- To objectively evaluate the performance of a computer you need a reference test, a **standard benchmark, for example Linpack**

- TOP500 http://www.top500.org/ , ranking of the 500 most powerful computers in the world

# Top 500 list November 2025

## GLOBAL TOP 10 SUPERCOMPUTERS

Ranking based on HPL Performance (Rmax)

Source: Top500.org (Novembe

| # | SYSTEM | CORES | RMAX (PFLOP/S) | POWER (KW) |
|---|---|---|---|---|
| 1 | El Capitan NEW #1 *USA (LLNL)* | 11.3M | **1,809.00** | 29,685 |
| 2 | Frontier *USA (ORNL)* | 9.1M | **1,353.00** | 24,607 |
| 3 | Aurora *USA (ANL)* | 9.3M | **1,012.00** | 38,698 |
| 4 | JUPITER Booster NEW *Germany (EuroHPC)* | 4.8M | **1,000.00** | 15,794 |
| 5 | Eagle *USA (Microsoft Azure)* | 2.1M | 561.20 | N/A |
| 6 | HPC6 NEW *Italy (Eni S.p.A)* | 3.1M | 477.90 | 8,461 |
| 7 | Fugaku *Japan (RIKEN)* | 7.6M | 442.01 | 29,899 |
| 8 | Alps *Switzerland (CSCS)* | 2.1M | 434.90 | 7,124 |
| 9 | LUMI *Finland (EuroHPC)* | 2.8M | 379.70 | 7,107 |
| 10 | Leonardo *Italy (EuroHPC)* | 1.8M | 241.20 | 7,494 |

# Top500 list historical chart


An estimate of the **computing power of the human mind is** a few dozen **petaflops or more.**

That estimate (a few dozen petaflops) is considered correct but conservative based on older theoretical models, though many modern neuroscientists and computer engineers now believe the number is significantly higher—likely in the **Exaflop (1,000 Petaflops) range.**

# Top500 list historical chart

## PERFORMANCE REALITY: PEAK VS. SUSTAINED

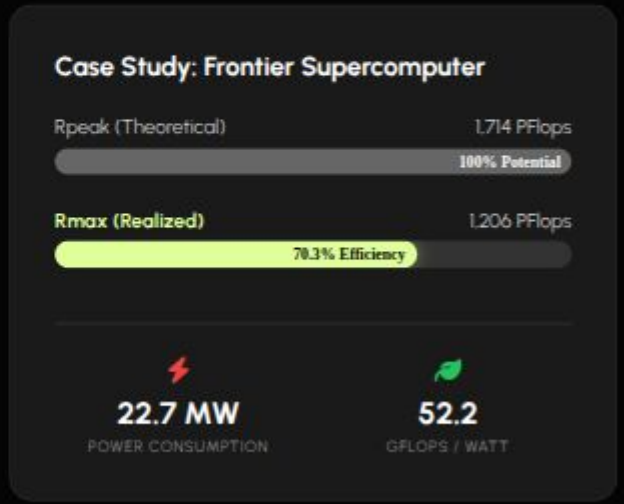
### Rpeak (Theoretical Peak)

The absolute hardware limit calculated by the sum of all processors running at maximum frequency without delays.
*"The Speedometer Max (300 km/h)"*

### Rmax (Sustained/Linpack)

The actual performance achieved running a real-world benchmark (HPL). This accounts for memory latency, interconnect overhead, and thermal limits.
*"The Actual Highway Speed (180 km/h)"*

⚠ **The Efficiency Gap:** We typically lose 20-40% of theoretical power to heat, latency, and system overhead.




In real applications the gap is much worst

HINE LEARNING

# Top500 list historical chart

## Where does the "Petaflop" estimate come from?

The estimate of $10^{15}$ to $10^{16}$ operations per second (1–50 Petaflops) was popularized by futurists like Hans Moravec and Ray Kurzweil in the late 1990s and early 2000s.

- They estimated the brain has roughly $10^{14}$ synapses (connections). If each synapse processes roughly 10 signals per second, you get $10^{15}$ operations (1 Petaflop).
- If the brain works simply (like a binary switch), "a few dozen petaflops" is correct.

# Top500 list historical chart

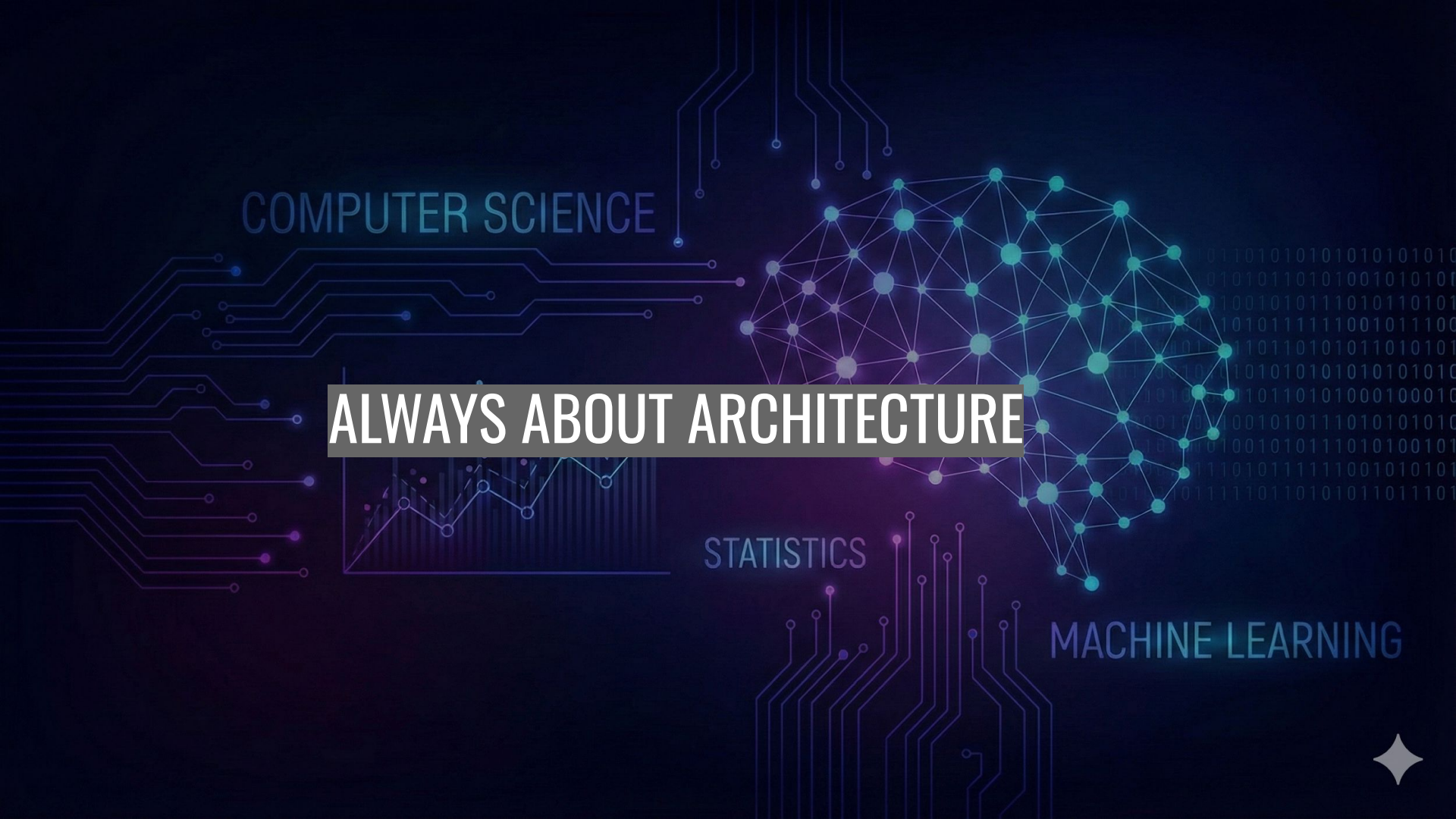
## The Modern View: The "Exaflop" Estimate

Recent research suggests the brain is much more complex. A synapse isn't just a simple on/off switch; it is a complex molecular machine that processes information non-linearly.

- If you need to simulate the chemical interactions within the synapses and dendrites to replicate the brain's function, the computational requirement jumps to 1 Exaflop ($10^{18}$) or even 1 Zettaflop ($10^{21}$).
- By this standard, the "few dozen petaflops" estimate is too low by a factor of 100 to 1,000.

# Top500 list historical chart

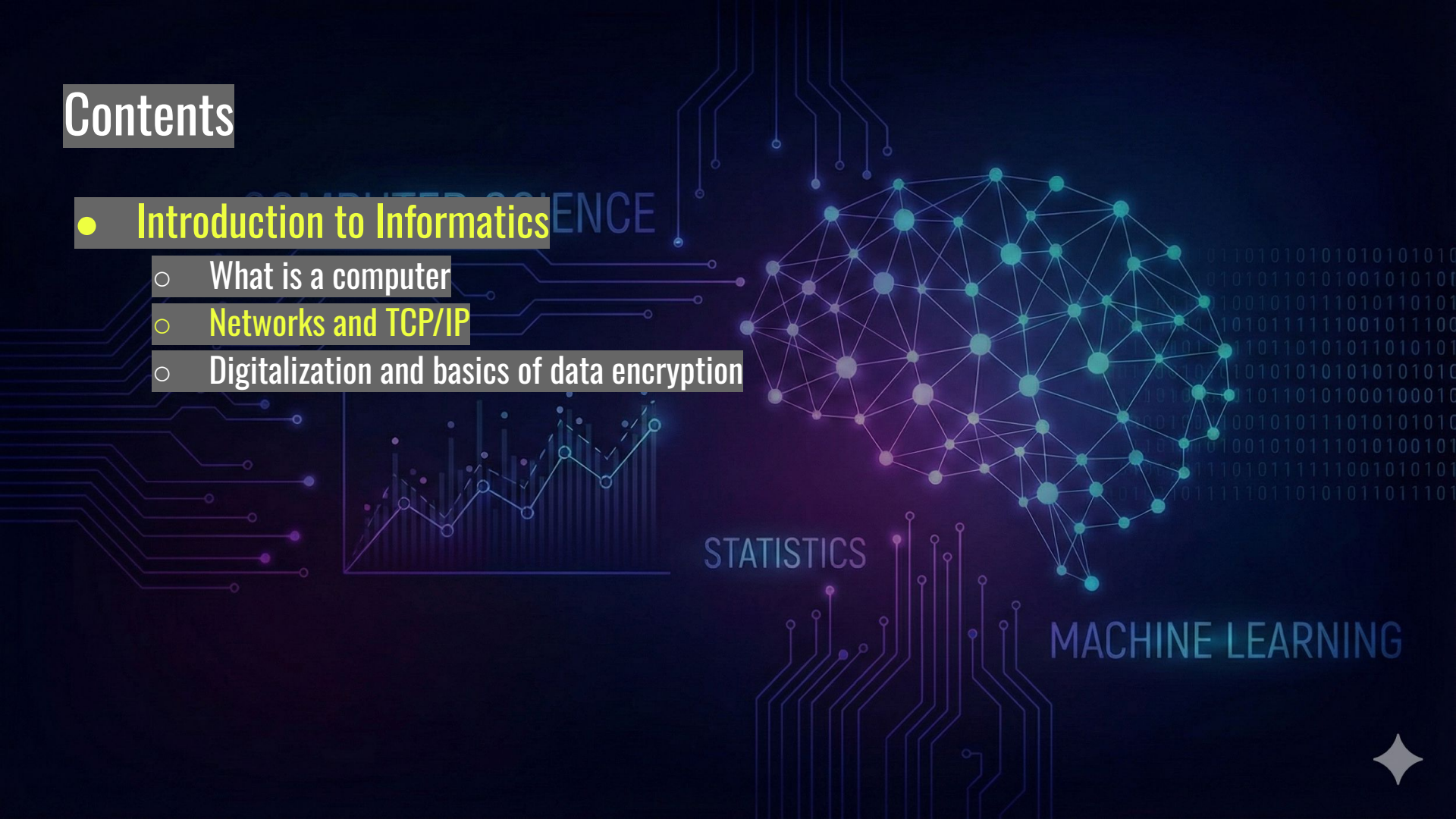
| Feature | Human Brain | El Capitan (Nov 2025 #1 Supercomputer) |
|---|---|---|
| **Speed (Est.)** | ~1 Exaflop (Variable) | 1.8 Exaflops (Sustained) |
| **Parallelism** | Massive (Billions of threads) | High (Millions of cores) |
| **Clock Speed** | Slow (~200 Hz / 0.0002 GHz) | Fast (1.8 GHz - 3.0 GHz) |
| **Power Usage** | **~20 Watts** (A dim lightbulb) | **~30,000,000 Watts** (A small city) |
| **Cooling** | Blood flow | Massive liquid cooling plant |


COMPUTER SCIENCE
ALWAYS ABOUT ARCHITECTURE
STATISTICS
MACHINE LEARNING

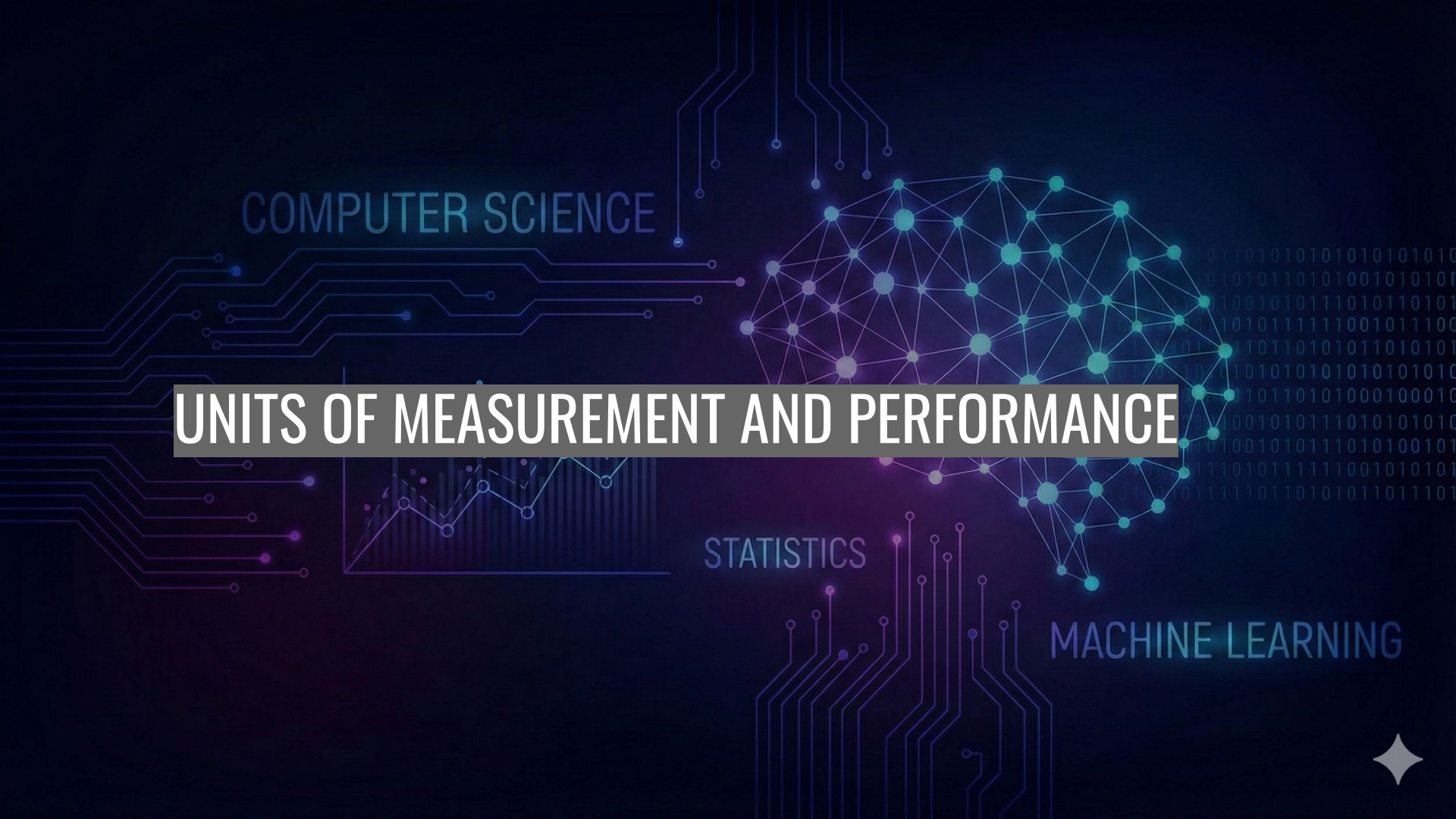
# FPGA

- **Field Programmable Gate Array** is an integrated circuit whose functions are programmable via software. FPGAs (Field Programmable Gate Arrays) are semiconductor devices based on an array of configurable logic blocks (CLBs) connected via programmable interconnects.
- FPGAs can be reprogrammed based on the desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from application-specific integrated circuits (ASICs), which are custom-built for specific design tasks. While one-time programmable (OTP) FPGAs are available, the dominant types are SRAM-based, which can be reprogrammed as the design evolves.
- **ASIC (Application Specific Integrated Circuit)**

# Contents




- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption




# UNITS OF MEASUREMENT AND PERFORMANCE

# Performance

**Bandwidth** is the maximum amount of data (number of bits) that can be transmitted in a channel. It is **often used as an approximation of actual throughput** (unit of measurement e.g. **Mbps** ).

**Throughput** ( **Performance** ) amount of data (number of bits) transmitted on the channel in a certain period of time (unit of measurement e.g. **Mbps** )

# Performance

- **Latency** or delay **is** the time taken by a message to go from one point to another ( **unit of measurement is time, for example ms = millisecond = (1/1000) s** ).
  - Transmission times are due to the **simple speed of propagation of the signal in the transmission medium** , and therefore to the **distance**
  - **Times required to process, for example, the header of transmitted packets**
- **Round Trip Time (RTT)** is the time it takes for a message to go from point A to point B and back again from B to A ( **ping and traceroute** ).

# Performance

```
traceroute to www.google.com (216.58.209.36), 30 hops max, 60 byte packets
 1  telecomitalia.it (192.168.1.1)  0.338 ms  0.203 ms  0.262 ms
 2  * * *
 3  172.18.25.0 (172.18.25.0)  3.892 ms  4.317 ms 172.18.25.4 (172.18.25.4)  3.046 ms
 4  172.18.25.212 (172.18.25.212)  3.731 ms 172.18.25.220 (172.18.25.220)  3.957 ms 172.18.25.212 (172.1
.25.212)  4.022 ms
 5  172.19.184.18 (172.19.184.18)  6.530 ms 172.19.184.22 (172.19.184.22)  7.524 ms 172.19.184.18 (172.1
.184.18)  7.087 ms
 6  172.19.177.72 (172.19.177.72)  10.343 ms 172.19.177.46 (172.19.177.46)  8.037 ms 172.19.177.72 (172.
9.177.72)  9.729 ms
 7  172.19.177.8 (172.19.177.8)  19.077 ms  18.159 ms 172.19.177.2 (172.19.177.2)  16.852 ms
 8  195.22.205.116 (195.22.205.116)  18.488 ms 195.22.192.144 (195.22.192.144)  20.639 ms 195.22.205.98
195.22.205.98)  20.207 ms
 9  72.14.204.72 (72.14.204.72)  22.495 ms 72.14.243.190 (72.14.243.190)  18.924 ms 72.14.219.236 (72.14
219.236)  18.668 ms
10  * 66.249.95.89 (66.249.95.89)  17.472 ms *
11  142.250.211.20 (142.250.211.20)  16.206 ms 142.251.235.174 (142.251.235.174)  14.762 ms 142.251.235.
72 (142.251.235.172)  16.842 ms
12  108.170.232.169 (108.170.232.169)  18.552 ms mil07s12-in-f4.1e100.net (216.58.209.36)  18.694 ms 192
178.104.214 (192.178.104.214)  20.536 ms
```

# Performance

```
PING 216.58.204.228 (216.58.204.228) 56(84) bytes of data.
64 bytes from 216.58.204.228: icmp_seq=1 ttl=114 time=16.6 ms
64 bytes from 216.58.204.228: icmp_seq=2 ttl=114 time=16.8 ms
64 bytes from 216.58.204.228: icmp_seq=3 ttl=114 time=16.9 ms
64 bytes from 216.58.204.228: icmp_seq=4 ttl=114 time=16.8 ms
64 bytes from 216.58.204.228: icmp_seq=5 ttl=114 time=16.9 ms
^C
--- 216.58.204.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 16.630/16.792/16.860/0.085 ms
                :~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.386 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.302 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.359 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=0.356 ms
```

# Performance

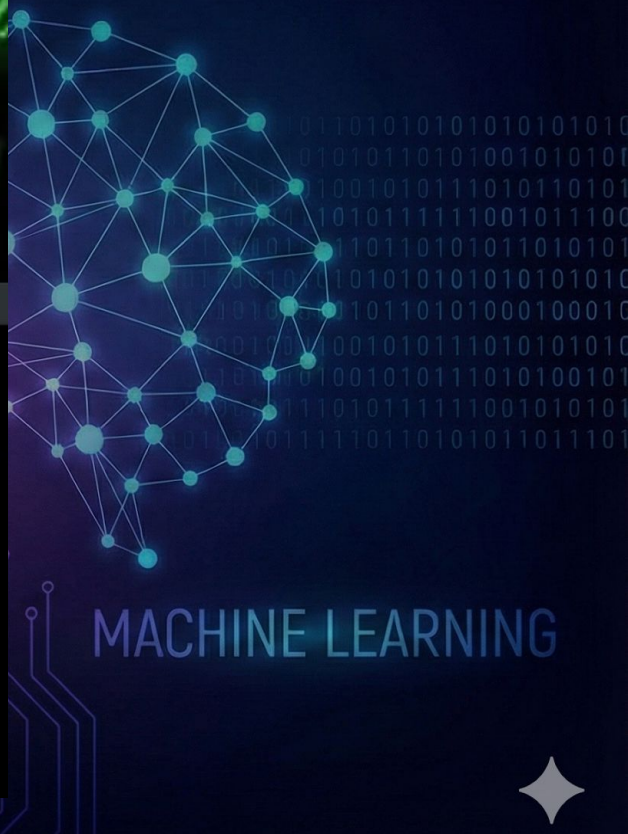
```
   Speedtest by Ookla

     Server: TIM SpA - Pescara (id = 36728)
        ISP: TIM
    Latency:     2.85 ms   (0.12 ms jitter)

   Download:   940.69 Mbps (data used: 682.2 MB)

     Upload:   313.48 Mbps (data used: 141.4 MB)
Packet Loss:     0.0%
```

# Performance

```
root@raspberrypi:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.10.22 port 5001 connected with 192.168.10.218 port 40518
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec   113 MBytes  94.2 Mbits/sec
```

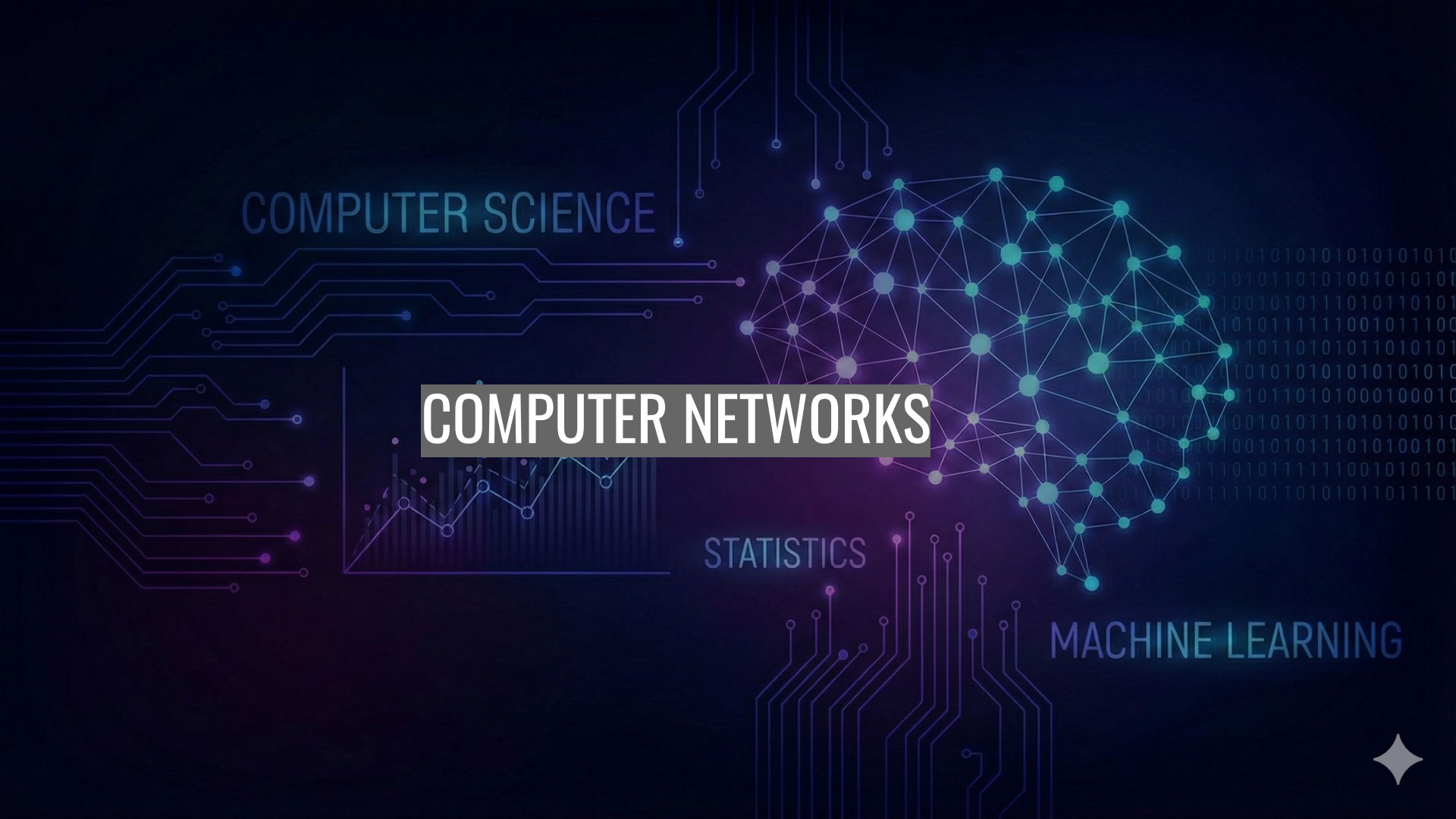
root@buchner: ~

```
root@buchner:~# iperf -c rpi.ego.eco
------------------------------------------------------------
Client connecting to rpi.ego.eco, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.218 port 40518 connected with 192.168.10.22 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   113 MBytes  94.6 Mbits/sec
root@buchner:~#
```



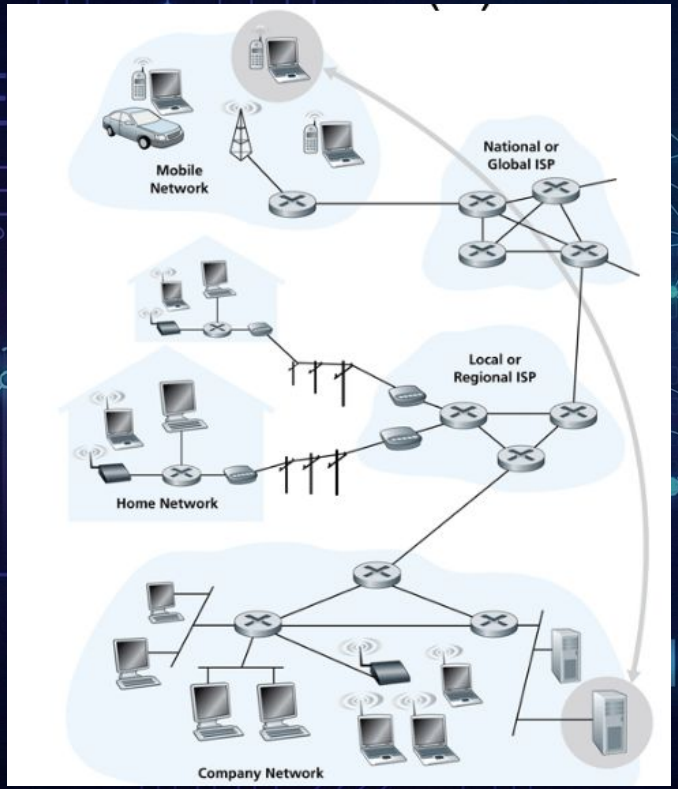
# Unit of measurement

- **Data transmission speed = amount of information / transfer time**
- In general, this speed is expressed in bits per second, i.e., **bit/s** (or **bps** , also called **bit rate ). The byte per second , byte/s** (or **Bps** ), is also used .
- Then we use the standard prefixes **k (=kilo 10 $^{3}$ ), M (=mega 10 $^{6}$ ), G (=giga 10 $^{9}$ )** , therefore not the approximations based on powers of two that are used in computer science.
- Converting from **bps to Bps is simple, just divide by 8.** For example, ADSL **10 Mbps = 10 Mbps/8 = 1250 KBps**
- **10 MiB** file with a **5 Mbps** line , it will take approximately **(10 * 1024 * 1024 * 8) / 5 10 $^{6}$ = 16.8 s** ( **ignoring** latency ) **.**


COMPUTER SCIENCE
COMPUTER NETWORKS
STATISTICS
MACHINE LEARNING

# Computer Networks

A set of connected autonomous computers, the network is seen as **providing logical channels** through which various applications can communicate with each other.

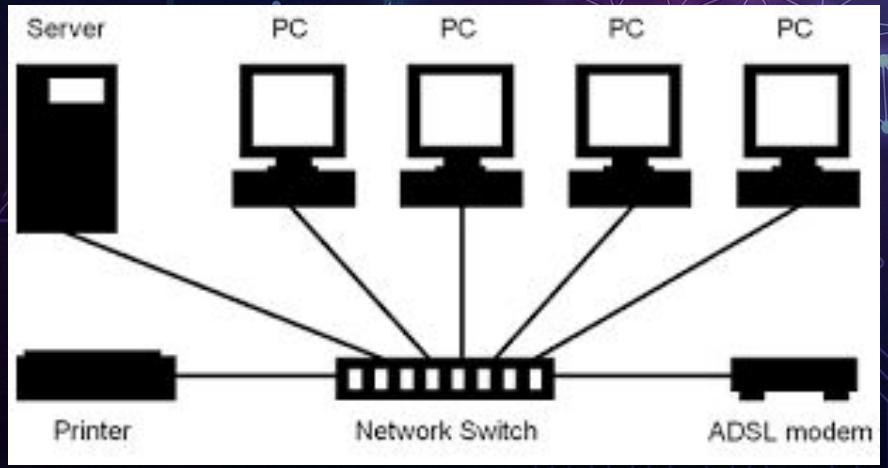


# Computer Networks

- A set of connected autonomous computers
  - **Nodes (hosts)** can be either servers or desktop PCs, or mobile devices or other
  - Connections are the channels that allow nodes to communicate. Obviously, they can be **transmitted using very different means, such as twisted pair telephone wire, optical fiber, or radio channels (wireless).**
    - I can have direct or indirect connections ( **switches** )

# Switch

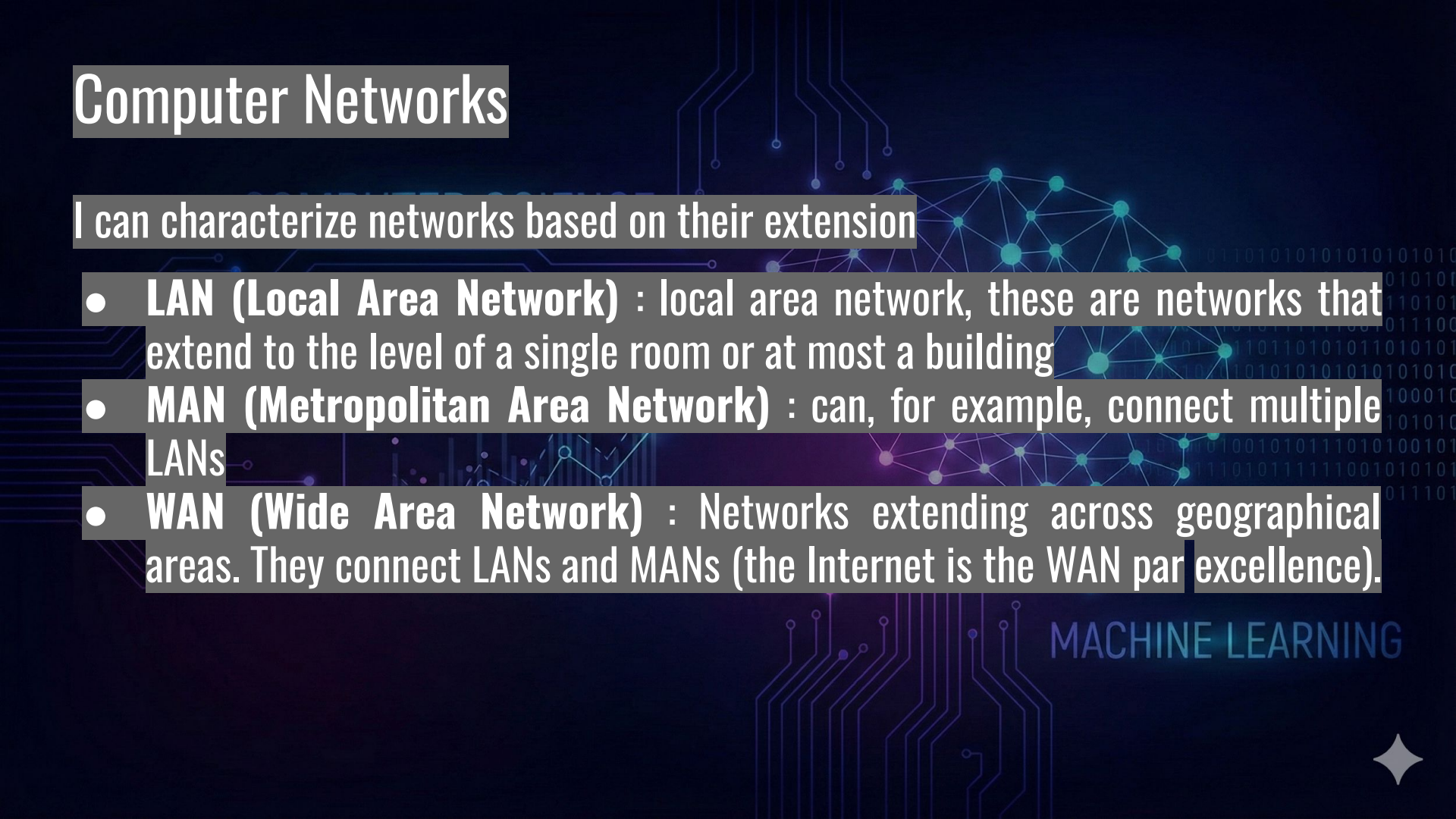
A network switch (also called a switching hub or bridging hub) is a network device that connects computer devices together using packet **switching to receive, process ,** and forward data to the destination device.

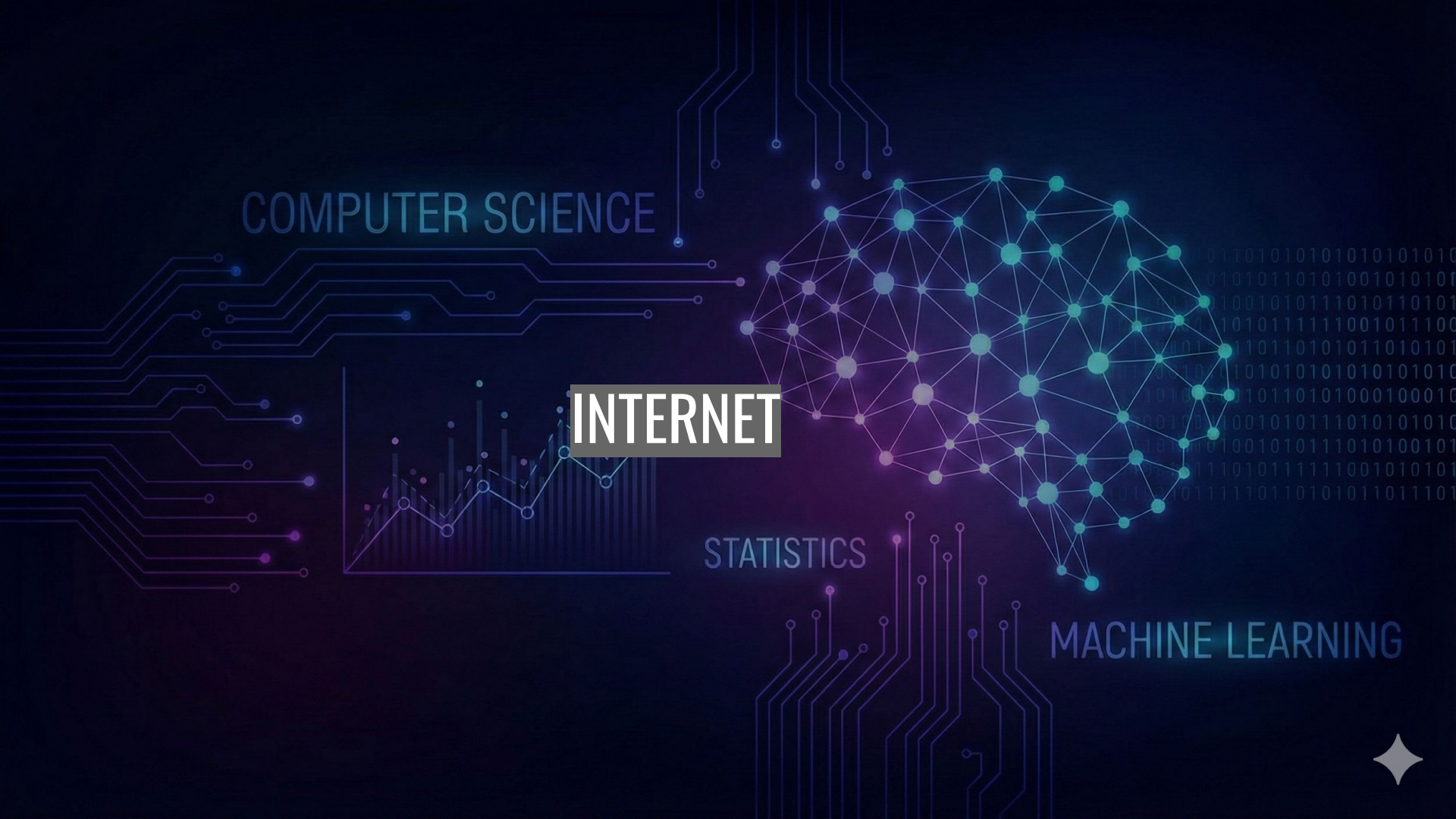


# Computer Networks

I can characterize networks based on their extension

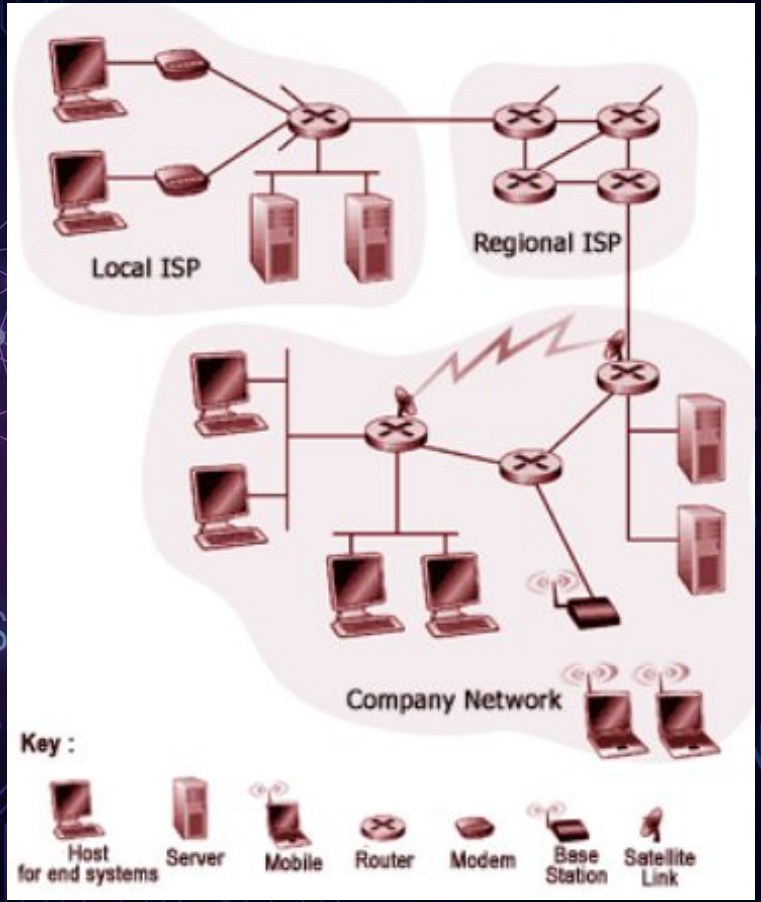
- **LAN (Local Area Network)** : local area network, these are networks that extend to the level of a single room or at most a building
- **MAN (Metropolitan Area Network)** : can, for example, connect multiple LANs
- **WAN (Wide Area Network)** : Networks extending across geographical areas. They connect LANs and MANs (the Internet is the WAN par excellence).




COMPUTER SCIENCE
INTERNET
STATISTICS
MACHINE LEARNING

# Internet

We can describe the internet first of all **from the point of view of the basic components** .
The Internet interconnects millions of devices around the world, including traditional desktop PCs, mobile devices, and servers (which store and transmit data, such as HTML pages, emails, etc.). These devices are called **hosts or end systems .**

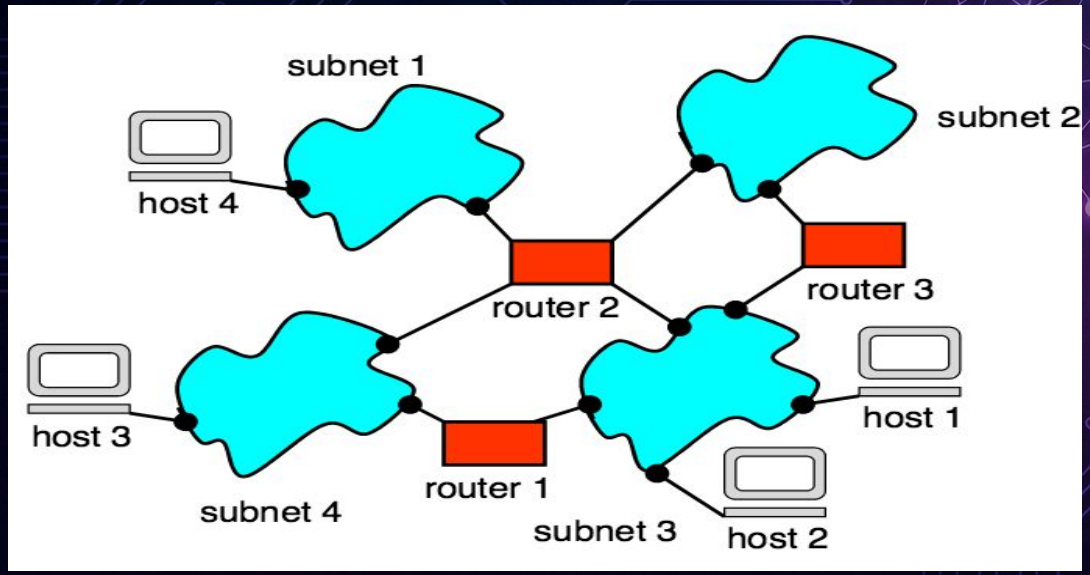


# Internet

- The various hosts are interconnected via communication channels. These channels can be of very different nature, such as coaxial cables, copper wires, optical fibers, or radio links.
- **Generally, hosts are not directly interconnected, but there are "switching" devices called routers** . **Routers are used to route data traffic** , and therefore take information arriving from one channel and send it to another channel.
- The Internet is a set of interconnected networks. **The various hosts, as well as other infrastructure devices, communicate with each other using common protocols** (the two main protocols are IP Internet Protocol and TCP Transmission Control Protocol).

# Internet

- The Internet today interconnects thousands of subnets




**Host** : computer connected to the Internet, it can be either a client or a server at the application level

**Router** : node used to route traffic ( **a Layer 3 network gateway device** ,)

**Subnet** : A set of hosts between which there is a layer 2 connection (for example, a LAN)

**To date, order of billions of Hosts**

# Internet: A Brief History

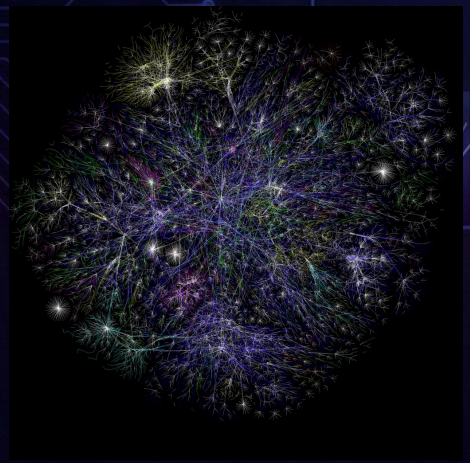
- **In 1969, the design of the ARPANET military network began** . Among other things, this network was designed to withstand nuclear attacks. In fact, it was capable of connecting interconnected devices, following different paths in the event of a failure.
- **1972 saw the birth of electronic mail (e-mail), file transfer via FTP, and remote login.**
- **1974 The IP and TCP** protocols were **officially introduced** .
- **1979**, the birth date of the **CSNet network** that interconnects **universities and research centers in the United States**

# Internet: A Brief History

- **1982 ARPANET and CSNet** are connected, this is considered the **official birth date** of the Internet in some ways
- **1990: NSFNet, a network connecting supercomputers, replaces the Arpanet.** This paves the way for civilian and commercial uses of the Internet.
- **1990** Same year **Tim Bernes-Lee who then worked at CERN in Geneva invented HTML** (Hyper Text Markup Language) which allows the management of information of different nature, text, images etc. This is the first step towards the WWW (World Wide Web)

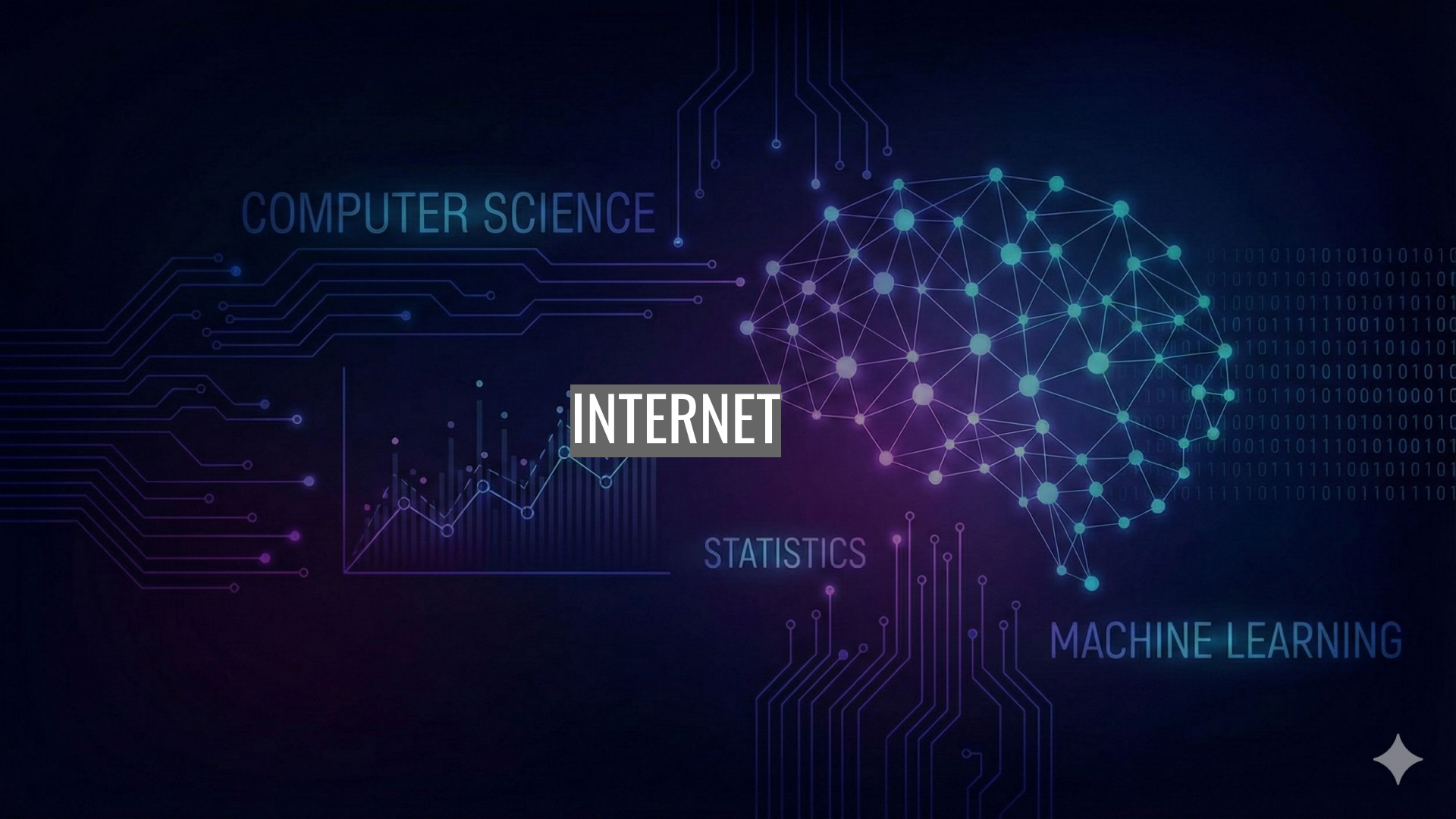
# Internet: A Brief History

- **1993 Mosaic,** the first browser, was created
- **In 1994, Yahoo!,** the first search engine, was born. In the second half of the 1990s, many others were created, leading to **the birth of Google in 1998** .



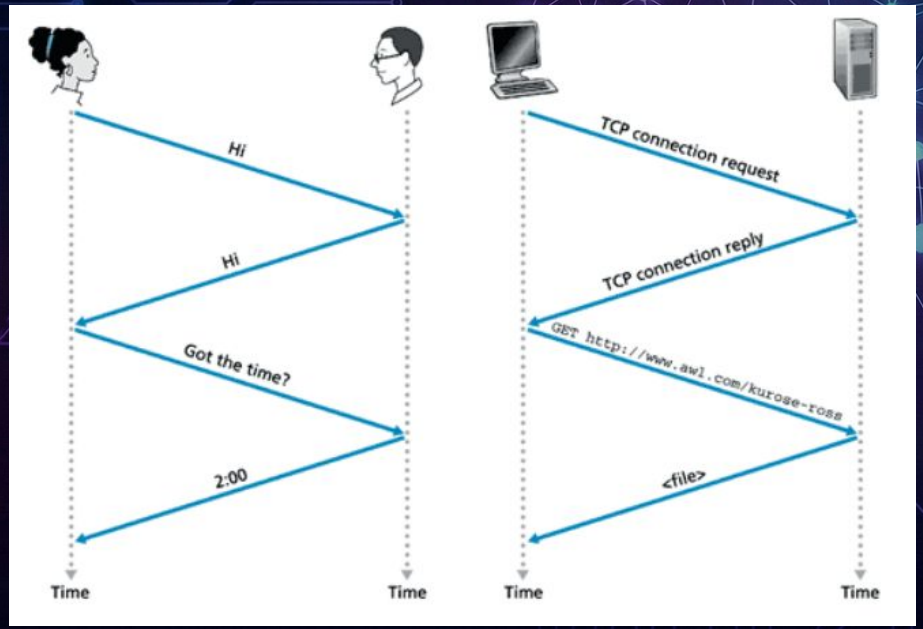
The Internet today contains approximately **10 billion computers, each computer (device) containing on average a couple of billion transistors. Therefore, the Internet interconnects $10^{19}$ ; in conclusion, there are 10,000 times more transistors than synapses in the human brain.**

Today, the Internet is used for selling goods and services. Audio and video transmission, online collaboration, work, online gaming, and social interaction.


COMPUTER SCIENCE
INTERNET
STATISTICS
MACHINE LEARNING

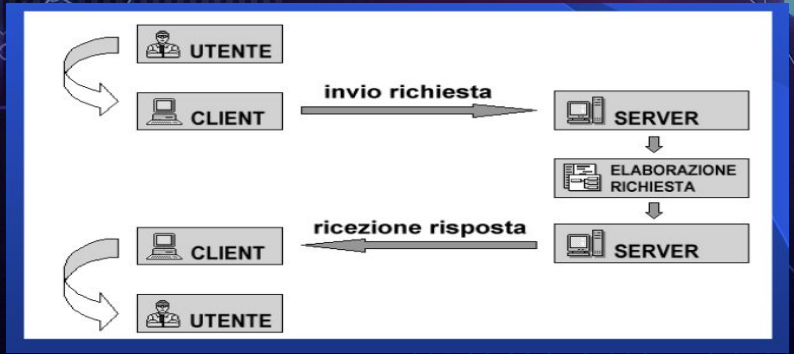
# Communication protocol

- A set of rules (formally described) that define the methods of communication between two or more entities.



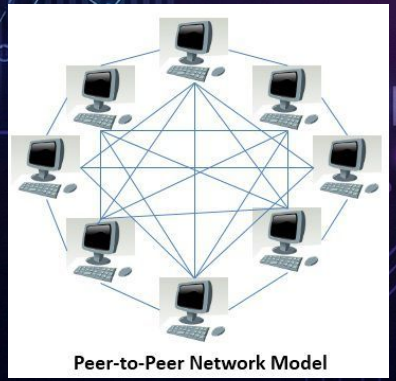
# Network Model: Client-Server

- Most of the telematic services offered by the Internet are based on the **client/server interaction mode (different from P2P) .**
- The client is equipped with special client software capable of sending service requests to a specific server. The client formats the requests in a way that is appropriate and understandable to the server, using a specific protocol (e.g., a web browser or server).

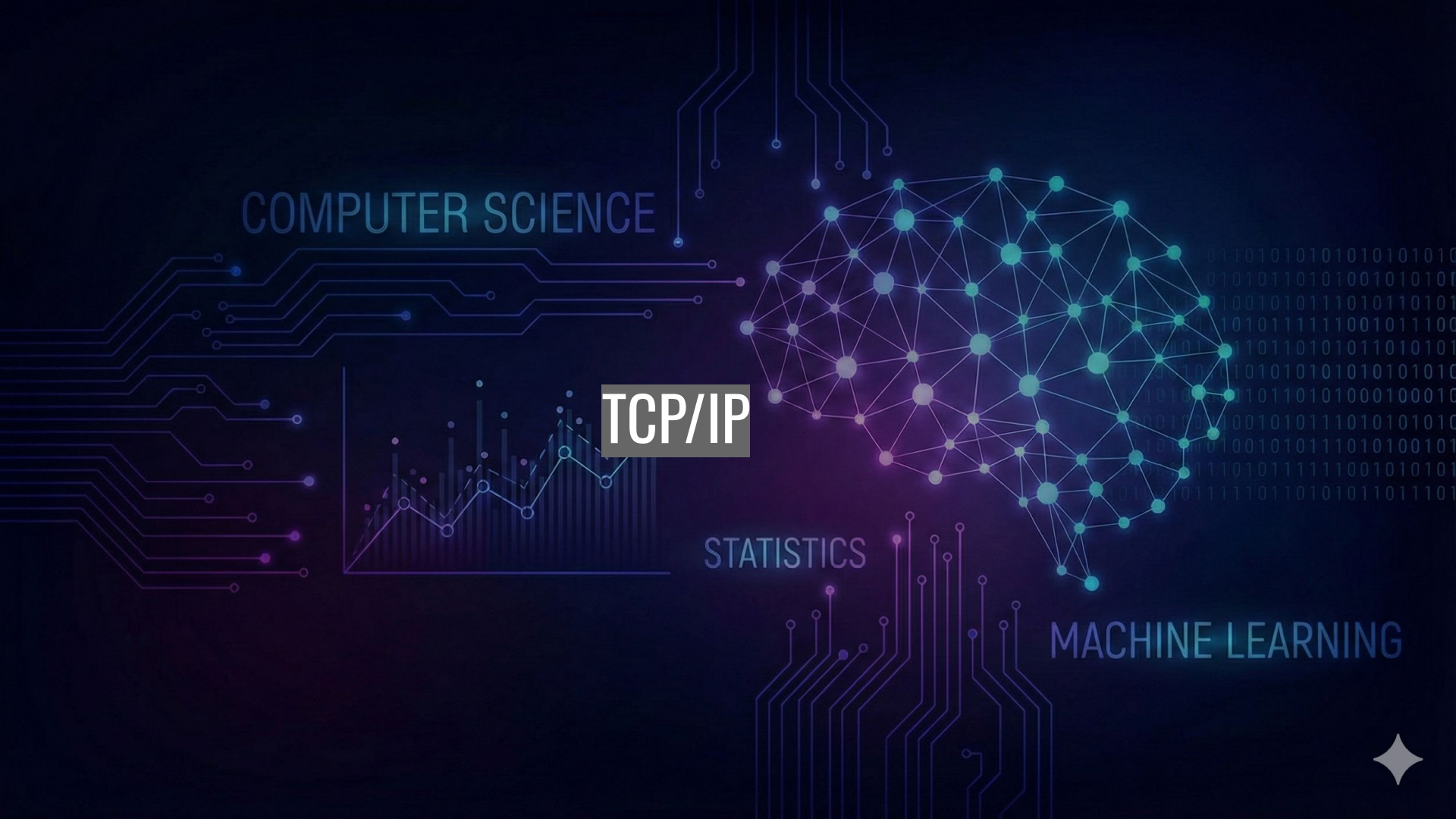


# Network model: P2P Peer-to-Peer

- In this case **there is no distinction between client and server, each node can instead act as either client or server** depending on whether the single node is requesting or providing data.
- To become part of the **P2P network after joining the node must start providing services and will be able to request services** from other nodes (e.g. BitTorrent)

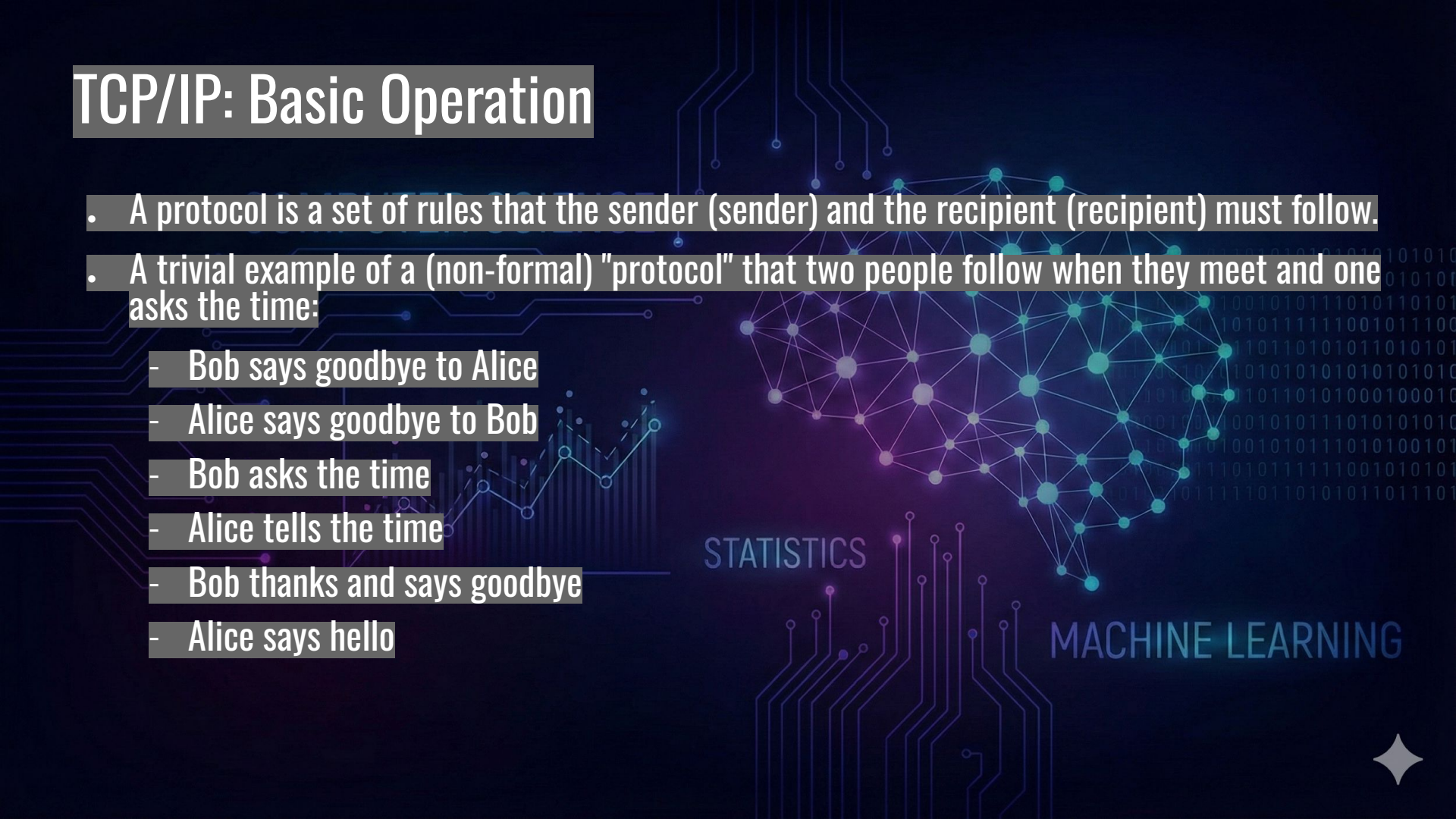


Peer-to-Peer Network Model


COMPUTER SCIENCE
TCP/IP
STATISTICS
MACHINE LEARNING

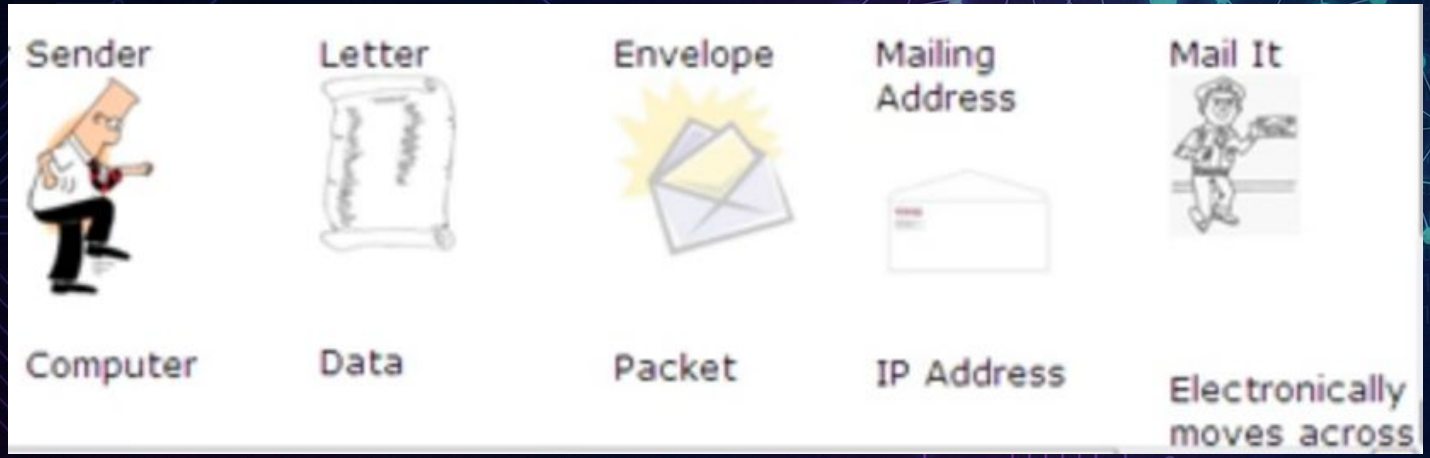
# TCP/IP: Basic Operation

- A protocol is a set of rules that the sender (sender) and the recipient (recipient) must follow.
- A trivial example of a (non-formal) "protocol" that two people follow when they meet and one asks the time:
  - Bob says goodbye to Alice
  - Alice says goodbye to Bob
  - Bob asks the time
  - Alice tells the time
  - Bob thanks and says goodbye
  - Alice says hello



# TCP/IP: Basic Operation

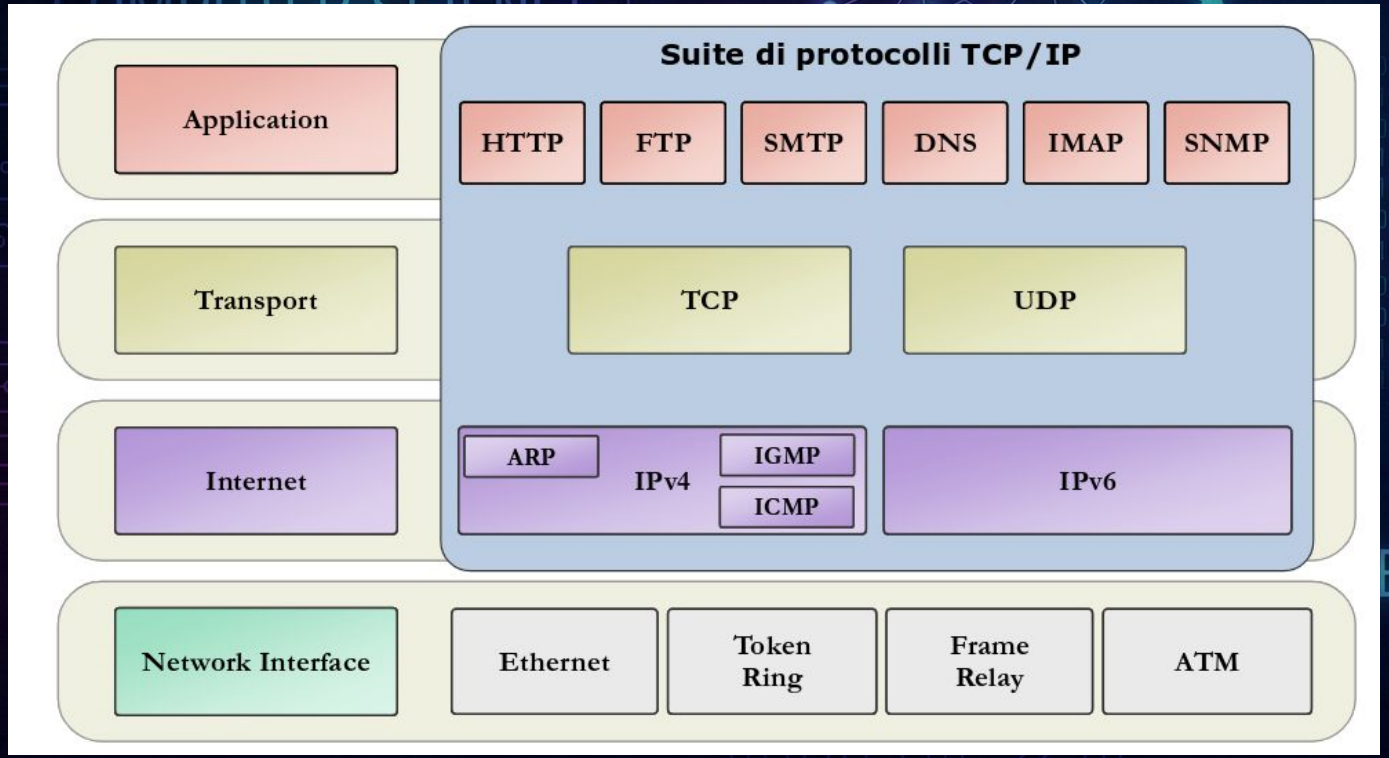
COMPUTER SCIENCE




MACHINE LEARNING

**And finally a receipt (ACK)**

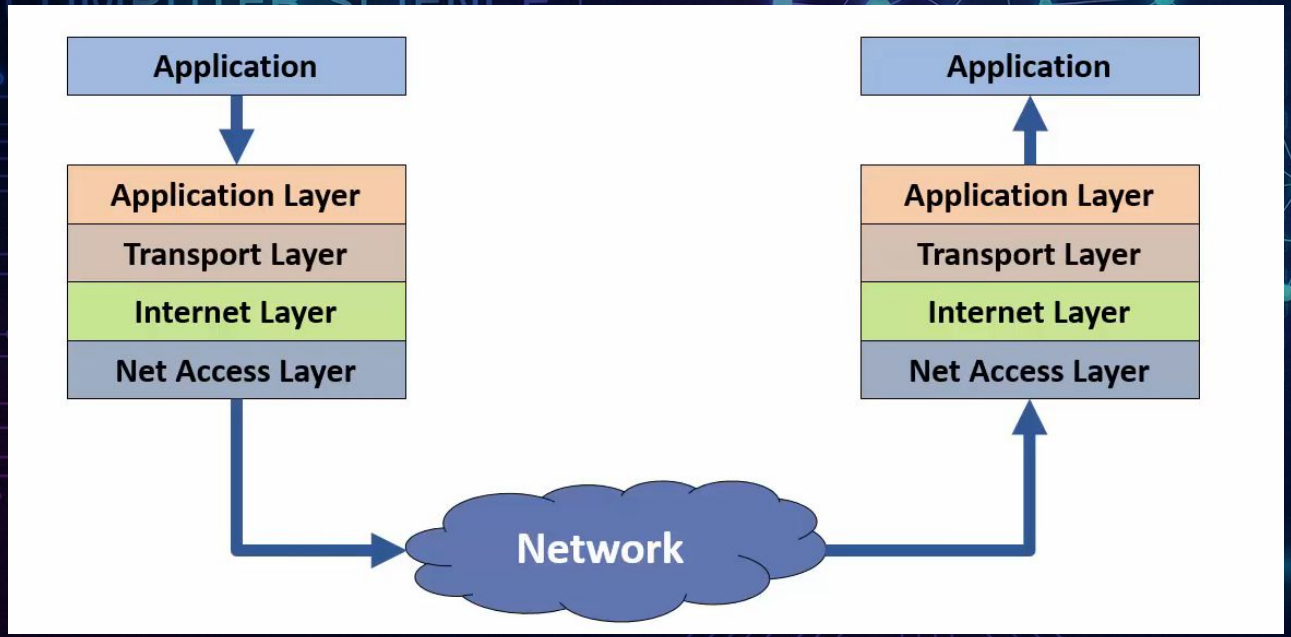
# TCP/IP Architecture



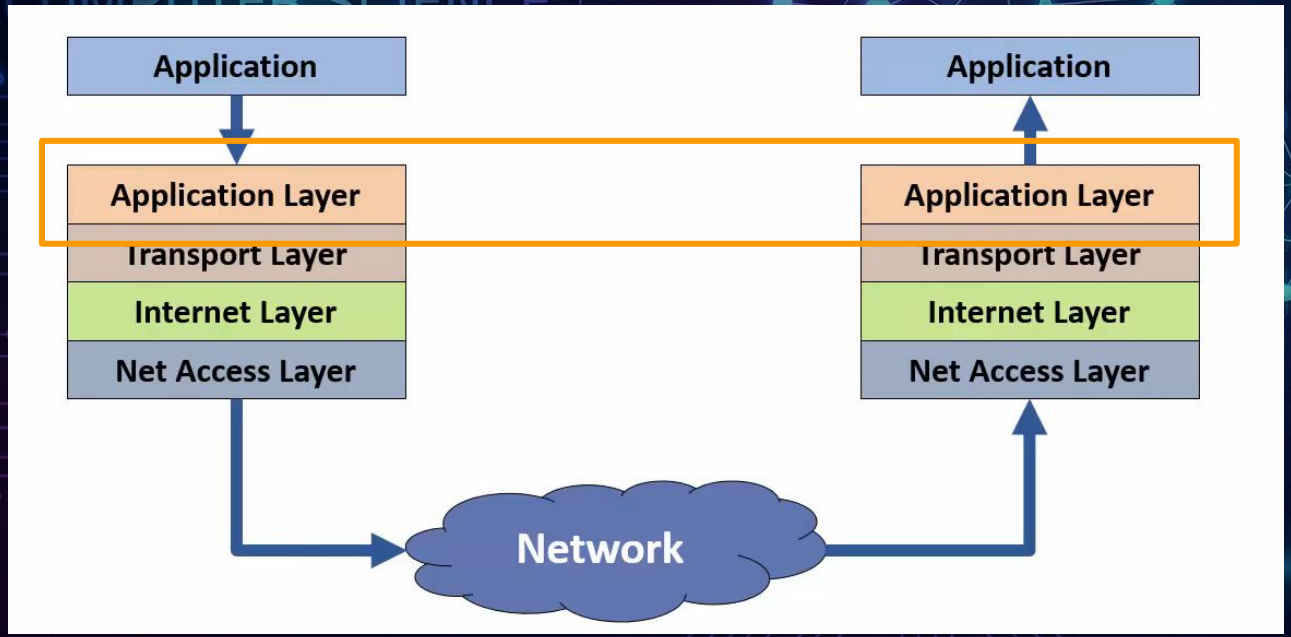
# TCP/IP: Basic Operation

- The idea of **layering** . For example, the **HTTP protocol is built on top of TCP** . A browser doesn't have to worry about how TCP is implemented, it just needs to know that it works.
- **The data that the transport layer receives from the application layer is fragmented into packets.** The data packets are reassembled at their destination ( **each packet can follow different paths** ).
- **TCP adds additional information to each packet** , such as the sequence number of which the packet is part.
- The packet is then passed to the network layer where **IP routes the packets to the destination host** in the most appropriate way.

# TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

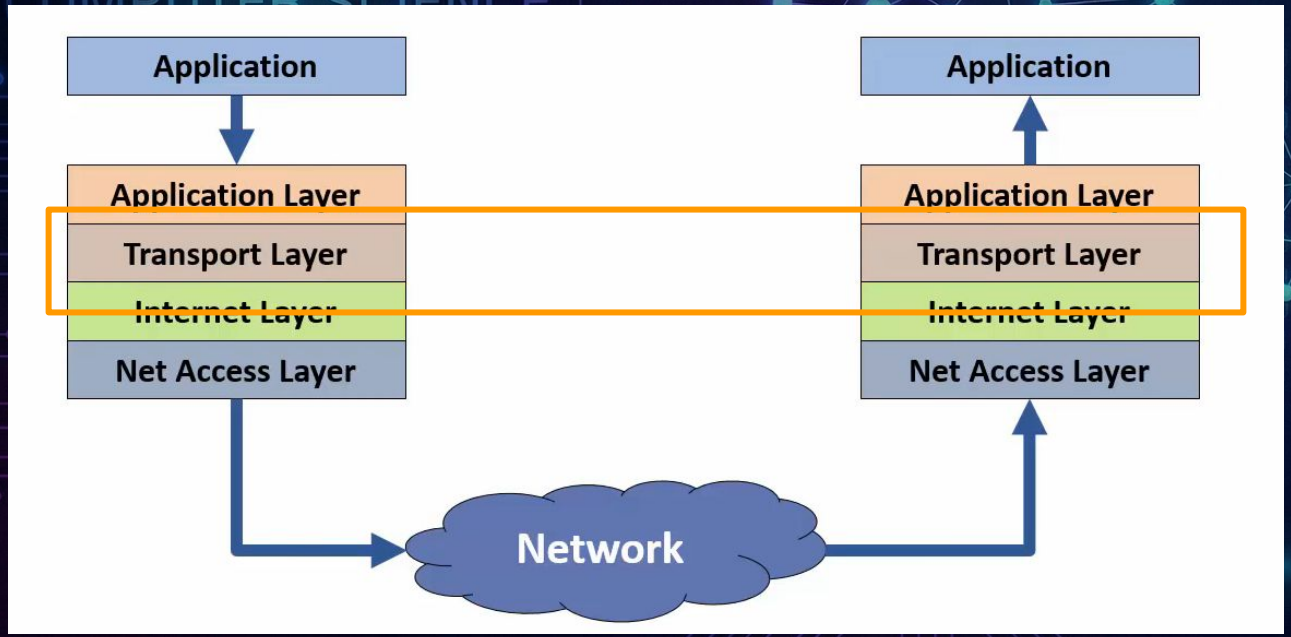
# TCP/IP


Application
Application
Application Layer
Application Layer
Transport Layer
Transport Layer
Internet Layer
Internet Layer
Net Access Layer
Net Access Layer
Network

# The TCP/IP STACK: Application Layer

- **Application layer** : we find numerous protocols such as HTTP, FTP, DNS, SMTP, etc. etc. At this level two applications exchange **messages** without worrying about how these will be delivered.
- This is the interface with the user, for example if we consult a web page the protocol manages the interaction session between our browser (client) and the web server.

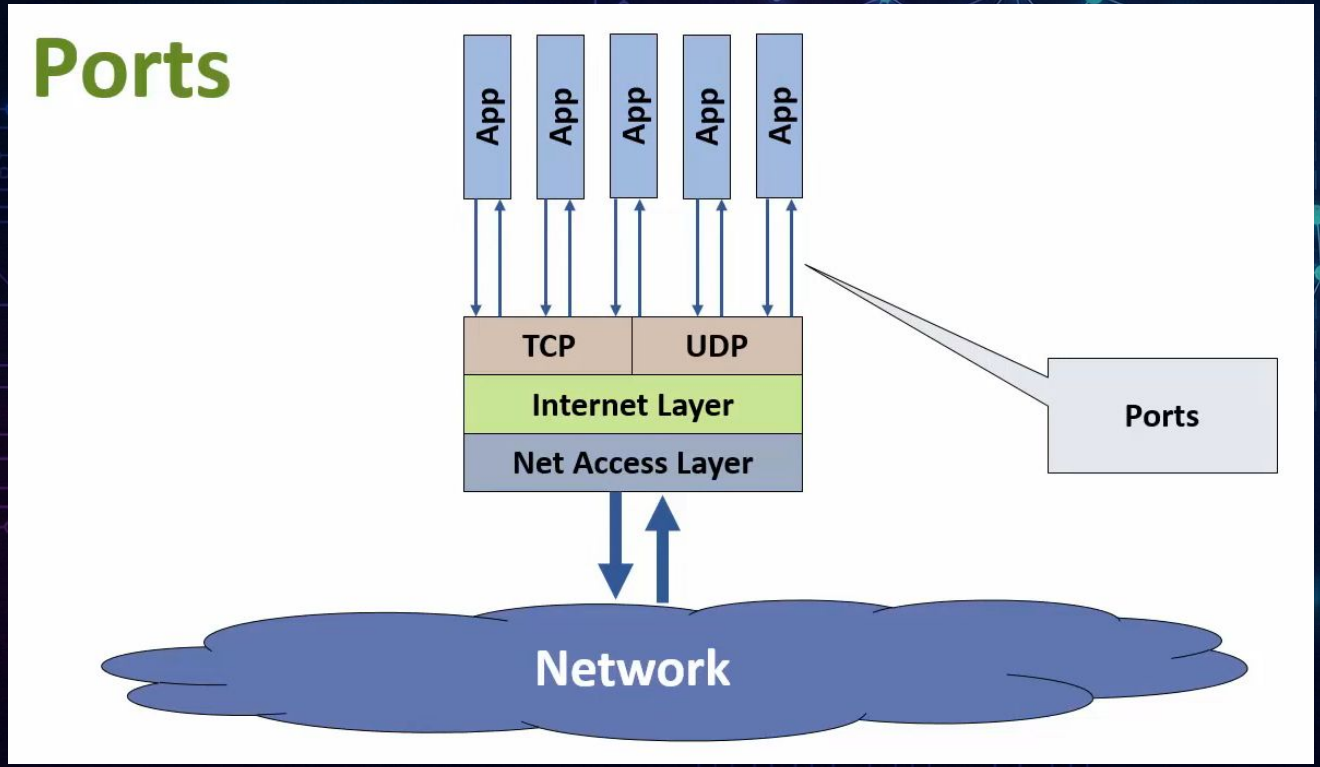
# TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

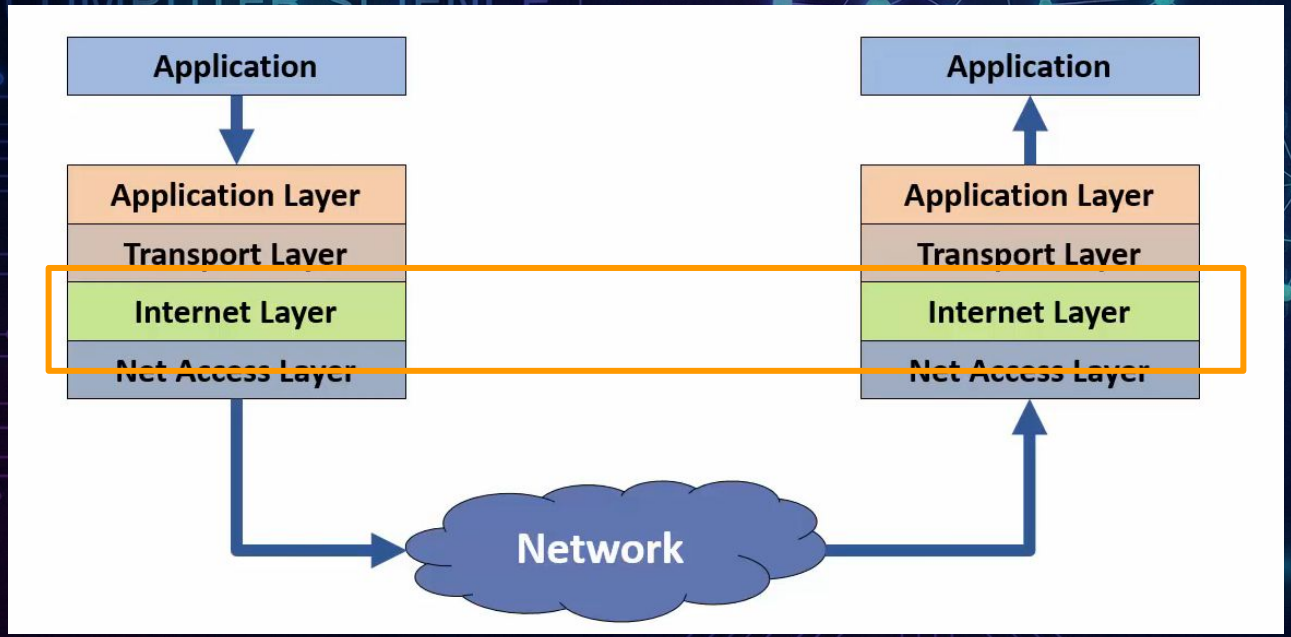
# The TCP/IP STACK: Transport Layer

- **Transport Layer** : At this level we find the two basic protocols **TCP** and **UDP,** at this level two hosts exchange segments.
- The protocols at this level **offer the transport service** to the application layer. For example, to manage **multiple sessions active at the same time, TCP and UDP use different port numbers** (logical ports).
- **TCP**
  - For each window of packets sent, TCP starts a time counter
  - The recipient sends an **ACK** if it has received the packet.
  - If the sender does not receive an **ACK** before the time expires (or...). The sender, for example, takes care of resend- ing the data

# The TCP/IP STACK: Transport Layer



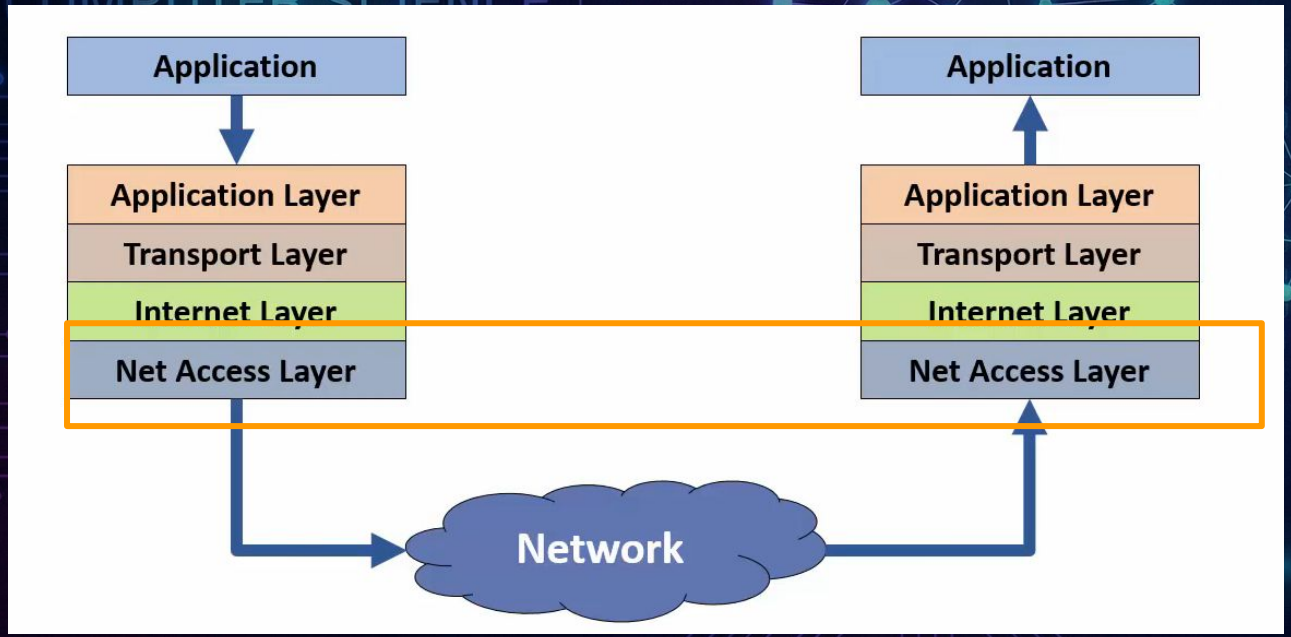
# TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

# The TCP/IP STACK: Network Layer

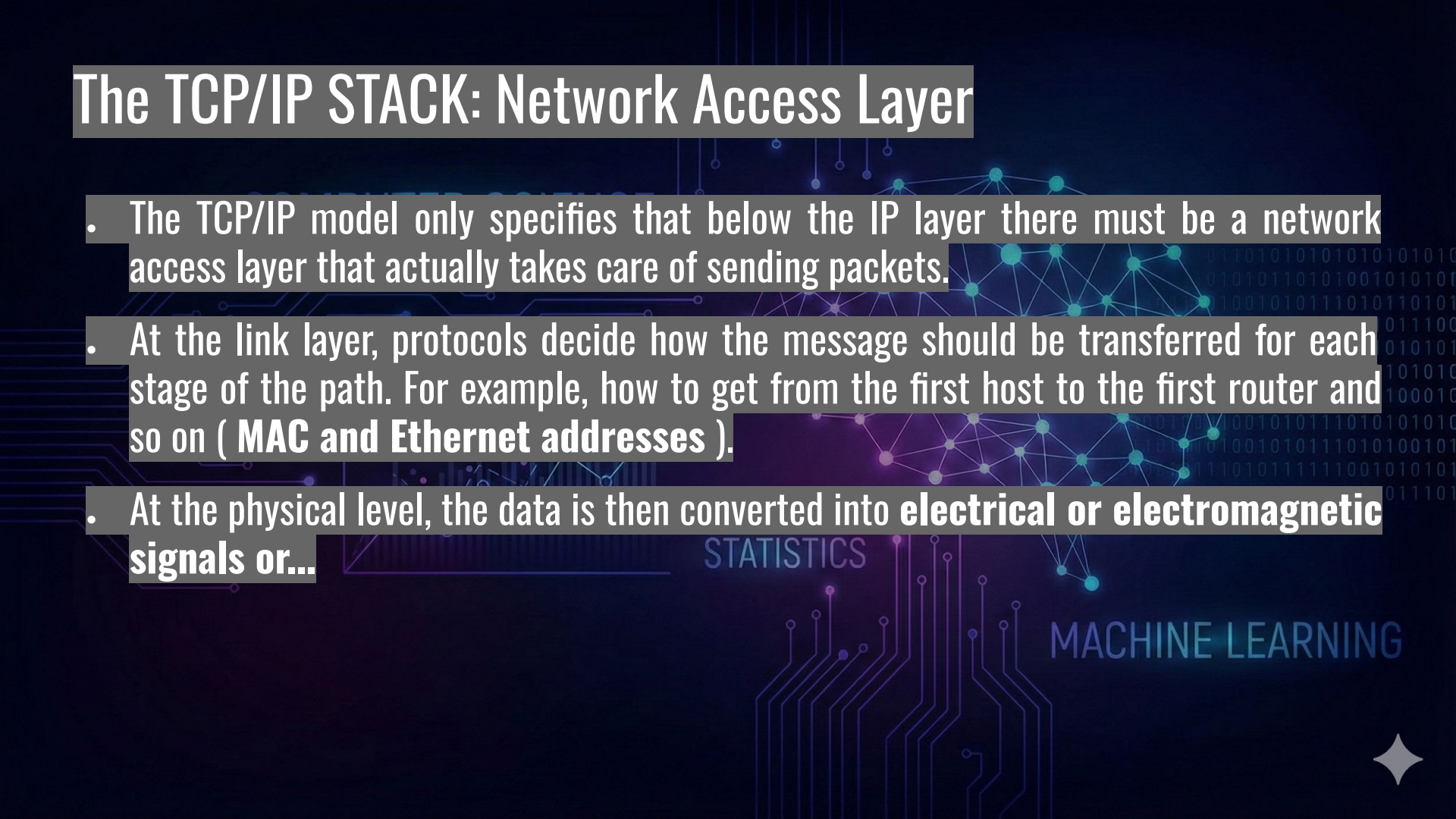
- **Network Layer** : At this layer, we find the **IP protocol. This protocol handles the addressing and routing of packets between sender and recipient.**
- **Addressing:** Each node is unambiguously identified by an **IP address.**
- **Routing:** This feature allows you to select the best path to follow to get the data from the sender to the recipient.

# TCP/IP

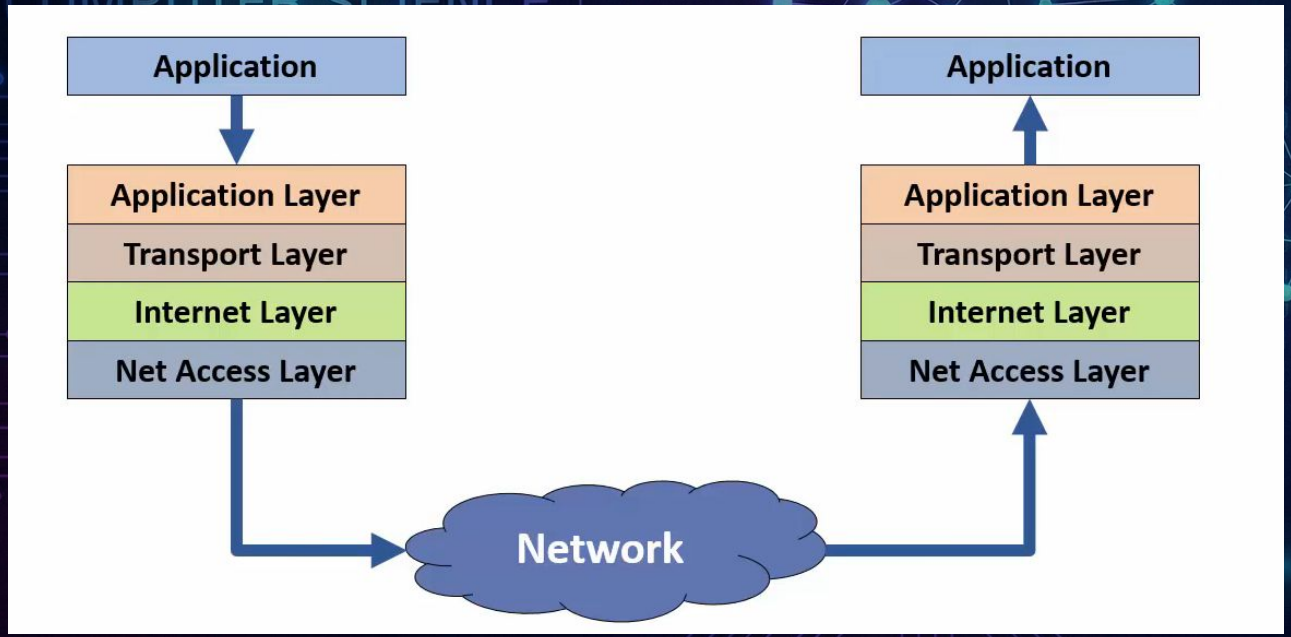

Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

# The TCP/IP STACK: Network Access Layer

- The TCP/IP model only specifies that below the IP layer there must be a network access layer that actually takes care of sending packets.
- At the link layer, protocols decide how the message should be transferred for each stage of the path. For example, how to get from the first host to the first router and so on ( **MAC and Ethernet addresses** ).
- At the physical level, the data is then converted into **electrical or electromagnetic signals or...**

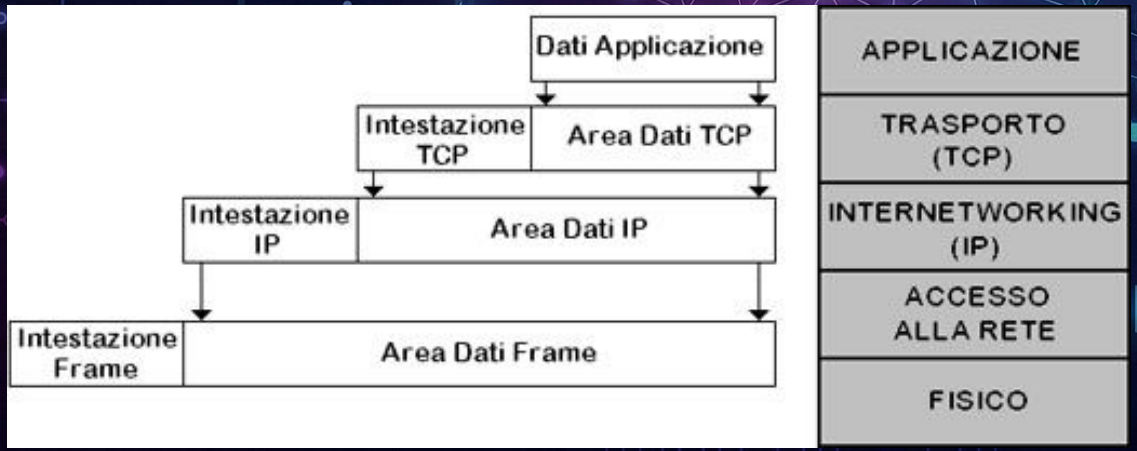


# TCP/IP

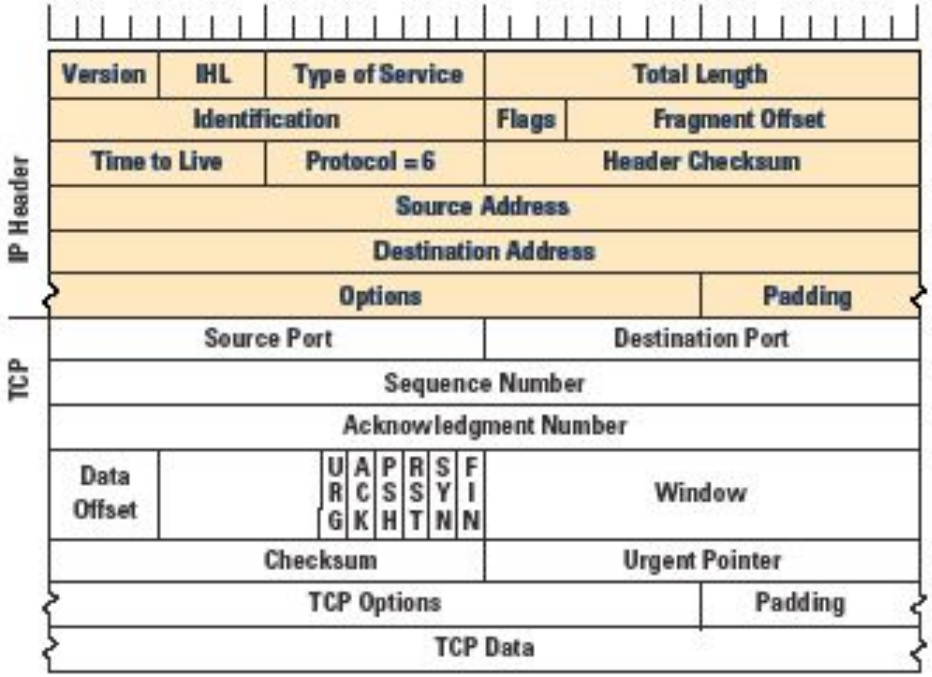

Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

# The TCP/IP STACK

- When an application needs to send data, it is passed down the layer, one at a time, until it reaches the underlying physical network. Along the way, each layer adds information to the data, creating a "network frame" (encapsulation):

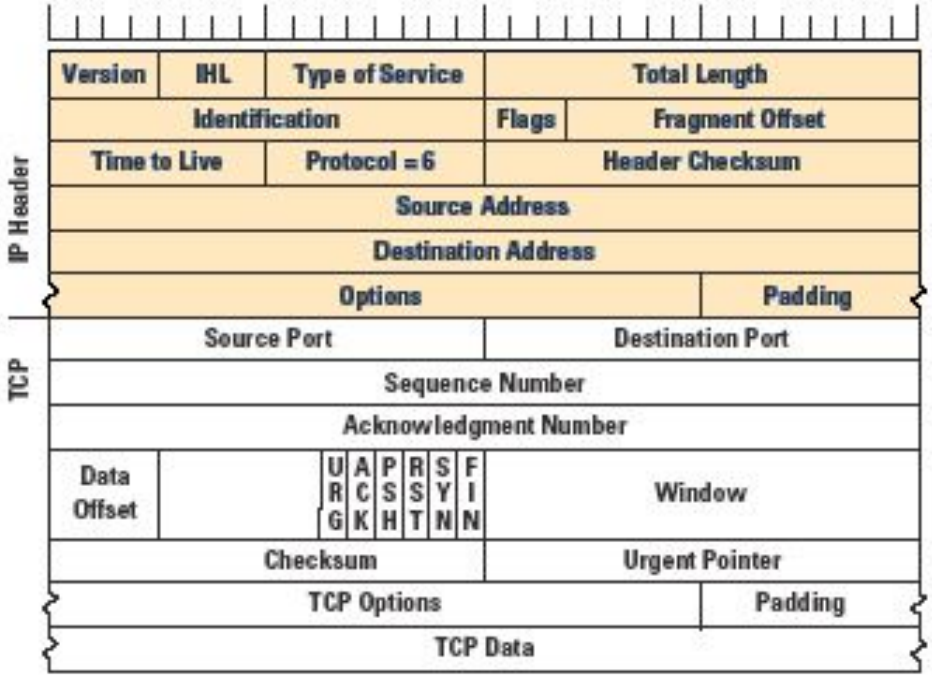


# TCP/IP: headers

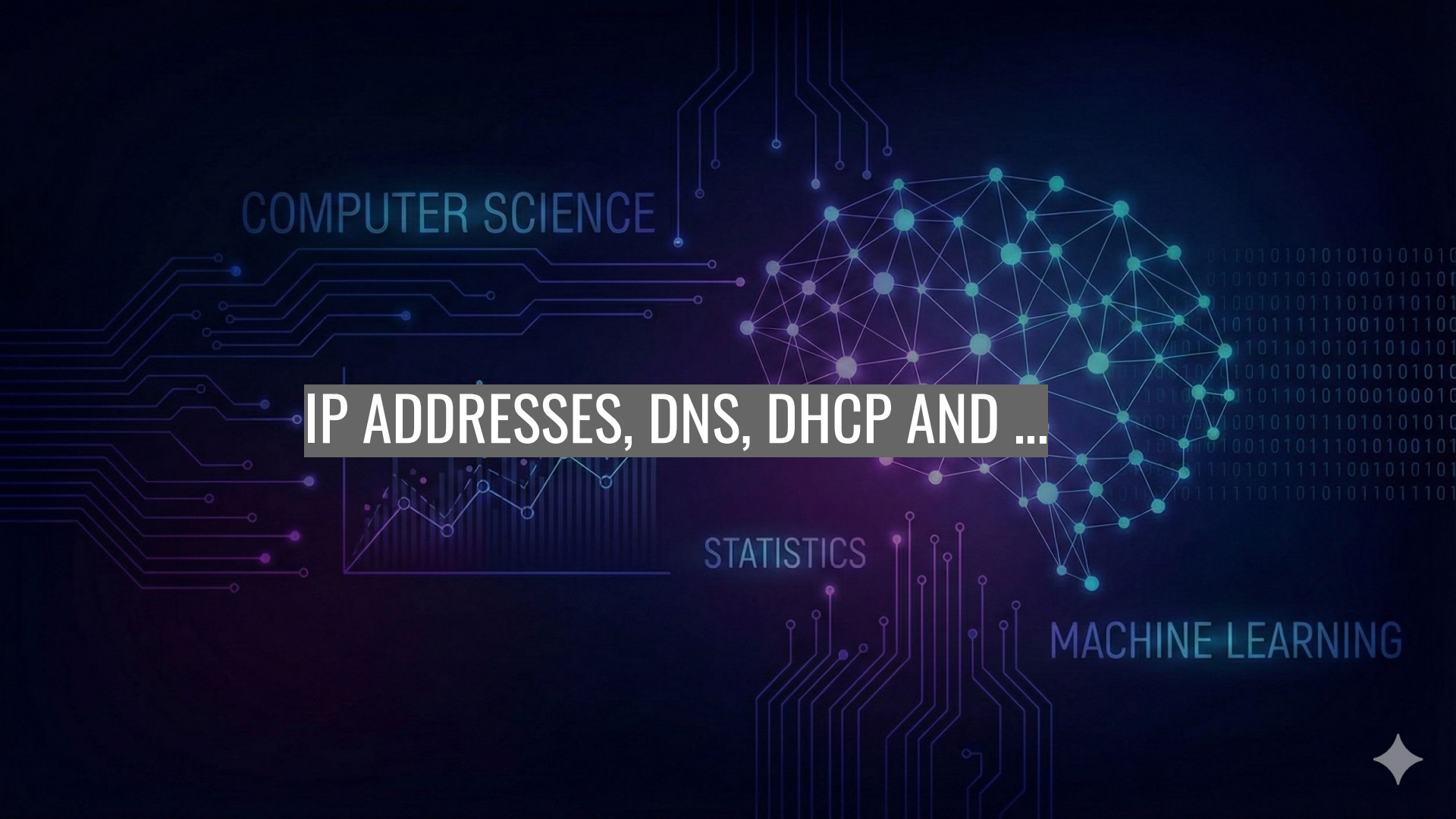



The TCP Header (The Logistics): This is the inner layer. It manages the specific conversation between applications. It uses Port Numbers to ensure the data goes to the correct program (like Chrome vs. Spotify) and Sequence Numbers to put the packets back in the correct order if they arrive scrambled.

# TCP/IP: headers




The IP Header (The Address): This is the outer layer. It handles the routing across the network. It contains the Source and Destination IP Addresses to ensure the packet finds the correct physical machine in the vastness of the internet.


COMPUTER SCIENCE
IP ADDRESSES, DNS, DHCP AND ...
STATISTICS
MACHINE LEARNING

# IP addresses

- **Each computer connected to the Internet is identified by its IP address** , which is made up of **four groups** of one byte each ( **32 bits in total** ). Each number can take values from **0 to 255.**
- For example: **192.167.12.66** (static or dynamic IPs, private IPs, etc.)
- The last number usually identifies a host, the previous numbers the subnet to which this host belongs.
- **The maximum number of IPv4 addresses is therefore 256*256*256*256**
- **IPv6** : **128-bit** and therefore $2^{128}$ approximately 3.4 x $^{10,38}$ addresses ( **IoT: Internet of things** )

# IP addresses

## ADDRESSING: IPV4 VS. IPV6

### IPv4 (Legacy)

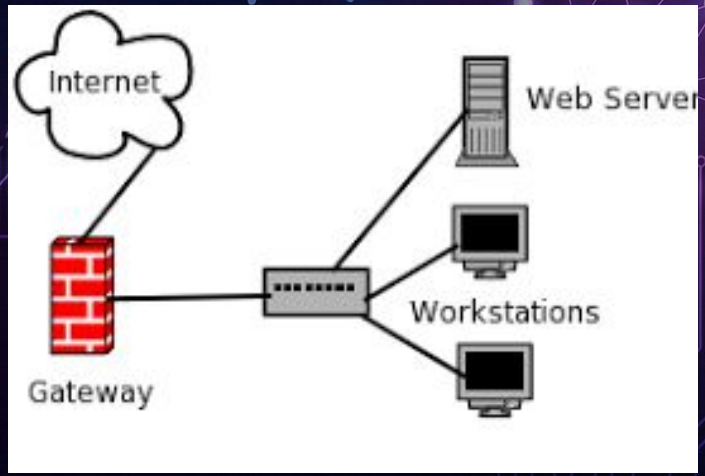
- **Structure:** 32-bit address (4 bytes).
- **Notation:** Dotted decimal (e.g., 192.168.1.1).
- **Capacity:** $2^{32}$ addresses (~4.3 Billion).
- **Limit:** Exhausted (Allocations depleted in 2011).

### IPv6 (Future Standard)

- **Structure:** 128-bit address (16 bytes).
- **Notation:** Hexadecimal (e.g., 2001:0db8::).
- **Capacity:** $2^{128}$ addresses.
- **Scale:** $3.4 \times 10^{38}$ (Undecillion).

# IP Addresses: Gateway

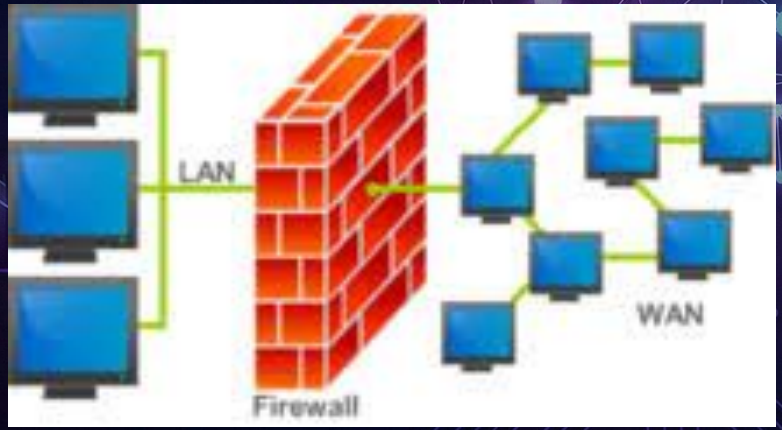
- The default **gateway is used to route packets to other destinations**
- **DHCP is** almost always used to automatically provide the client with the default gateway IP address.




MACHINE LEARNING

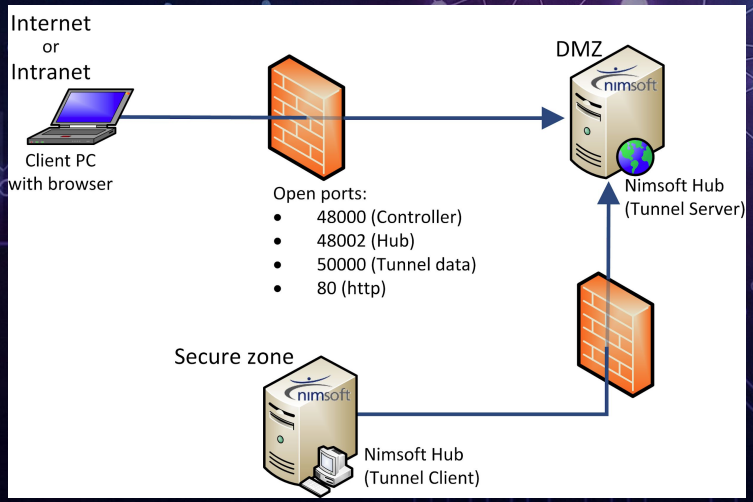
# IP addresses: firewall

- The **firewall** is a network security system that controls all incoming and outgoing traffic according to well-defined rules.



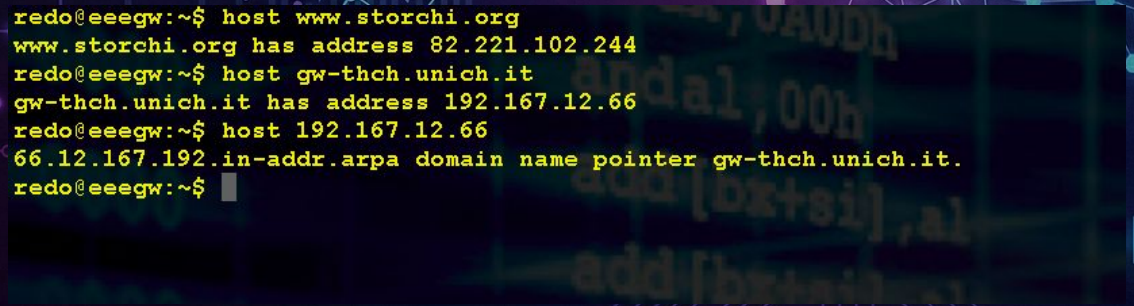
# IP Addresses: DMZ

- A **DMZ** (demilitarized zone) is a physical or logical subnetwork that contains and exposes an external organization's services to an unprotected network, usually a larger network such as the Internet.



# DNS

- It's difficult for humans to memorize numbers, much easier to memorize names. That's why there are DNS ( **Domain Name System** ) services. These are systems that are useful for translating names and addresses back and forth.

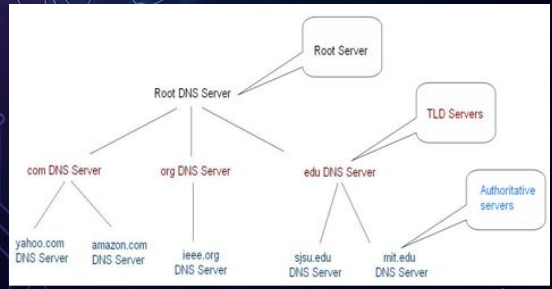


# DNS

- Each host is therefore identified by the user by a symbolic name:
  - gw-thch.unich.it
- Names are assigned uniquely and managed administratively in a hierarchical manner.
- Names uniquely identify a host within a domain:
  - it is the domain
  - unich is the subdomain within ad it
- The main domains are:
  - .gov .edu .com essentially in the USA associated with the type of organization
  - The various countries instead have domains of the type: .it, .uk, .fr, .de ....

# DNS

- Before the introduction of the DNS system, the correspondence between IP addresses and names was managed by the SRI-NIC which essentially maintained a list in a hosts.txt file.
- In practice, **DNS is a distributed database** . The information is distributed across many computers, DNS servers, **each of which is responsible for a certain portion of the name, called the domain** .

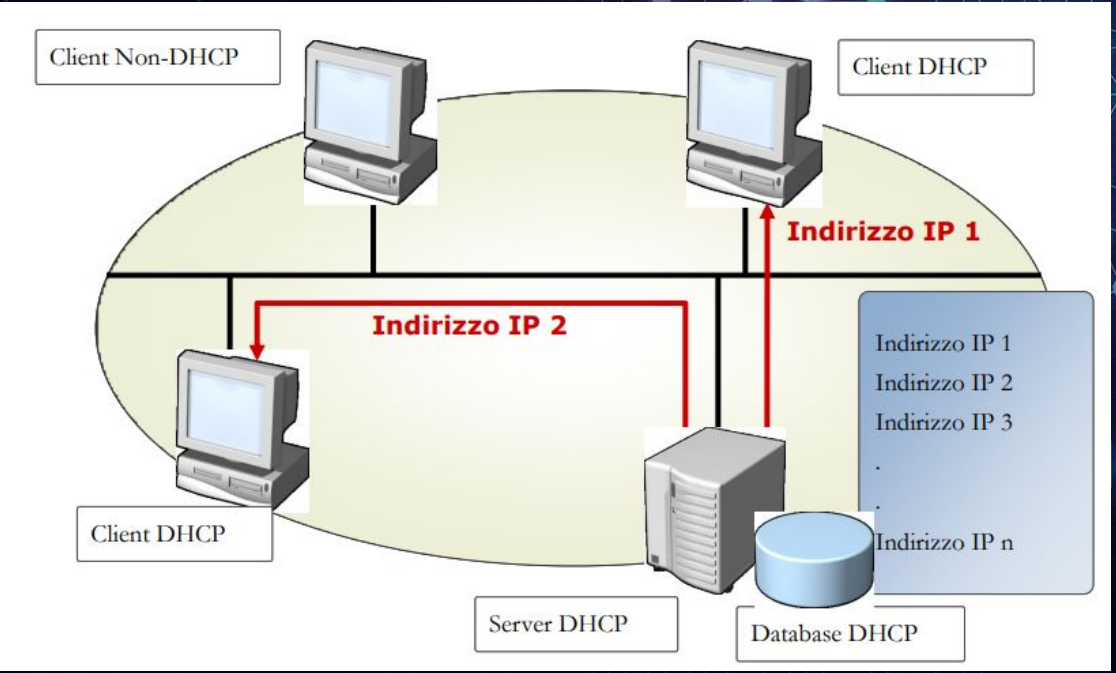


# DNS

- Servers are organized in a **hierarchical tree structure**
- When requesting a given address, for example www.storchi.org , the DNS server of “your network” checks whether the address corresponding to the **name is present in the cache** .
- If it is not present, **contact the Root servers, which provide the address for the TLD servers (like .org or .it).**
- **This last server will return the IP address corresponding to “WWW” (Domain Name System (DNS) names are "case insensitive")**

# DHCP

- For example the ADSL router you have at home



# High-level protocols

- Different types of protocols are used, each for a specific service:
  - **HTTP (HyperText Transfer Protocol)** Access to hypertext pages (WEB) within the WWW (encrypted https)
  - **FTP (File Transfer Protocol)** transfer and copy files
  - **SMTP (Simple Mail Transfer Protocol)** Sending email messages POP3 **for** downloading email messages to your computer **IMAP is useful when checking messages from multiple devices**

A resource on the network is therefore “identified” by the URL:

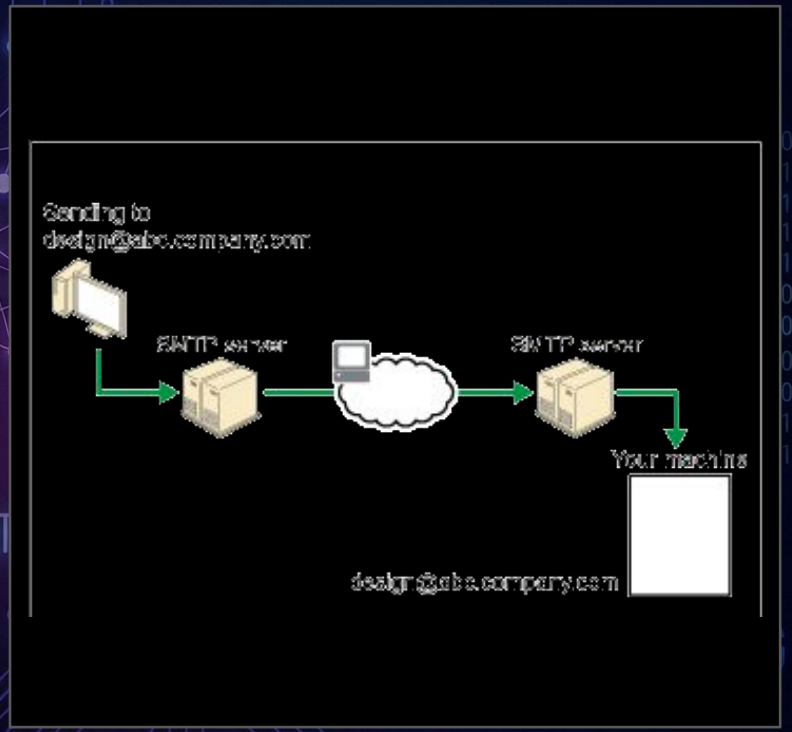
http://hostname.it/index.html

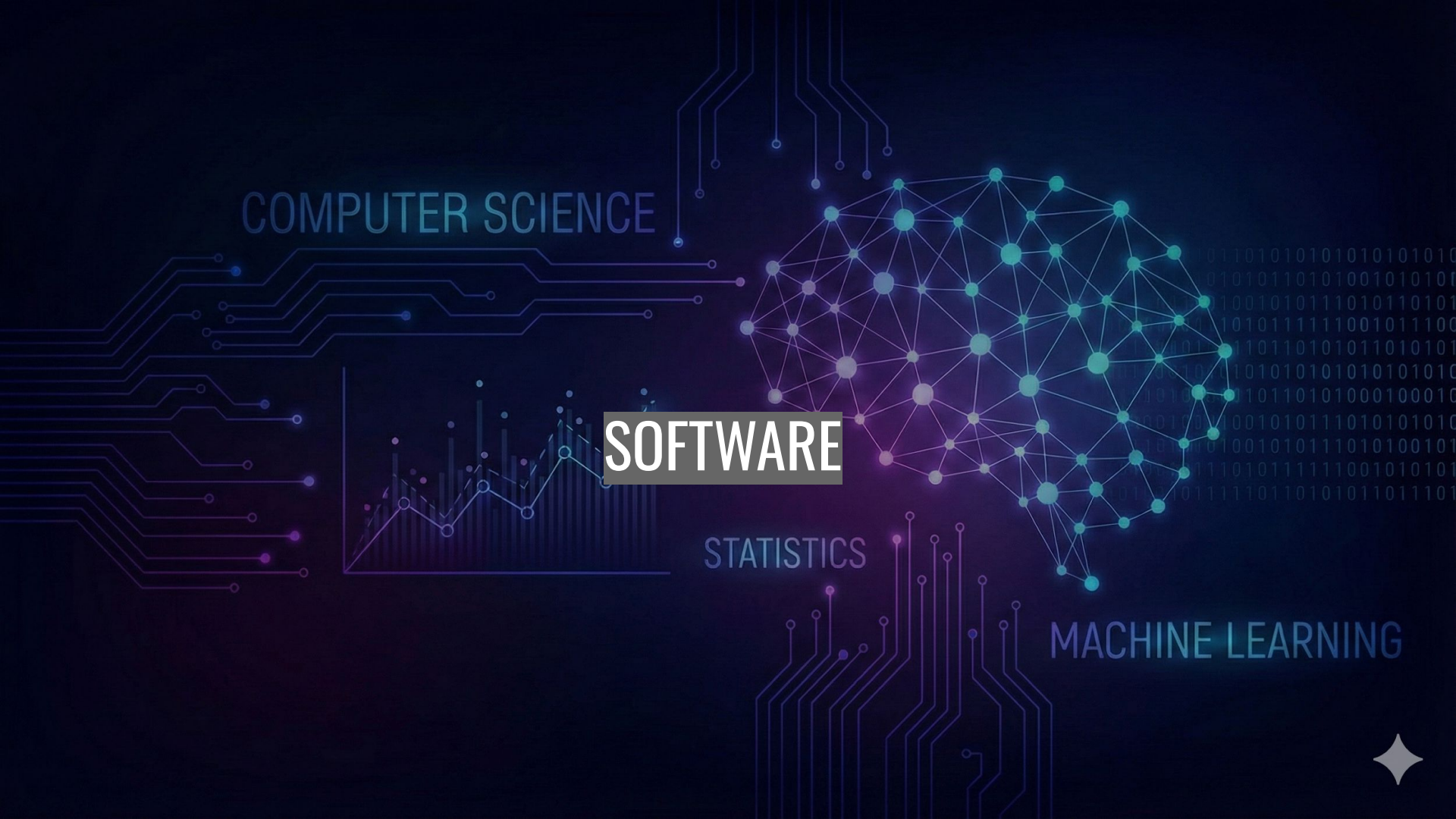
# SMTP example

## SMTP Protocol Exchange

```
S:   250 SMTPUTF8

C:   EHLO example.com

S:   250-mx.google.com at your service, [999.999.999.999]
S:   250-SIZE 35882577
S:   250-8BITMIME
S:   250-AUTH LOGIN PLAIN XOAUTH XOAUTH2 PLAIN-CLIENTTOKEN
S:   250-ENHANCEDSTATUSCODES
S:   250-PIPELINING
S:   250-CHUNKING
S:   250 SMTPUTF8

C:   AUTH XOAUTH2 dXNlcj1hbWFnYWtpLnRv...

S:   235 2.7.0 Accepted
```

Specify Initial Client Response which is created from username and access token

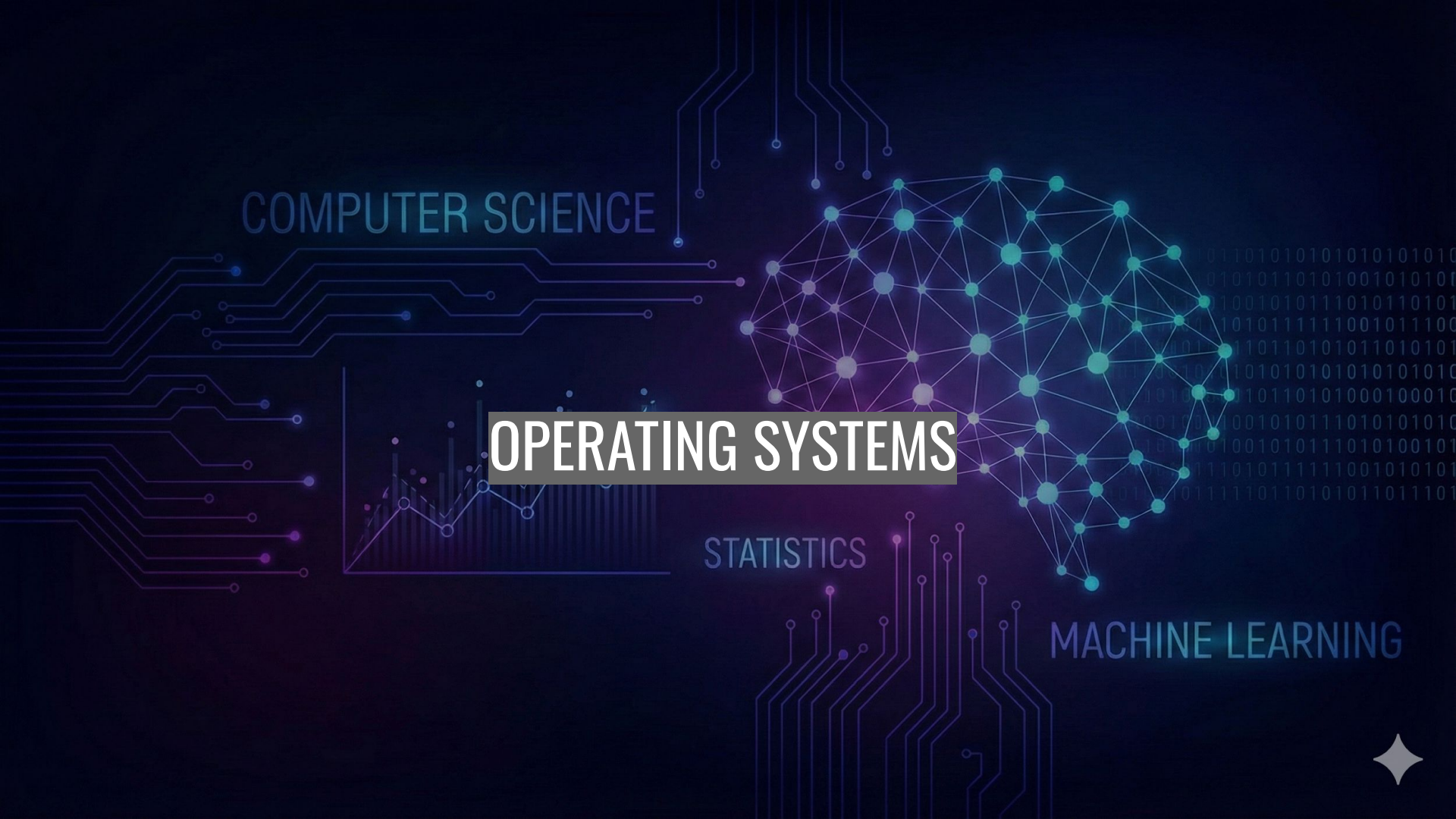



COMPUTER SCIENCE
SOFTWARE
STATISTICS
MACHINE LEARNING

# Software

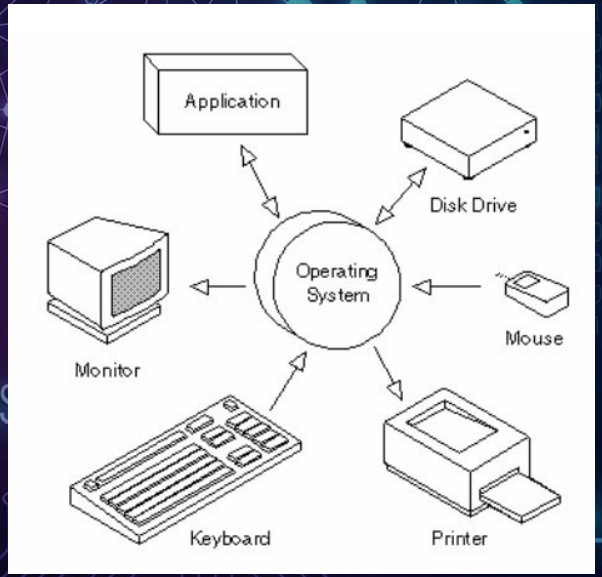
- **Basic software** : dedicated to the management of the computer itself, for example the Operating System

- **Application software** : dedicated to the creation of specific applications, such as internet browsing, word processing, or other.


COMPUTER SCIENCE
OPERATING SYSTEMS
STATISTICS
MACHINE LEARNING

# Operating System

- **It is low-level software that helps the user and high-level applications** interact with the hardware and the data that the programs store on the computer.
- An **OS performs basic operations** , such as recognizing input from the keyboard, sending output to the display, keeping track of directories and files on disk, and controlling peripherals such as printers.

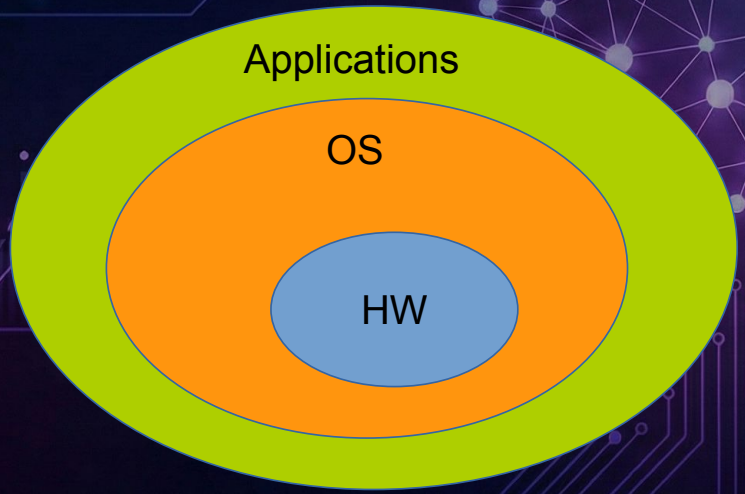


# Operating System

- **Running programs** : OSs provide an environment where the user can run application programs without having to worry about **memory allocation or CPU scheduling.**
- **I/O Operations** : Every application requires some input and produces some output. The **OS hides the details needed to handle these types of operations** from the underlying hardware.
- **Communication** : There are situations where two **applications need to communicate with each other, whether they are on the same computer (different processes) or on different computers** . The OS is responsible for managing this type of **inter-process communication.**

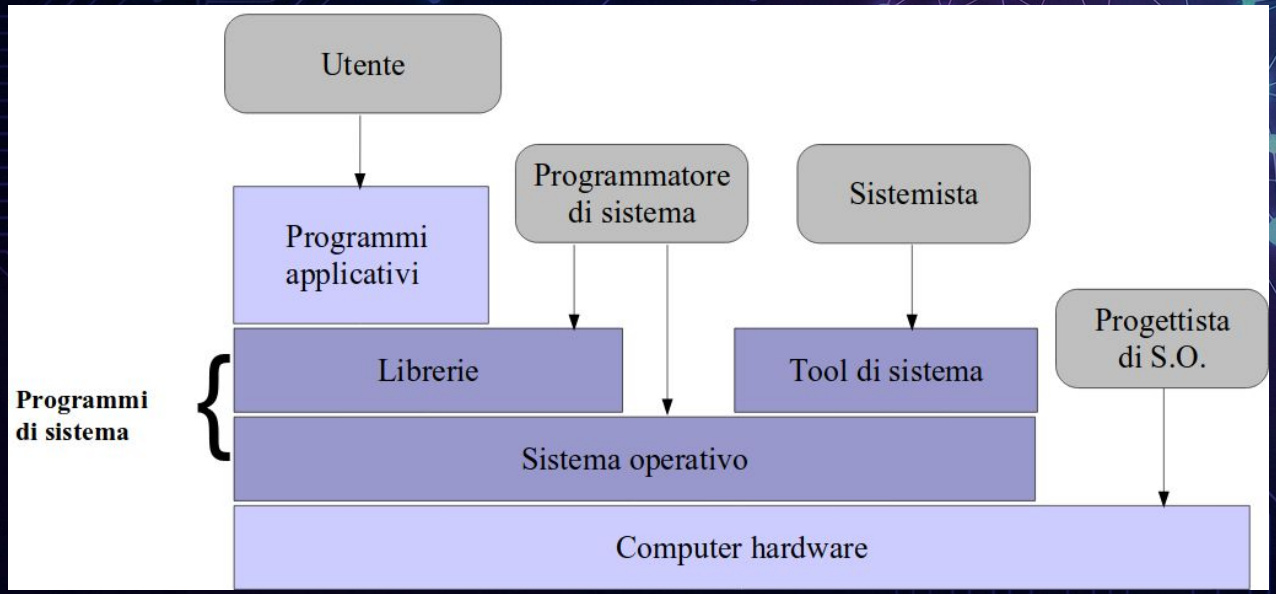
# Operating Systems

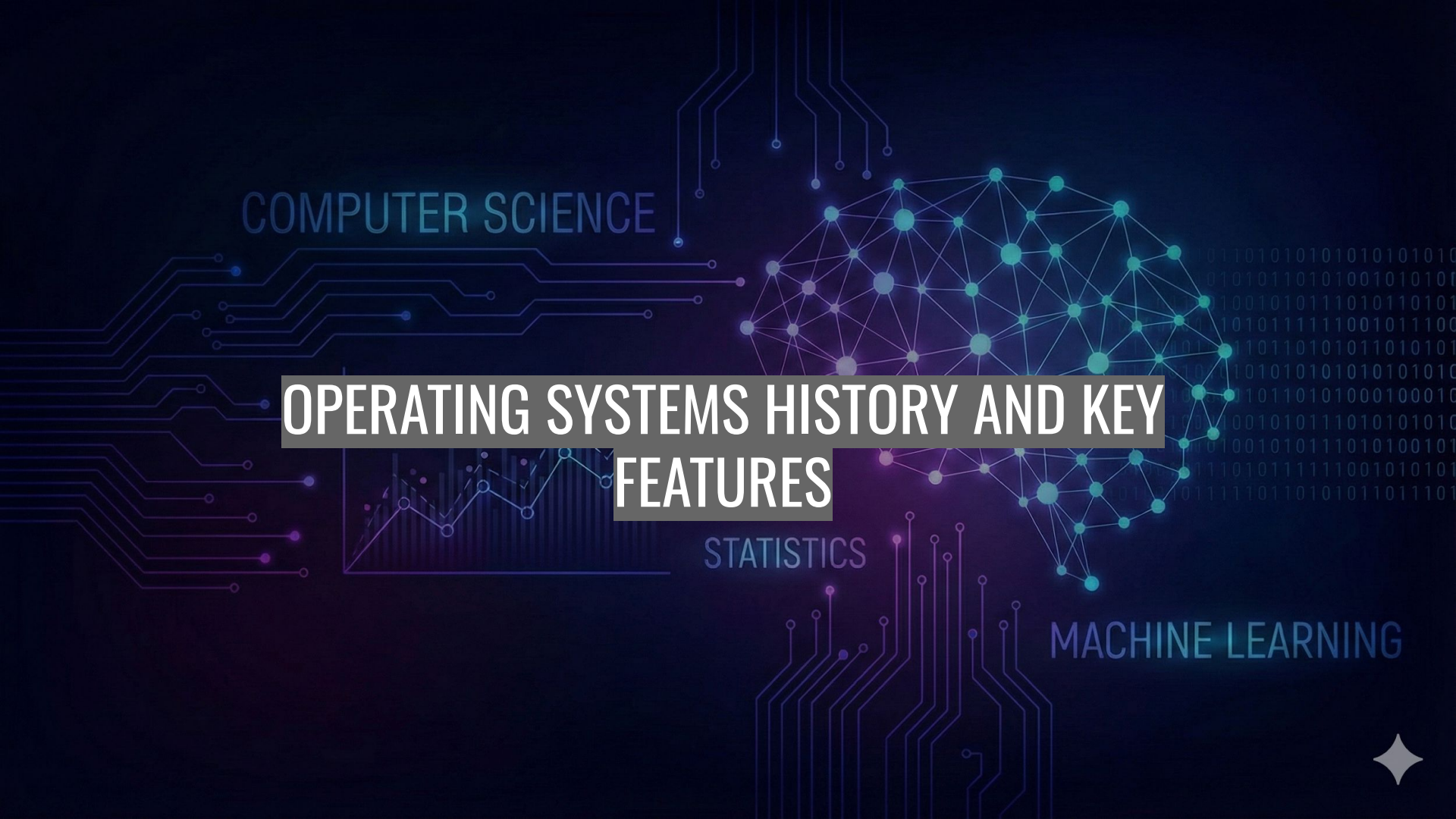
They make it easier to write application programs that do not have to worry about specific hardware characteristics.



# Operating Systems

Layered view of hardware and software components, e.g. provides the programmer with **an easy-to-use API** , hides hardware details






# OPERATING SYSTEMS HISTORY AND KEY FEATURES

# Operating System: History

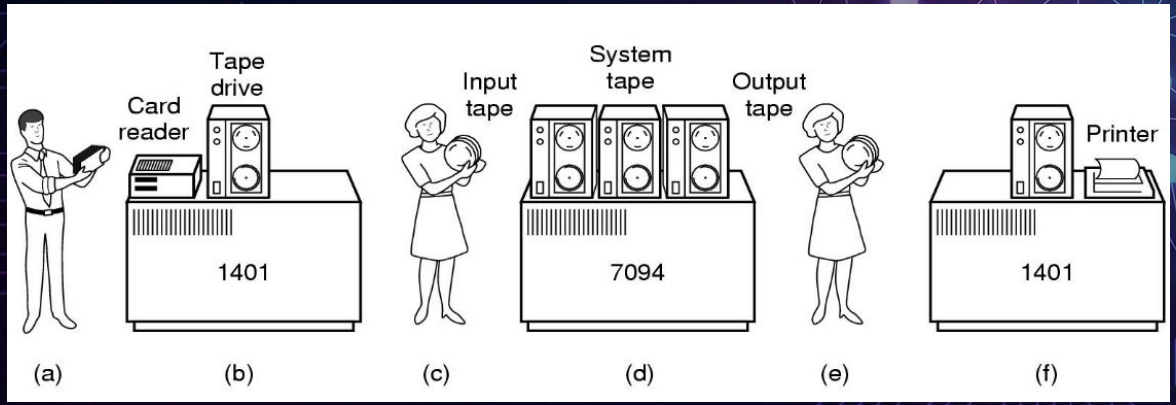
- **Babbage** (1792-1871) Tries to build an analytical and programmable **mechanical machine without an operating system**
  - **The first female programmer in history, Lady Ada Lovelace** (daughter of Byron)
- **Valve machines** (1944-1955), are **designed, built and programmed by a single group of people**
  - **Programmed in machine language** , used for numerical calculation only
  - **There is no OS** , there is no distinction between designer, programmer and user
  - **The single user writes the program** , loads it, loads the data, waits for the output and then moves on to execute the next program.
  - **No one imagines that computers will ever go beyond research laboratories.**

# Operating System: History

- **Transistors** (1955-1965) can build more reliable and cheaper machines
  - They are starting to be used for tasks other than basic **numerical calculation**
  - **The person who builds the machine is different from the person who programs it and therefore uses it (user == programmer)**
  - **first "high level" languages such as Assembly and FORTRAN** are introduced and **punched cards** are used.
  - **First examples of OS,** called **Resident Monitors :**
    - **Control** over the machine is given to the **monitor**
    - The **check** is **passed to the job** that is followed
    - Once the job is finished, **control returns to the monitor**

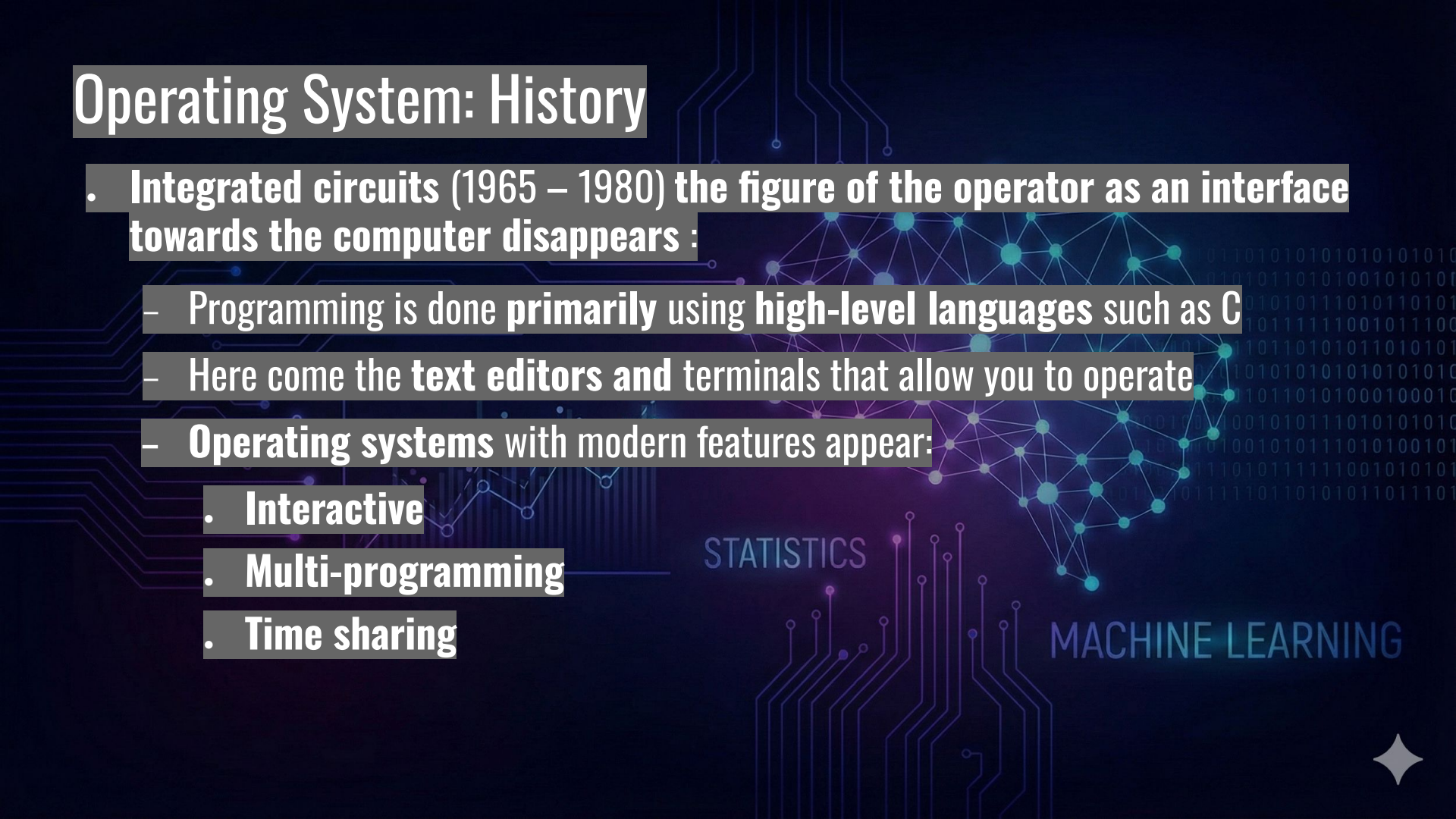
# Operating System: History

- To avoid downtime between the execution of one job and another, **tapes for storing jobs are introduced.**



# Operating System: History

- **Integrated circuits** (1965 – 1980) **the figure of the operator as an interface towards the computer disappears** :
  - Programming is done **primarily** using **high-level languages** such as C
  - Here come the **text editors and** terminals that allow you to operate
  - **Operating systems** with modern features appear:
    - **Interactive**
    - **Multi-programming**
    - **Time sharing**



# Operating System: Multiprogramming

- **Use the processor while other jobs are doing I/O**
- So there are **multiple jobs** in memory at the same time.
- The **scheduler** (OS component) **manages the CPU usage** , while one job does I/O the other uses the CPU, **thus eliminating dead times** (CPU idle)
- So **the I/O routines must be provided by the OS**
- The operating system must take care of **allocating memory for multiple jobs**
- Likewise the OS must be able to **allocate I/O resources between different processes.**

# Operating System: Time-sharing

- It is essentially the **extension of the concept of multiprogramming.**
- The CPU's execution time is divided into intervals ( **quanta** ). When a quantum expires, the current job's execution is interrupted, even if it doesn't need to perform I/O, and another process (job) is switched. Processes generally belong to different users.
- The **context-switch** is very fast and transparent to the user who has the **impression that many programs are being executed in parallel.**
- The presence of **multiple users** makes it necessary to include memory and file system protection mechanisms.
- A job is loaded from disk to memory, and vice versa ( **swapping** )

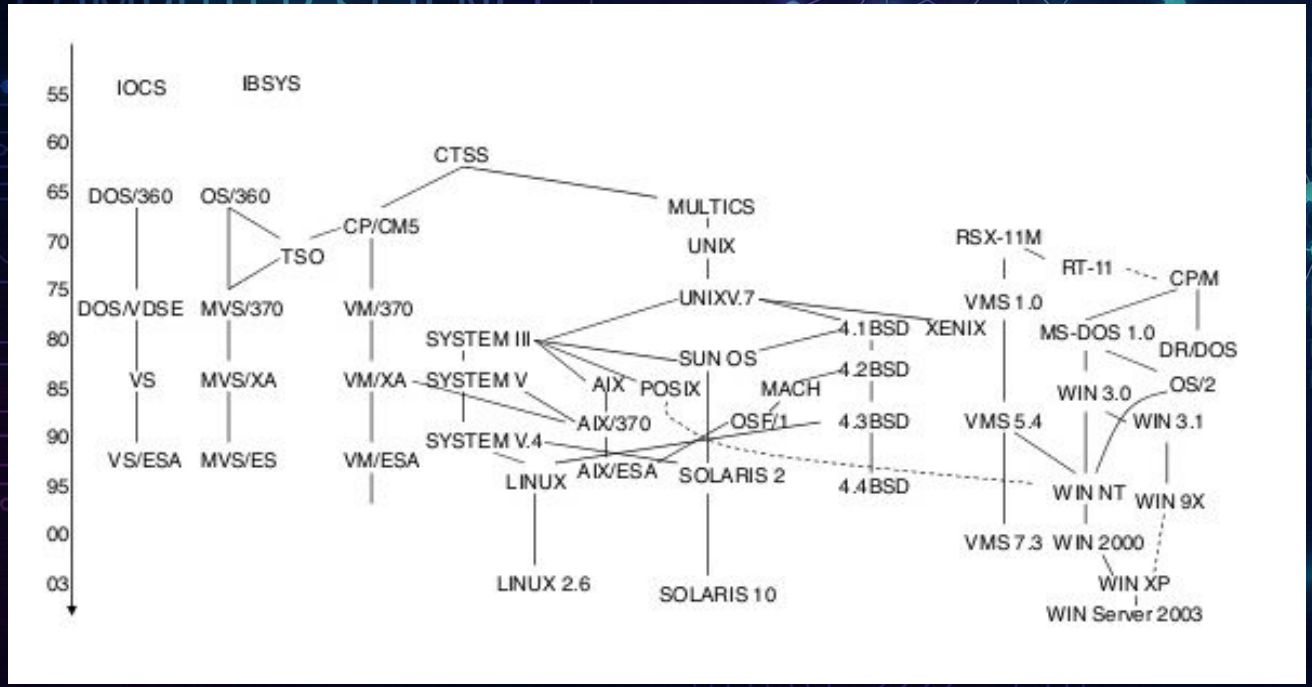
# Operating System: History

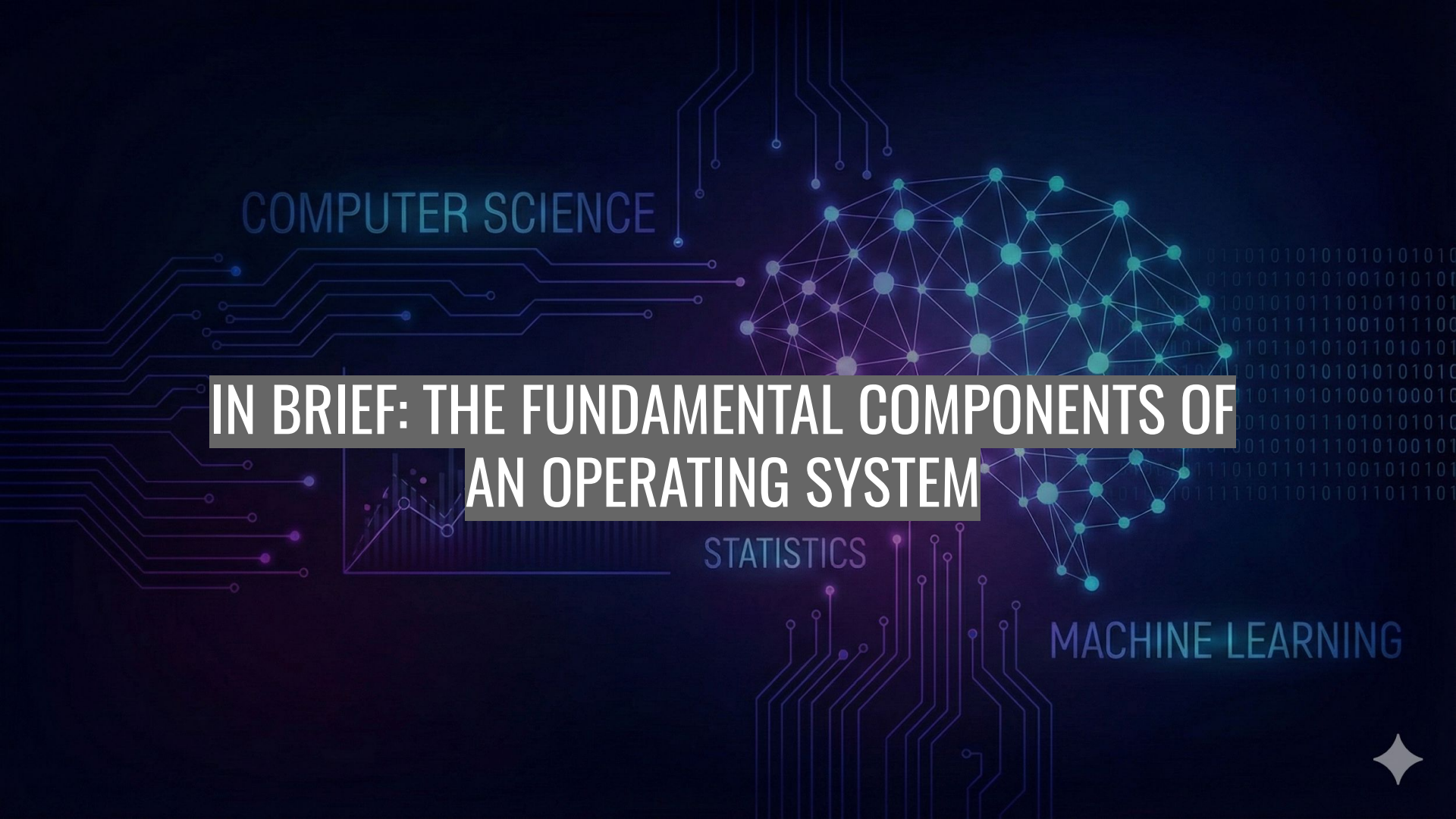
COMPUTER SCIENCE

- **CTSS (Compatible Time-Sharing System) 1965** : The concept of multi-programming and the concept of time-sharing are introduced
- **Multics 1965** : introduces the concept of process
- **Unix 1970** : derivative of Multics and CTSS initially developed at Bell Labs. Initially developed on two specific architectures, **then developed in C** (initially all in assembly).

MACHINE LEARNING

# Operating Systems: History






# IN BRIEF: THE FUNDAMENTAL COMPONENTS OF AN OPERATING SYSTEM

# Operating System: Process Management

- A process is basically **a running program, which therefore uses resources**
- The OS must be able to:
  - **Create and terminate the processes** themselves
  - **Manage communication between the processes** themselves
  - **Suspend and resume processes**

# Operating System: Main Memory Management

- Main memory is essentially an array of individually addressable bytes, which can be shared between the CPU and I/O devices.
- The OS must be able to:
  - **For example, keep track of which memory areas are used and by whom** (by which process)
  - **For example, decide which processes to allocate a given memory area** to when it is free.
  - Ultimately **allocate to free up memory space**

# Operating System: Secondary Memory Management

COMPUTER SCIENCE

- Computers we know are equipped with a **secondary, non-volatile memory** of large size (larger than the main memory which is volatile).
- The OS must be able to handle the following activities:
  - **Allocate the required space and free it** when no longer used
  - **Manage device access scheduling** (e.g. hard disks, CDs/DVDs, or USB sticks)

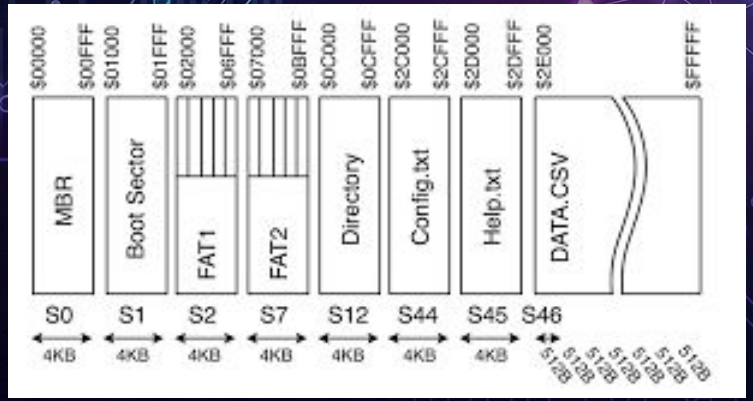
MACHINE LEARNING

# Operating System: Filesystem Management

- A **file** (archive) is the computer abstraction of the concept of an information container, regardless of the device on which it is stored.
- A **filesystem** is composed by many **files** (a **directory** contains references to other files). The OS must be able to handle the following tasks:
  - **Create and delete files and directories,** manipulate them
  - **Encrypt the filesystem** on secondary storage
  - Examples, ext3, ext4, NTFS, FAT32 ...

# Operating System: Filesystem Management

- The **filesystem** : must manage the allocation of disk space, for example maintaining an index of where the data relating to a given file is stored. Maintaining the characteristics of a file, such as access data, permissions, names, etc.



# Operating System: I/O Management

- The OS must manage I/O and therefore the interaction with the various hardware devices:
  - A common interface for managing various device **drivers**
  - Have different **drivers (kernel modules)** for different devices, therefore specific components for the various hardware components of the system
  - A system for **buffering and caching** information to and from various devices

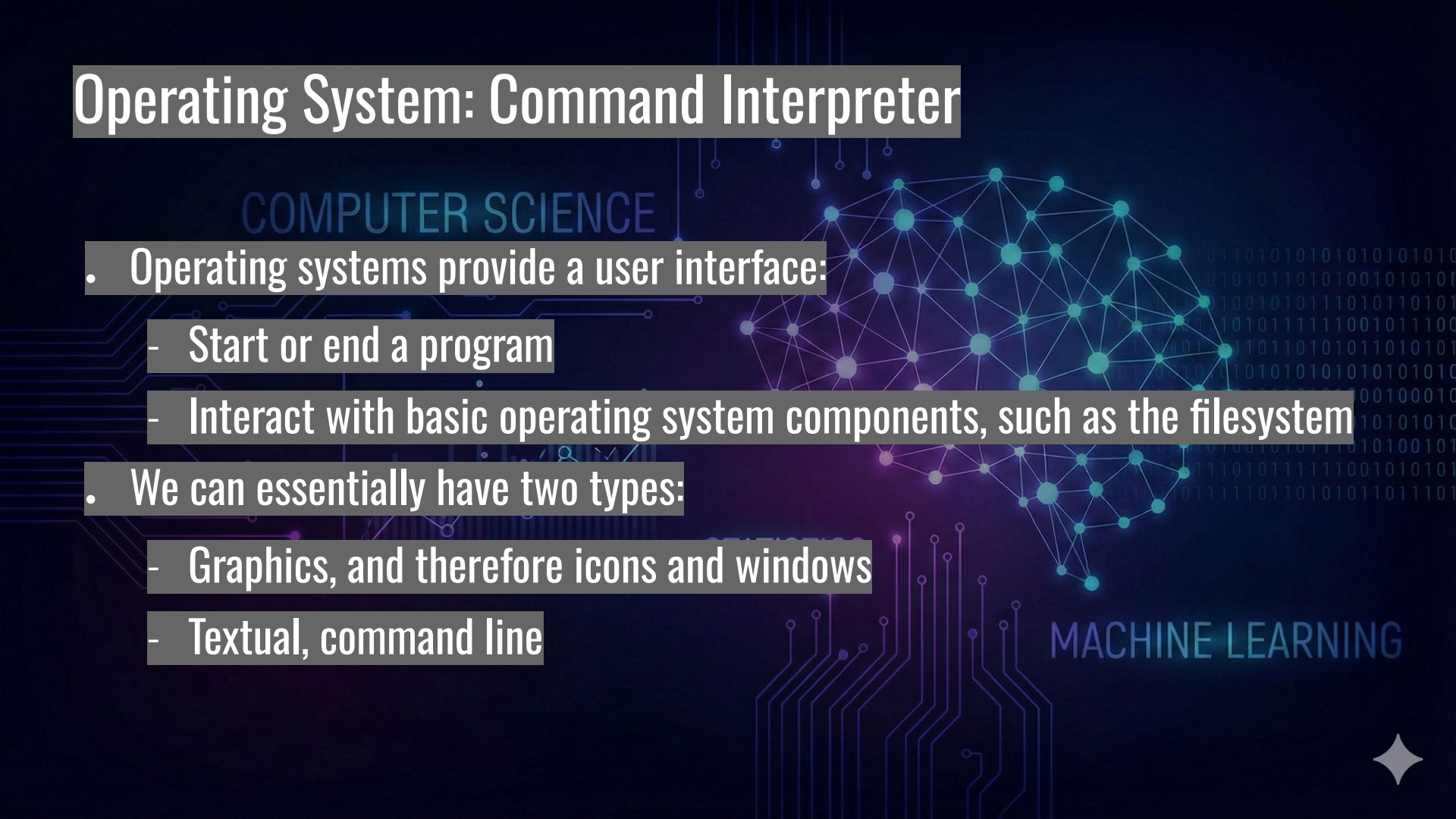
# Operating System: Protection

- The OS must implement a **software protection mechanism** . This means managing and controlling the access of various programs (processes) to shared system resources (e.g., memory).
  - Distinguish between authorized and unauthorized use
  - Provide basic mechanisms to implement protection

# Operating System: Networking

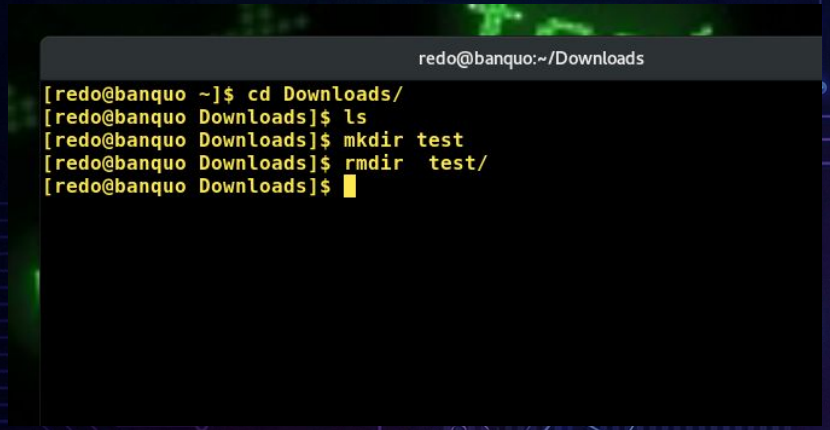
- Networking is now an essential component of an OS, that is, the ability to make two or more computers (processes) communicate with each other and share resources.
- An OS provides basic communication protocols such as **TCP/IP, UDP**
- **As well as high-level communication services** such as shared file systems (SMB, NFS) and many others.

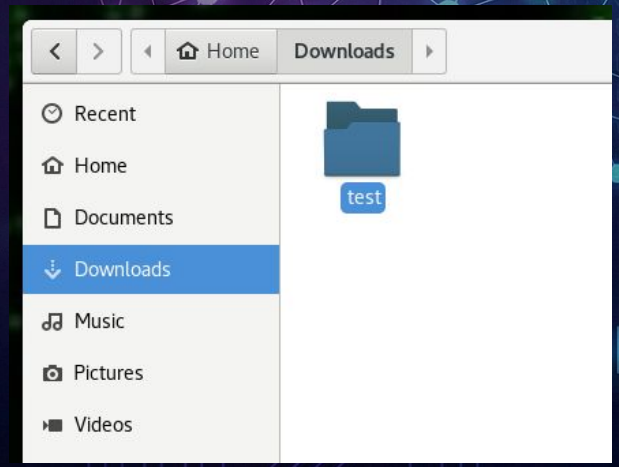
# Operating System: Command Interpreter



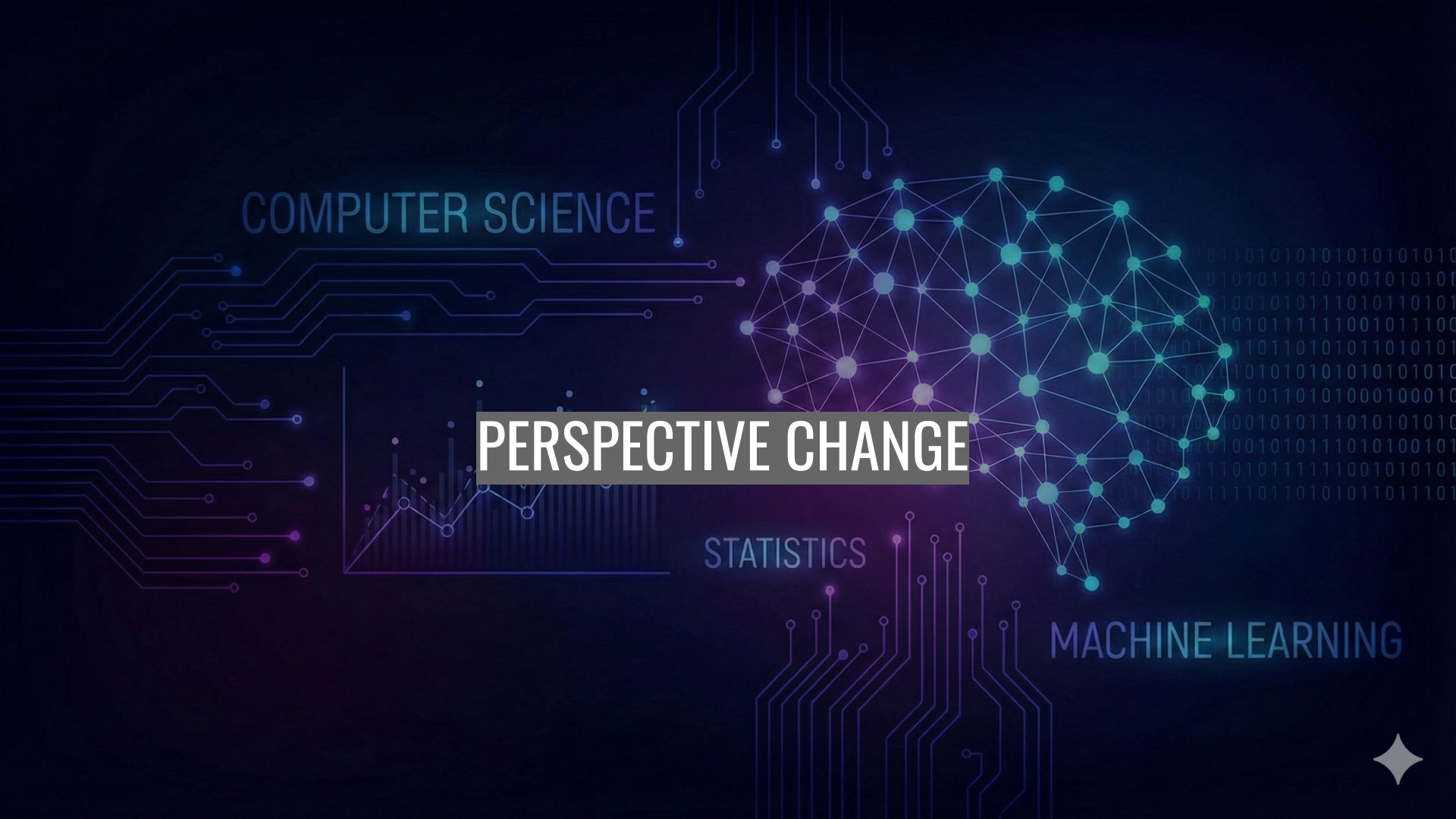

- Operating systems provide a user interface:
  - Start or end a program
  - Interact with basic operating system components, such as the filesystem
- We can essentially have two types:
  - Graphics, and therefore icons and windows
  - Textual, command line

# Operating System: Command Interpreter



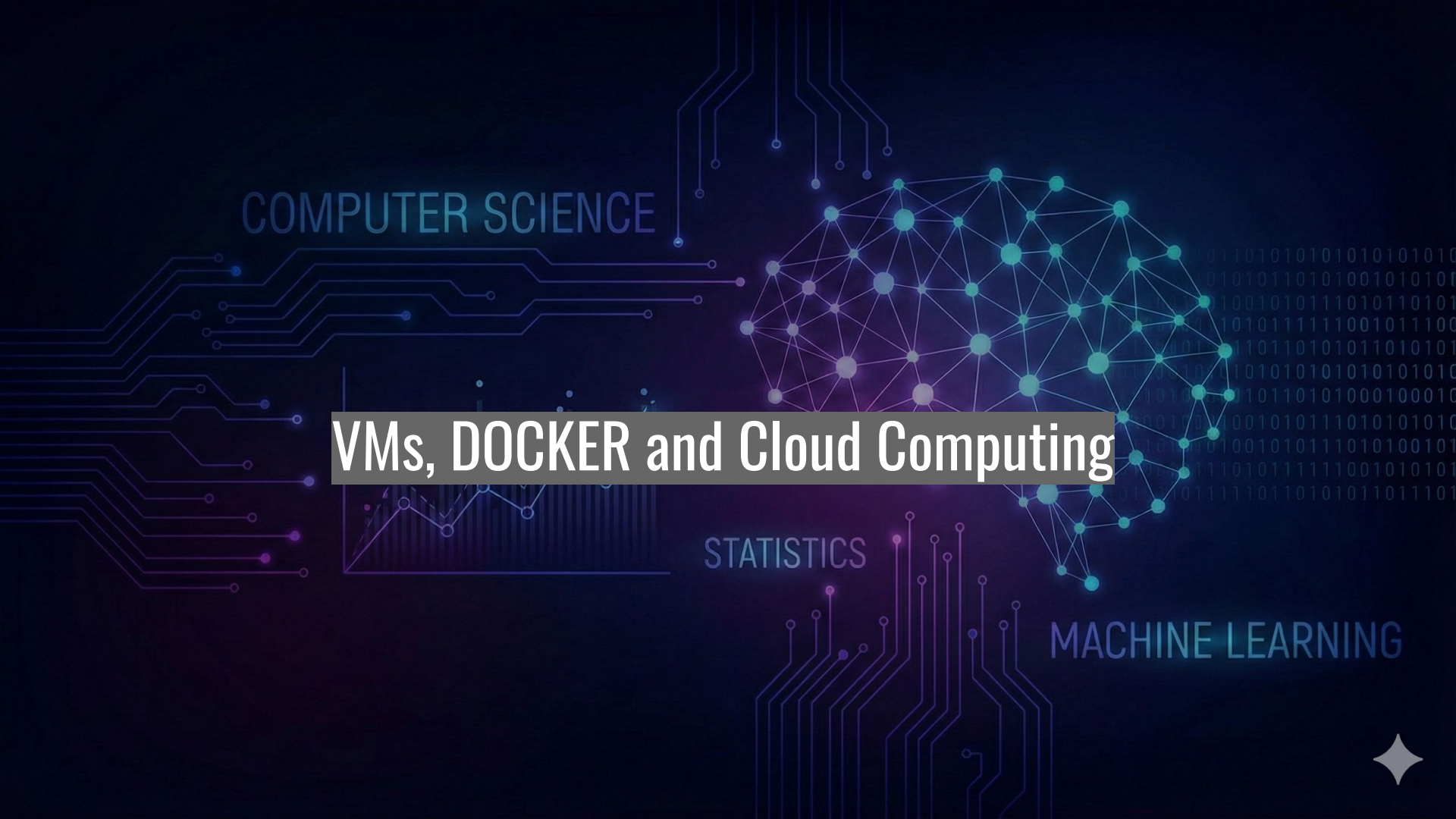

INE LEARNING


COMPUTER SCIENCE
PERSPECTIVE CHANGE
STATISTICS
MACHINE LEARNING

# Operating System: Programmer's Perspective

- The System Call (system calls) for example:

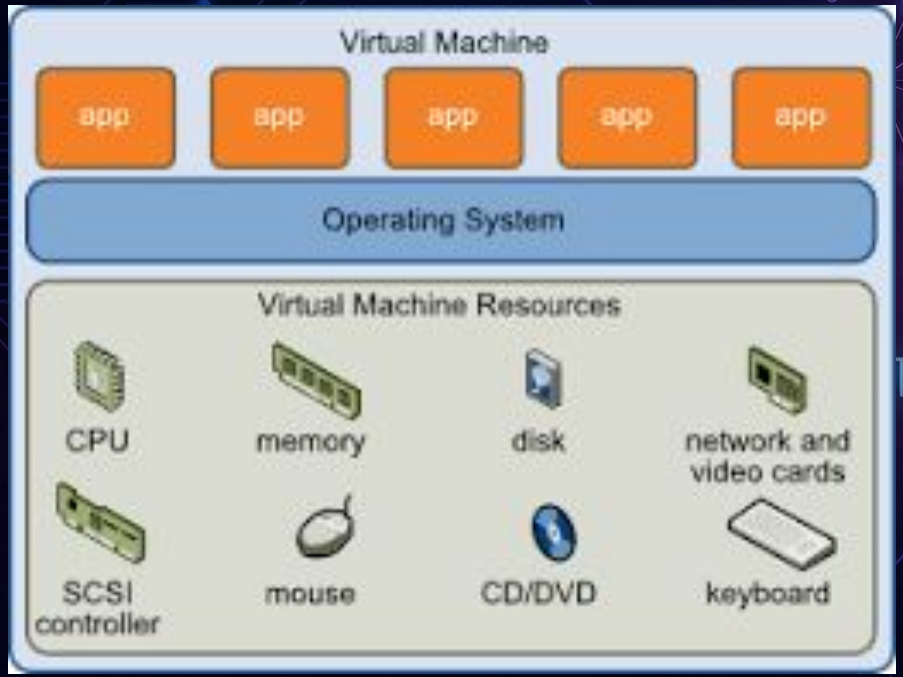
| UNIX | WIN32 | UNIX | WIN32 |
|---|---|---|---|
| fork | CreateProcess | mkdir | CreateDirectory |
| waitpid | WaitForSingleObject | rmdir | RemoveDirectory |
| execve | - | link | - |
| exit | ExitProcess | unlink | DeleteFile |
| open | CreateFile | mount | - |
| close | CloseHandle | umount | - |
| read | ReadFile | chdir | SetCurrentDirectory |
| write | WriteFile | chmod | - |
| lseek | SetFilePointer | kill | - |
| stat | GetFileAttributesEx | time | GetLocalTime |




# VMs, DOCKER and Cloud Computing

# Cloud computing

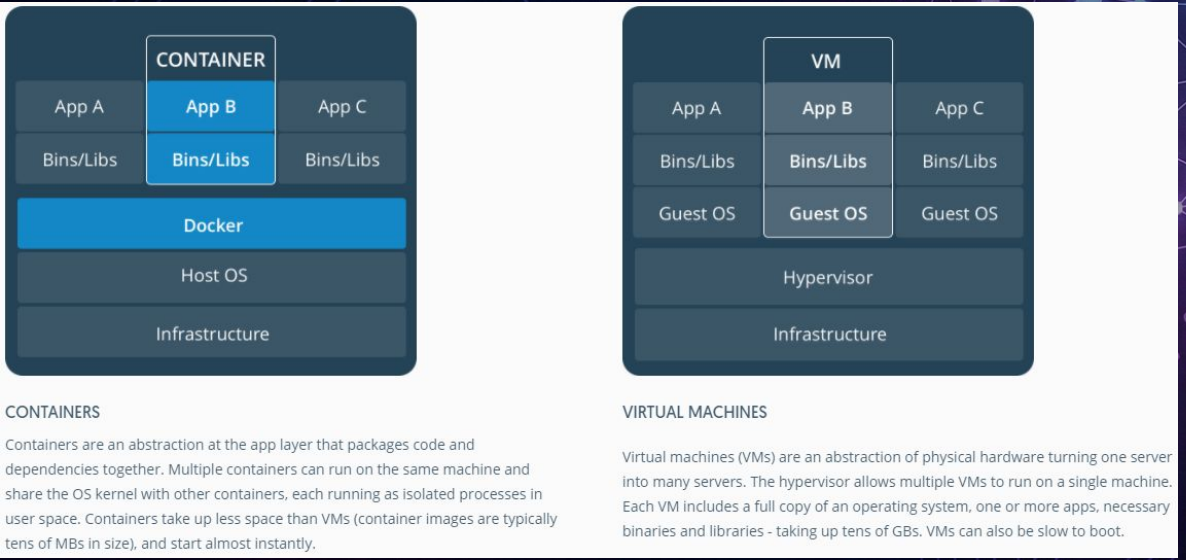
- Virtual Machine; turning one physical server into many virtual servers.




A Virtual Machine (VM) is a software emulation of a physical computer that runs its own completely isolated operating system and applications, functioning exactly like a separate physical machine while sharing the underlying hardware of a host server.

# Cloud computing

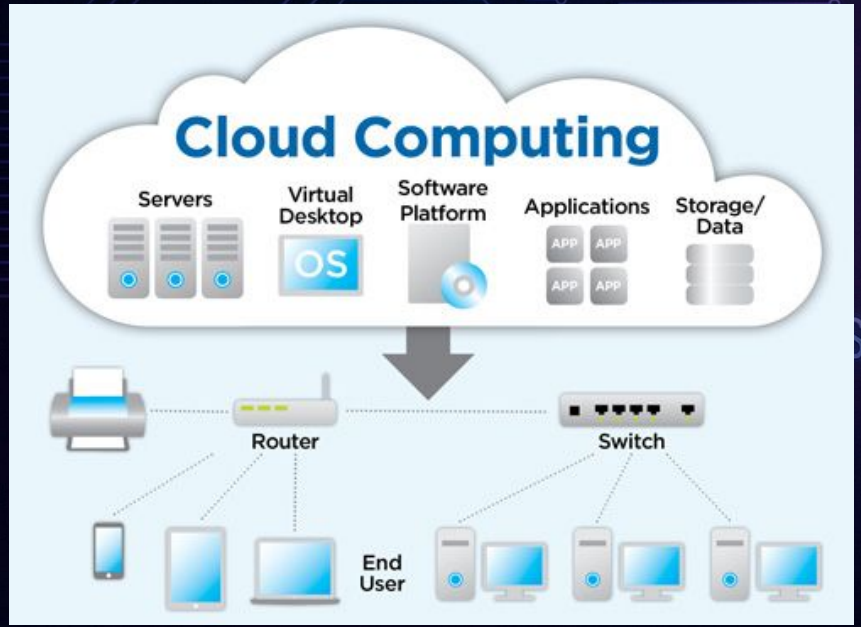
- Docker vs VM




While a Virtual Machine (VM) emulates an entire computer system with its own heavy operating system kernel, a Docker container packages only the application and its dependencies, sharing the host system's kernel to run much faster and more efficiently.

# Cloud computing

- A virtual infrastructure




Cloud computing is the on-demand delivery of computing services—such as servers, storage, databases, and software—over the Internet, allowing users to access technology resources as a pay-as-you-go utility rather than owning and maintaining physical data centers.

# Contents

- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption

# Binary representation of numbers (short digression)

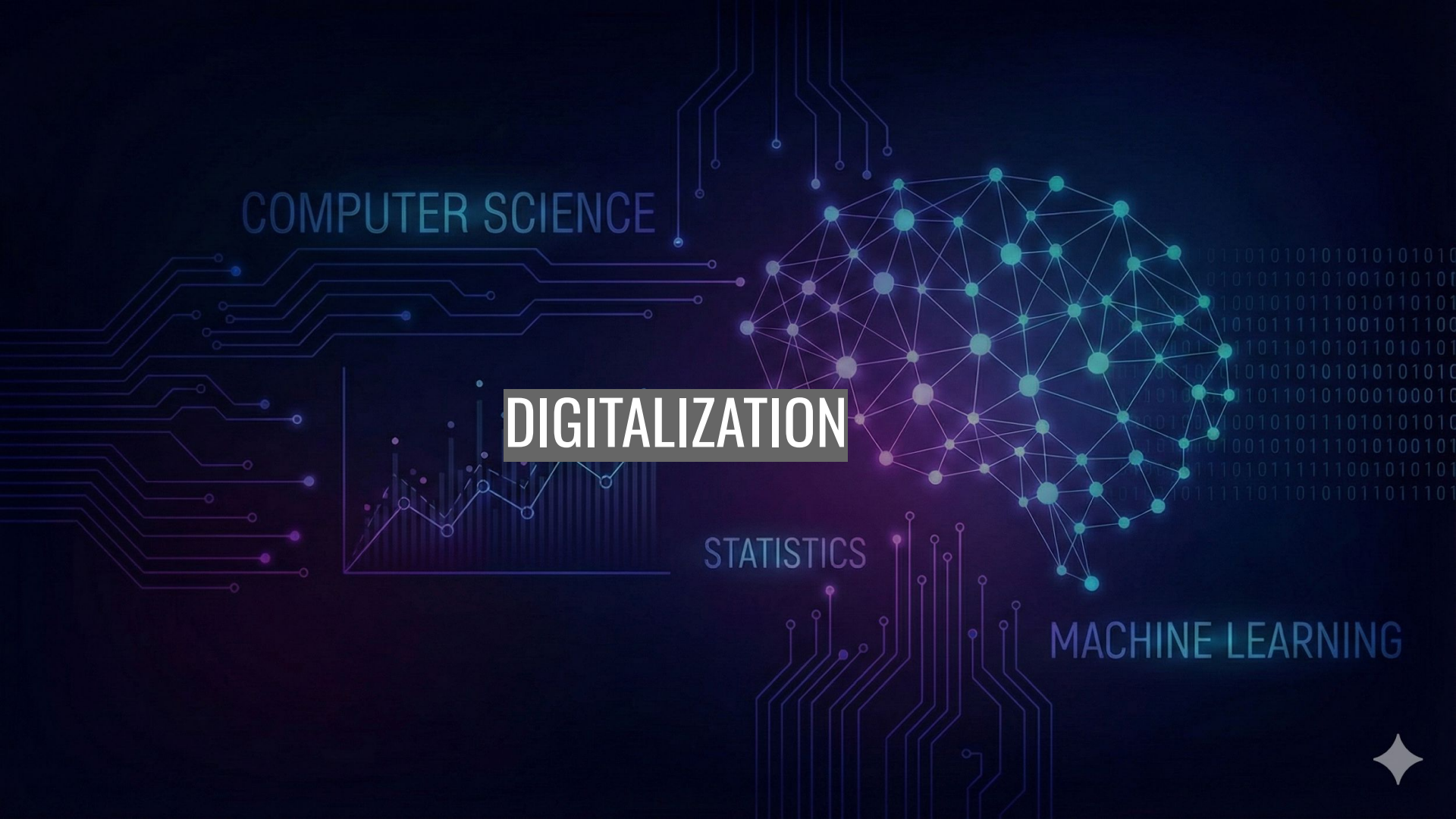
- In a **positional numbering system,** given the **base** , this directly defines the number of **symbols (digits)** used to write the number.
  - For example in the decimal system we use 10 symbols (0,1,2,3,4,5,6,7,8,9)
- Modern numbering systems are positional, **so the number is written specifying the order of the digits, and each digit takes on a value depending on its position.**
  - For example $423 = 4 * 10^{2} + 2 * 10^{1} + 3 * 10^{0}$
    - If you want 4 hundreds, 2 tens and 3 ones

# Binary representation of numbers (short digression)

- In general, given a **base b, I will have b symbols** (digits) and therefore **an integer N** will be written as:
  - Value $N = c_n * b^n + c_{n-1} * b^{n-1} + \ldots + c_0 * b^0$
- Similarly if I have a number $N = 0.c_1 c_2 \ldots c_n$
  - N value $= c_1 * b^{-1} + c_2 * b^{-2} + \ldots + c_n * b^{-n}$
- Let us now consider the simplest case, that of the representation of **unsigned integers (the Naturals).**
- If I use a numbering system with **base b with n digits** I will be able to represent a **maximum of $b^n$ different numbers** , therefore all the numbers from 0 to $ab^n - 1$
- For example, in **base 10** it is clear that using **two digits I can represent all the numbers from 0 to 99,** therefore **$10^2 = 100$** distinct numbers.

# The binary base

- Using a base of 2, therefore only two symbols 0 and 1, always remaining within the scope of the representation of positive integers using n digits I will be able to represent at most all the numbers between **0 and $2^n$ -1**
  - For example, using two digits I can represent 4 distinct numbers:
  - $00 = 0 * 2^1 + 0 * 2^0 = 0$
  - $01 = 0 * 2^1 + 1 * 2^0 = 1$
  - $10 = 1 * 2^1 + 0 * 2^0 = 2$
  - $11 = 1 * 2^1 + 1 * 2^0 = 3$


COMPUTER SCIENCE
DIGITALIZATION
STATISTICS
MACHINE LEARNING

# The binary base

- A byte (a morsel) modernly represents the sequence of 8 bits and has historically become the basic element of addressability and therefore the basic unit of measurement of information.
- **8 bits means that with 1 byte I can represent $2^8 = 256$ different values** . So in the case of unsigned integers, the numbers from 0 to 255. **If I use a bit to indicate the sign, for example 0 is positive and 1 is negative,** I can represent the integers from -128 to 127 (from 10000000 to 01111111).
- Or with 8 bits I can represent 256 different characters.

# ASCII

- Extended ASCII
- use 8-bit
- Original ASCII
- US-ASCII 7-bit

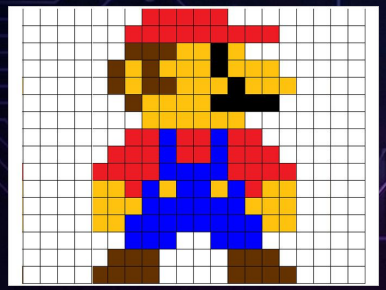
| 000 | NUL | 033 | ! | 066 | B | 099 | c | 132 | ä | 165 | Ñ | 198 | ã | 231 | þ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 001 | Start Of Header (SOH) | 034 | " | 067 | C | 100 | d | 133 | à | 166 | ª | 199 | Ã | 232 | Þ |
| 002 | Start Of Text (STX) | 035 | # | 068 | D | 101 | e | 134 | å | 167 | º | 200 | ╚ | 233 | Ú |
| 003 | End Of Text (ETX) | 036 | $ | 069 | E | 102 | f | 135 | ç | 168 | ¿ | 201 | ╔ | 234 | Û |
| 004 | End Of Transmission (EOT) | 037 | % | 070 | F | 103 | g | 136 | ê | 169 | ® | 202 | ╩ | 235 | Ù |
| 005 | Enquiry | 038 | & | 071 | G | 104 | h | 137 | ë | 170 | ¬ | 203 | ╦ | 236 | ý |
| 006 | Acknowledge (ACK) | 039 | ' | 072 | H | 105 | i | 138 | è | 171 | ½ | 204 | ╠ | 237 | Ý |
| 007 | Bell | 040 | ( | 073 | I | 106 | j | 139 | ï | 172 | ¼ | 205 | ═ | 238 | ¯ |
| 008 | Backspace (BS) | 041 | ) | 074 | J | 107 | k | 140 | î | 173 | ¡ | 206 | ╬ | 239 | ´ |
| 009 | Horizontal Tab | 042 | * | 075 | K | 108 | l | 141 | ì | 174 | « | 207 | ¤ | 240 | - |
| 010 | Line Feed **(LF)** | 043 | + | 076 | L | 109 | m | 142 | Ä | 175 | » | 208 | ð | 241 | ± |
| 011 | Vertical Tab | 044 | , | 077 | M | 110 | n | 143 | Å | 176 | ░ | 209 | Ð | 242 | ‗ |
| 012 | Form Feed **(FF)** | 045 | - | 078 | N | 111 | o | 144 | É | 177 | ▒ | 210 | Ê | 243 | ¾ |
| 013 | Carriage Return **(CR)** | 046 | . | 079 | O | 112 | p | 145 | æ | 178 | ▓ | 211 | Ë | 244 | ¶ |
| 014 | Shift Out | 047 | / | 080 | P | 113 | q | 146 | Æ | 179 | │ | 212 | È | 245 | § |
| 015 | Shift In | 048 | 0 | 081 | Q | 114 | r | 147 | ô | 180 | ┤ | 213 | ı | 246 | ÷ |
| 016 | Dataline Escape (DLE) | 049 | 1 | 082 | R | 115 | s | 148 | ö | 181 | Á | 214 | Í | 247 | ¸ |
| 017 | DC 1 (XON) | 050 | 2 | 083 | S | 116 | t | 149 | ò | 182 | Â | 215 | Î | 248 | ° |
| 018 | DC 2 | 051 | 3 | 084 | T | 117 | u | 150 | û | 183 | À | 216 | Ï | 249 | ¨ |
| 019 | DC 3 (XOFF) | 052 | 4 | 085 | U | 118 | v | 151 | ù | 184 | © | 217 | ┘ | 250 | · |
| 020 | DC 4 | 053 | 5 | 086 | V | 119 | w | 152 | ÿ | 185 | ╣ | 218 | ┌ | 251 | ¹ |
| 021 | Negative Acknowledge (NAK) | 054 | 6 | 087 | W | 120 | x | 153 | Ö | 186 | ║ | 219 | █ | 252 | ³ |
| 022 | Synchronous Idle | 055 | 7 | 088 | X | 121 | y | 154 | Ü | 187 | ╗ | 220 | ▄ | 253 | ² |
| 023 | End Of Transmission Block | 056 | 8 | 089 | Y | 122 | z | 155 | ø | 188 | ╝ | 221 | ¦ | 254 | ■ |
| 024 | Cancel | 057 | 9 | 090 | Z | 123 | { | 156 | £ | 189 | ¢ | 222 | Ì | 255 |  |
| 025 | End Of Medium | 058 | : | 091 | [ | 124 | \| | 157 | Ø | 190 | ¥ | 223 | ▀ |  |  |
| 026 | Substitude | 059 | ; | 092 | \ | 125 | } | 158 | × | 191 | ┐ | 224 | Ó |  |  |
| 027 | Escape **(ESC)** | 060 | < | 093 | ] | 126 | ~ | 159 | ƒ | 192 | └ | 225 | ß |  |  |
| 028 | File Seperator | 061 | = | 094 | ^ | 127 (DEL) | ⌂ | 160 | á | 193 | ┴ | 226 | Ô |  |  |
| 029 | Group Seperator | 062 | > | 095 | _ | 128 | Ç | 161 | í | 194 | ┬ | 227 | Ò |  |  |
| 030 | Record Seperator | 063 | ? | 096 | ` | 129 | ü | 162 | ó | 195 | ├ | 228 | õ |  |  |
| 031 | Unit Seperator | 064 | @ | 097 | a | 130 | é | 163 | ú | 196 | ─ | 229 | Õ |  |  |
| 032 | *SPACE* (SP) | 065 | A | 098 | b | 131 | â | 164 | ñ | 197 | ┼ | 230 | µ |  |  |

# Information coding

- An encoding is a convention, as seen before, therefore the bit sequence **01001100 can represent the character L (uppercase L) or if instead we mean an unsigned integer value it represents the decimal number 76 .**
- For example **, every image is made up of pixels. If I used only 1 bit for each pixel, I could only have black and white images (1 black pixel, 0 white pixels).**

# Information coding

If I use more bits to represent each pixel, I can instead obtain ranges of grays or colors. And from there, sounds, videos...



The pixel represents the smallest autonomous element of the image. Each pixel is therefore characterised by the own position

The total number of pixels in a digital image is called its **resolution** . For example, if I have a 10 x 10 grid, the image will be made up of 100 pixels.

**dpi** = **number of dots per inch,** for example in a typical monitor I will have 72 pixels per inch

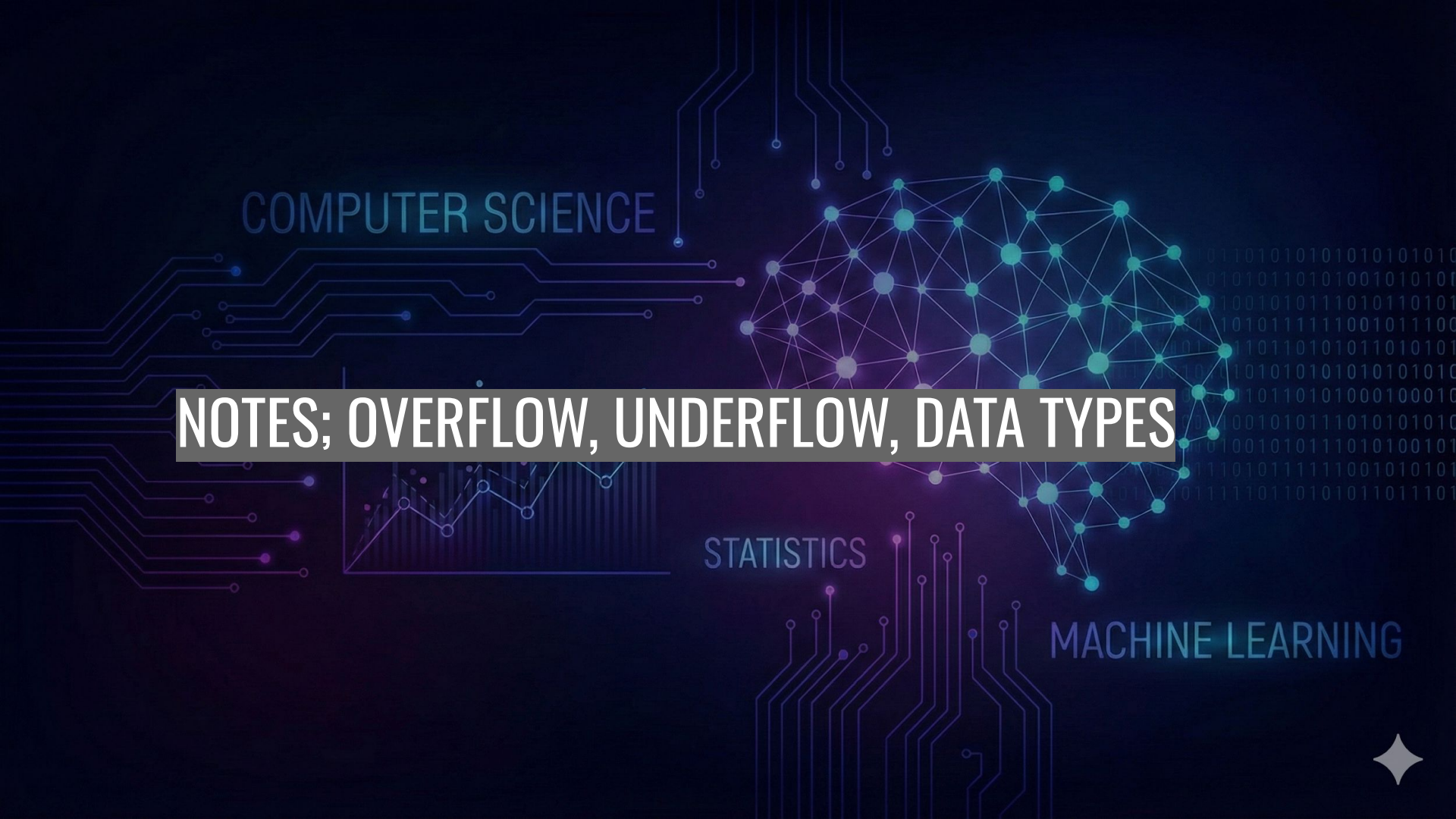
**Depth:** in the case of a grayscale image, 8 bits can be used for each pixel, thus having **$2^8$ = 256 shades of gray available.**

# Information coding

- Color images

In the case of color images, **each pixel** is characterized by **three color scales** for the three primary colors, **RGB (Red, Green, Blue).**

For example, an **8-bit image** will have 256 possible shades of red, 256 possible shades of green, and 256 possible shades of blue for each color. Therefore, a total of **1,677,7216** possible shades of color. In the case of 12-bit images (high quality professional ones), we will instead have $\mathbf{2^{12}}$ = **4,096 shades of red for each color** , for a total of **4096 x 4096 x 4096 = 68,719,476,736$** possible colors for each pixel.




# NOTES; OVERFLOW, UNDERFLOW, DATA TYPES

# The computers' memory is running out

- **OVERFLOW** : There are not enough bits to represent the result. **UNDERFLOW:** Number too small 3/2, I cannot represent 1.5, I can only represent 1 or 2.
- For example, if I use 8 bits to represent positive integers, I can only represent numbers from 0 to 255, so what happens if I try to add 1 to 255?

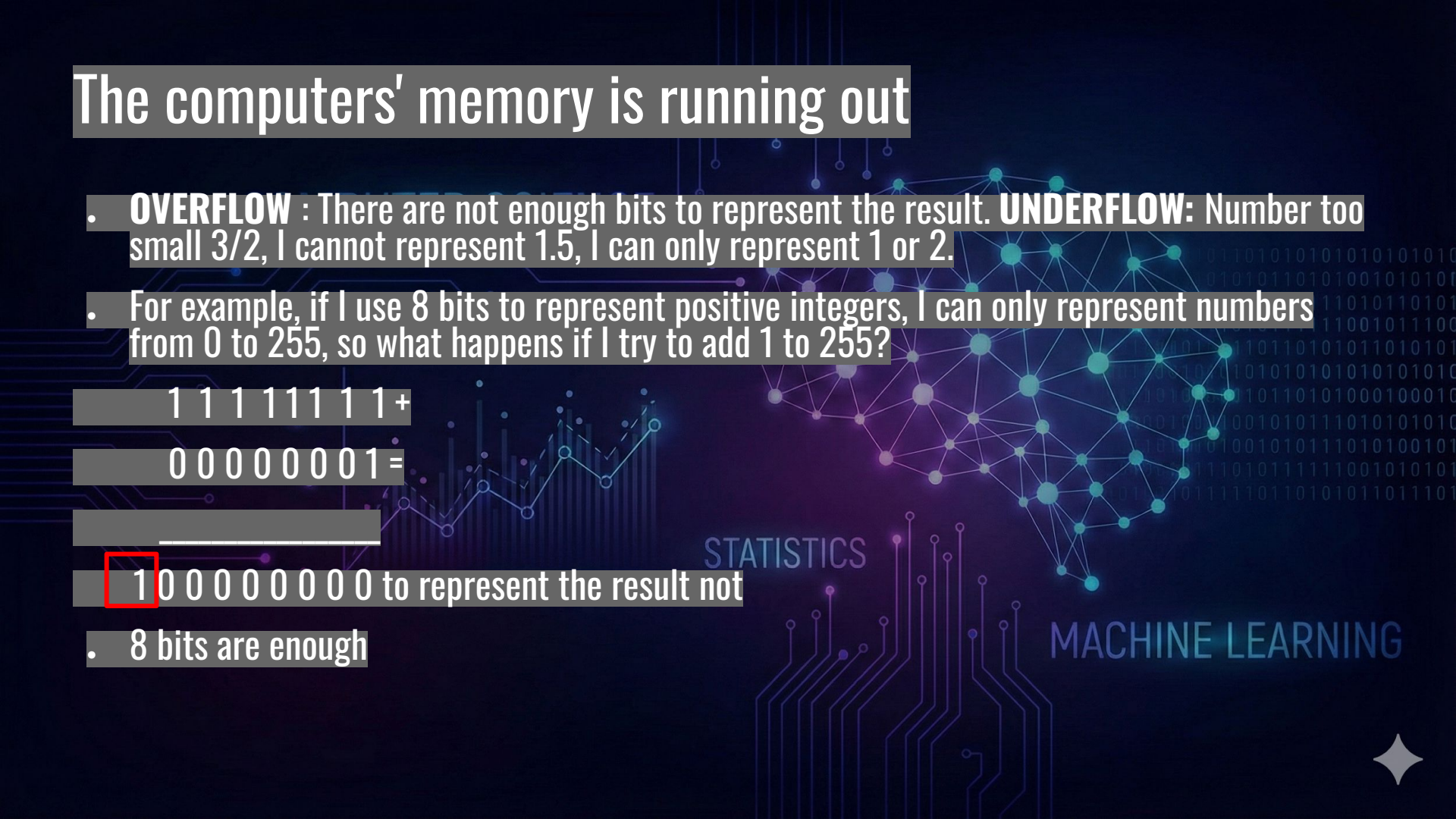
1 1 1 1 1 1 1 1 +

0 0 0 0 0 0 0 1 =

_____

1 0 0 0 0 0 0 0 0 to represent the result not

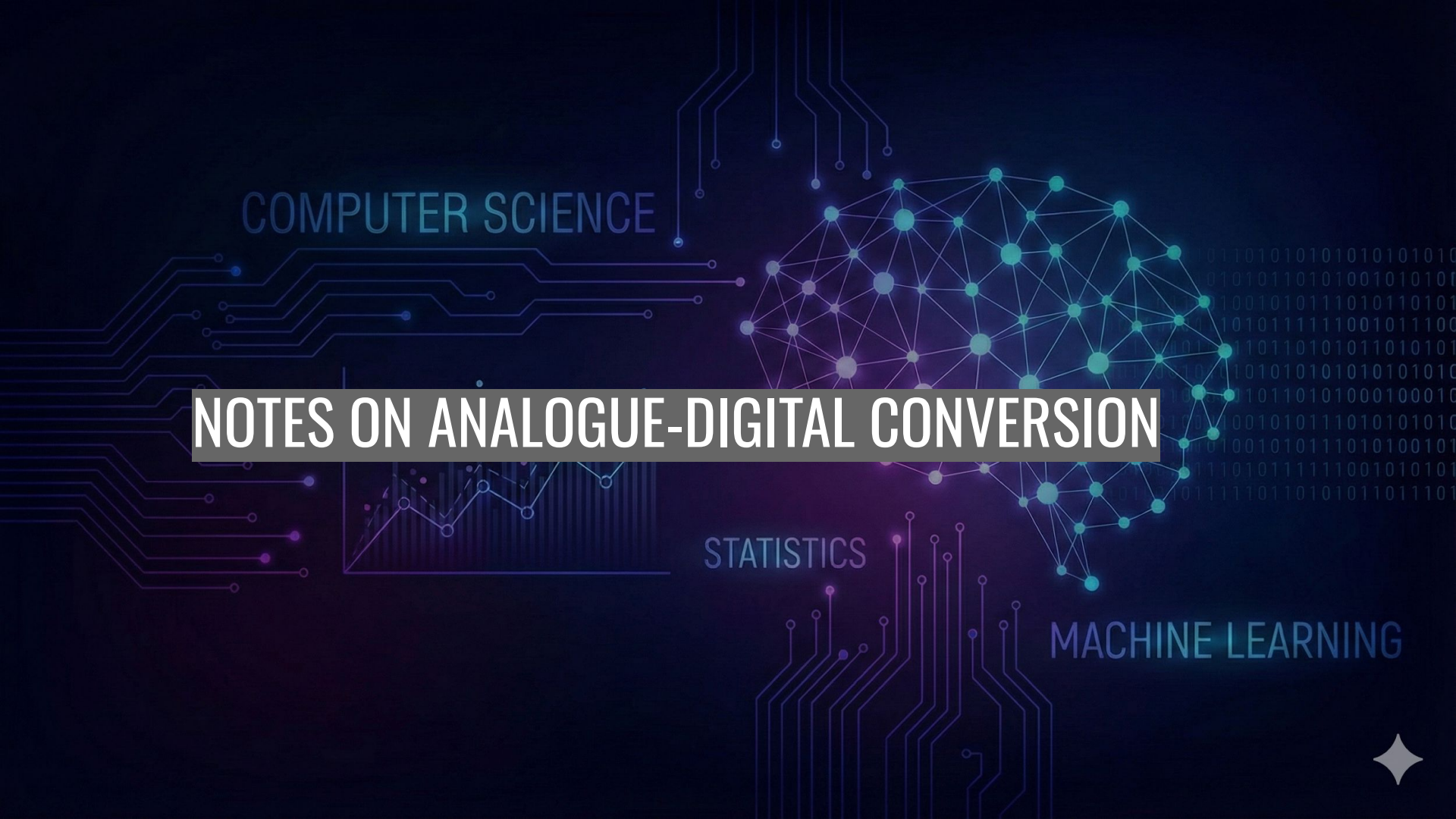
- 8 bits are enough



# Data types

- See the difference between strongly typed and non-strongly typed languages, dynamic and static typing
- Programming languages have native data types, such as integers, floating-points, booleans, and characters. Different types have different sizes and therefore different ranges (exact algebra and non-exact algebra...)

| label | size (bytes) | smallest value | largest value |
|---|---|---|---|
| byte | 1 | -128 | 127 |
| short | 2 | -32768 | 32767 |
| int | 4 | -2147483648 | 2147483647 |
| long | 8 | -9223372036854775808 | 9223372036854775807 |
| char | 2 | 0 | 65535 |

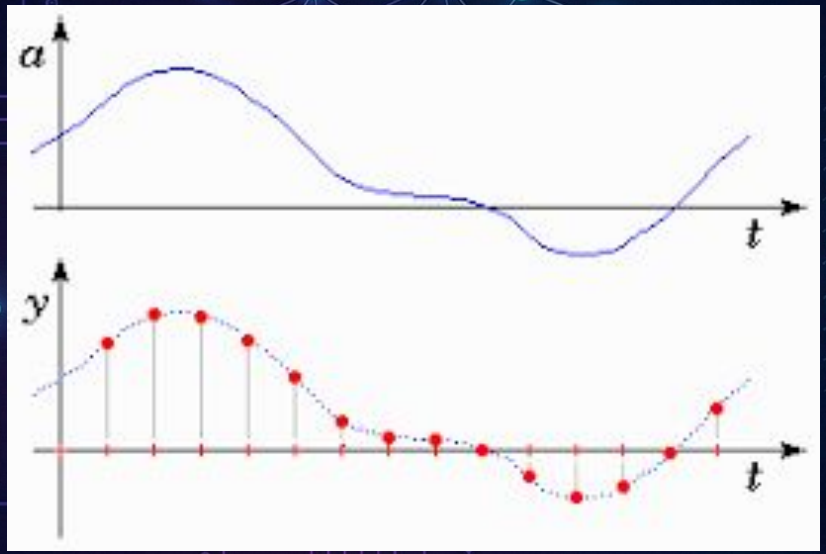



# NOTES ON ANALOGUE-DIGITAL CONVERSION

# Analog signal

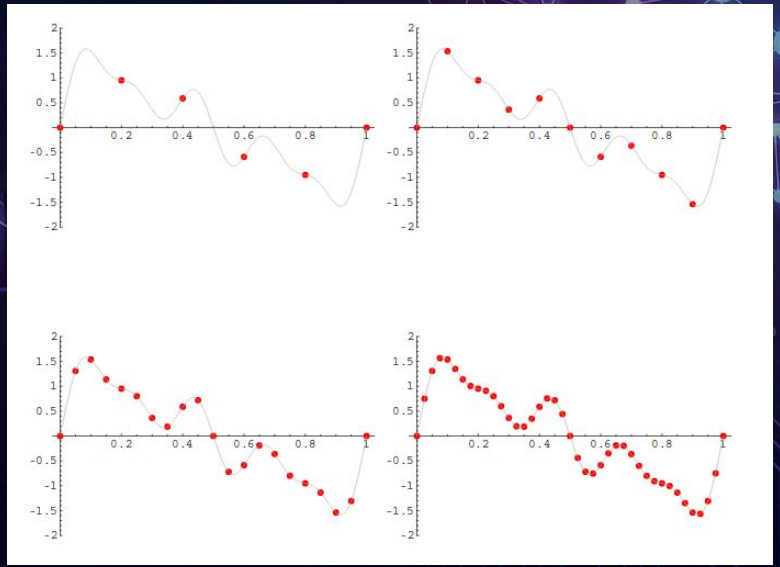
## Analog to Digital Conversion

**sampling - time discretization** (Nyquist-Shannon sampling theorem)
**quantization - discretization of the amplitude**
coding - using binary “words” to express the value of the signal



# Analog signal

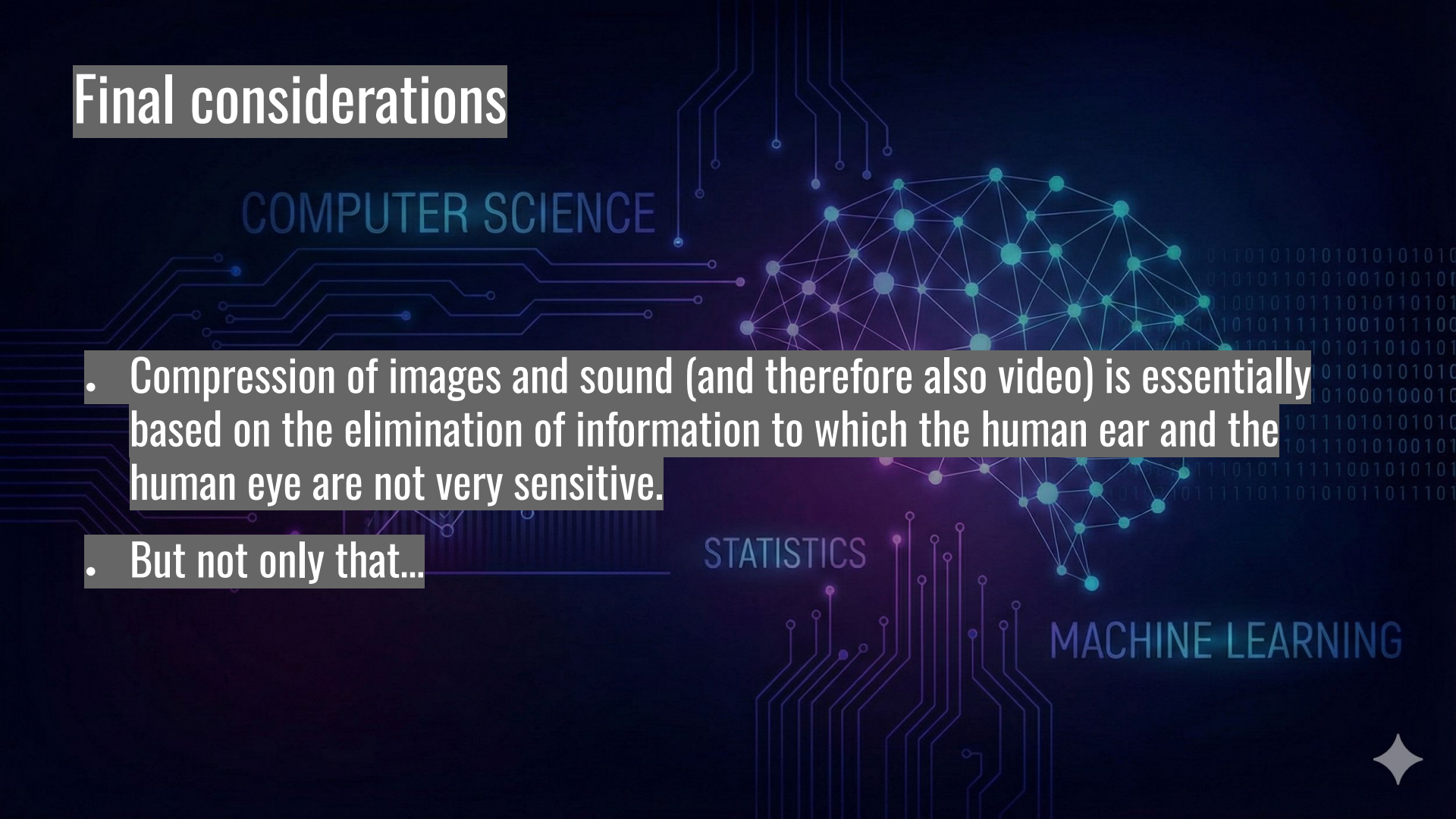
Obviously, fidelity improves as the number of samples per unit of time (sampling frequency) and the number of quantization levels increase.



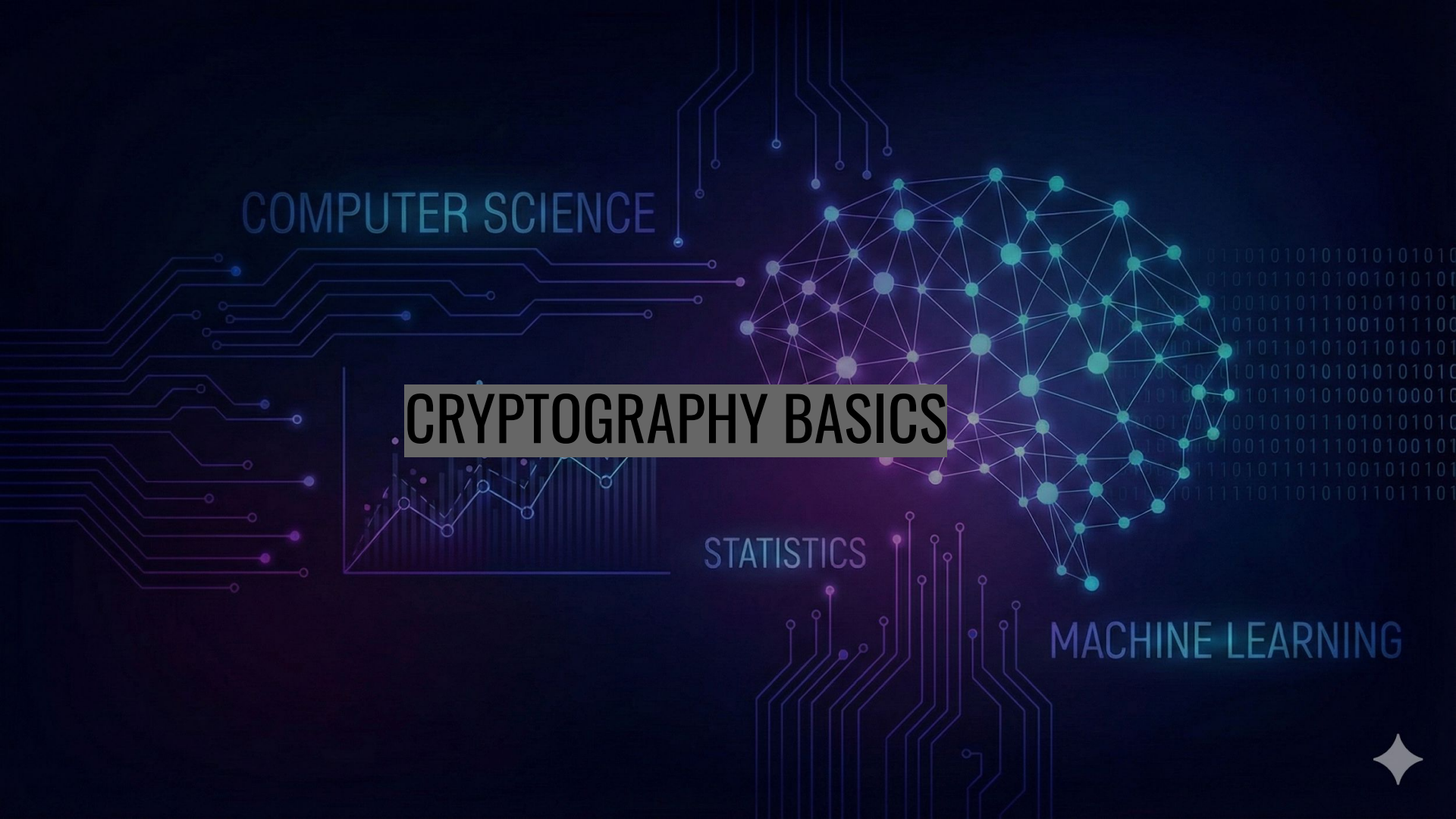
# Digital audio

- Since the human ear can hear frequencies in a certain range (about 20, 20000 Hz) the sampling theorem tells us that we should sample at 40 kHz.
- Typically, a number of quantization levels much larger than 256 is used, often 16 bits.
- For example, in the case of a **CD** we have two channels (stereo) at (practical engineering requires slightly higher than 40 kHz) **44.1 kHz and 16 bit ($2^{16}$ = 65536)**
  - So the **bit rate = 44100 samples/s * 16 bits * 2 channels = 1411200 bits/s = 176400 Bytes/s**
  - If we want **1 minute** of music we need **60*176400 Byte/s = 10584000 Bytes** which is about 10 MiB

# Final considerations




- Compression of images and sound (and therefore also video) is essentially based on the elimination of information to which the human ear and the human eye are not very sensitive.
- But not only that...

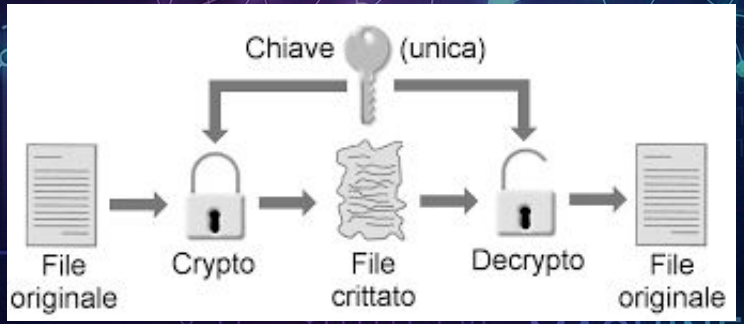

COMPUTER SCIENCE
CRYPTOGRAPHY BASICS
STATISTICS
MACHINE LEARNING

# Cryptography basics

- In computers, information is stored as sequences of bits.
- **Cryptographic techniques modify these sequences (strings) to obtain different sequences that can then be transmitted and transformed back by the recipient into the original sequence** . The mathematical functions used in the transformation use one or more secret keys.
  - **Symmetric encryption:** used since the Egyptians and the ancient Romans
  - **Asymmetric encryption** : dates back to the 70s

# Symmetric encryption

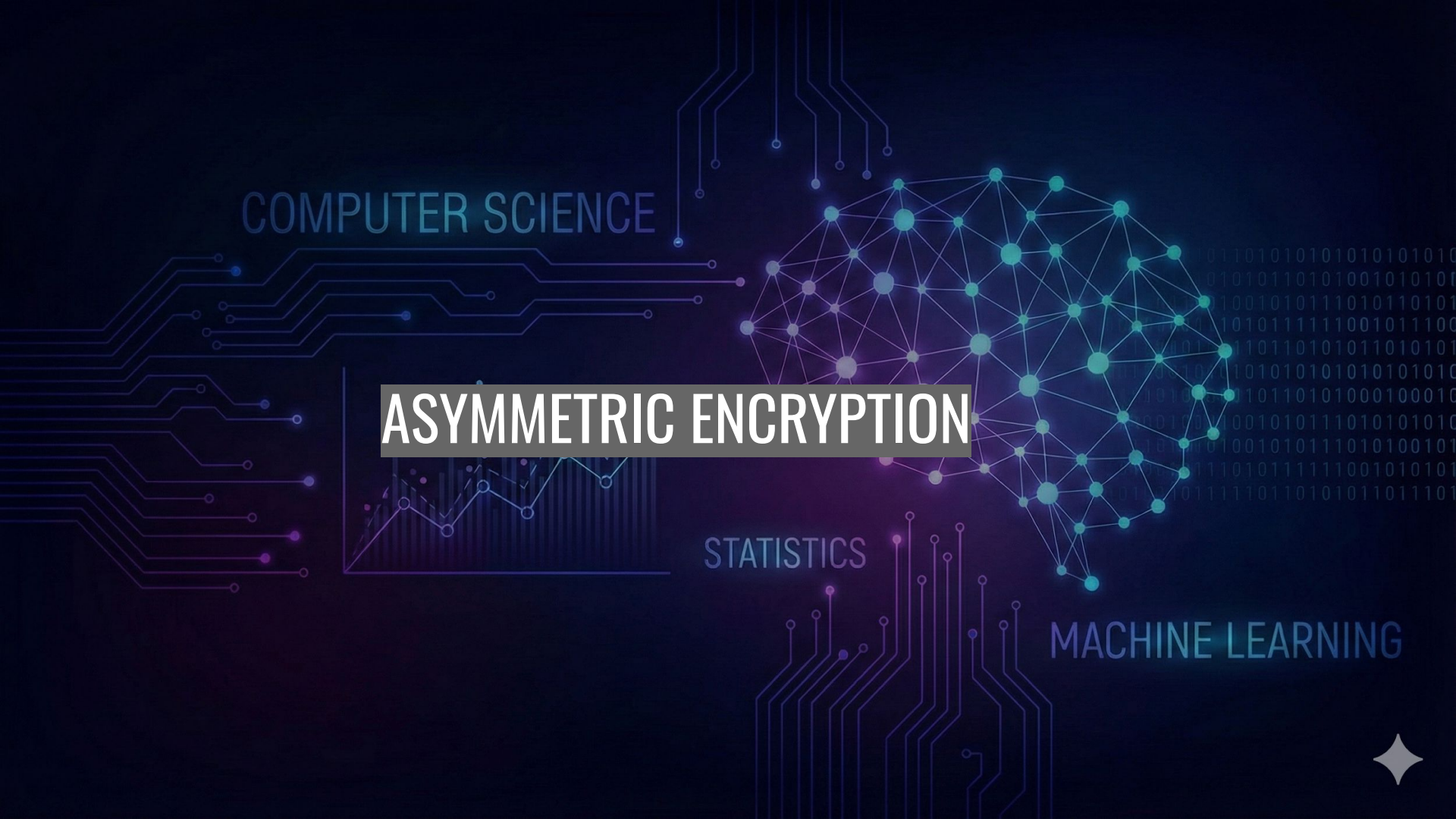
- **The key used for encryption and decryption, and therefore by the sender and recipient, is unique** . For example, in the Caesar cipher, each character is replaced with another character offset by k places (the key is the value of k).
- Monoalphabetic cipher




| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| m | n | b | v | c | x | z | a | s | d | f | g | h | j | k | l | p | o | i | u | y | t | r | e | w | q |

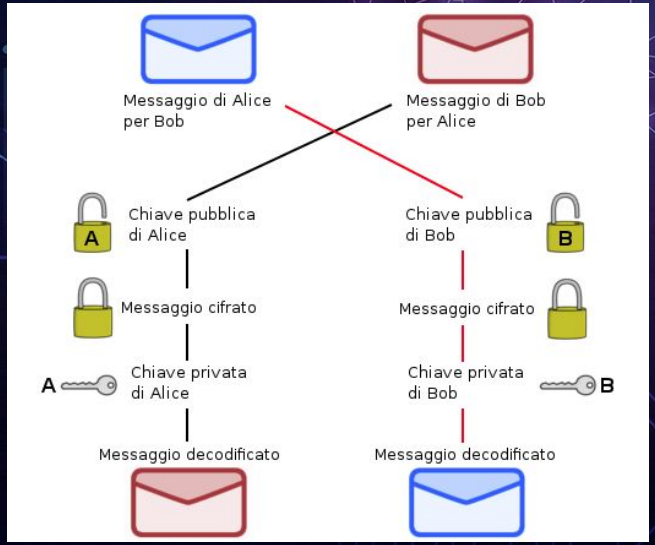
# Symmetric encryption

- **I need to find a secure way to exchange the** secret key which is unique for sender and recipient
- - force attack ( for example in the case of the Caesar cipher) I try all the values of ke I see when I get "correct" sentences and words
- Examples of modern algorithms: **Blowfish, Twofish, Standard DES or Triple DES, Standard AES** . They are all based on mathematical problems that are difficult to solve without knowing the secret key.


COMPUTER SCIENCE
ASYMMETRIC ENCRYPTION
STATISTICS
MACHINE LEARNING

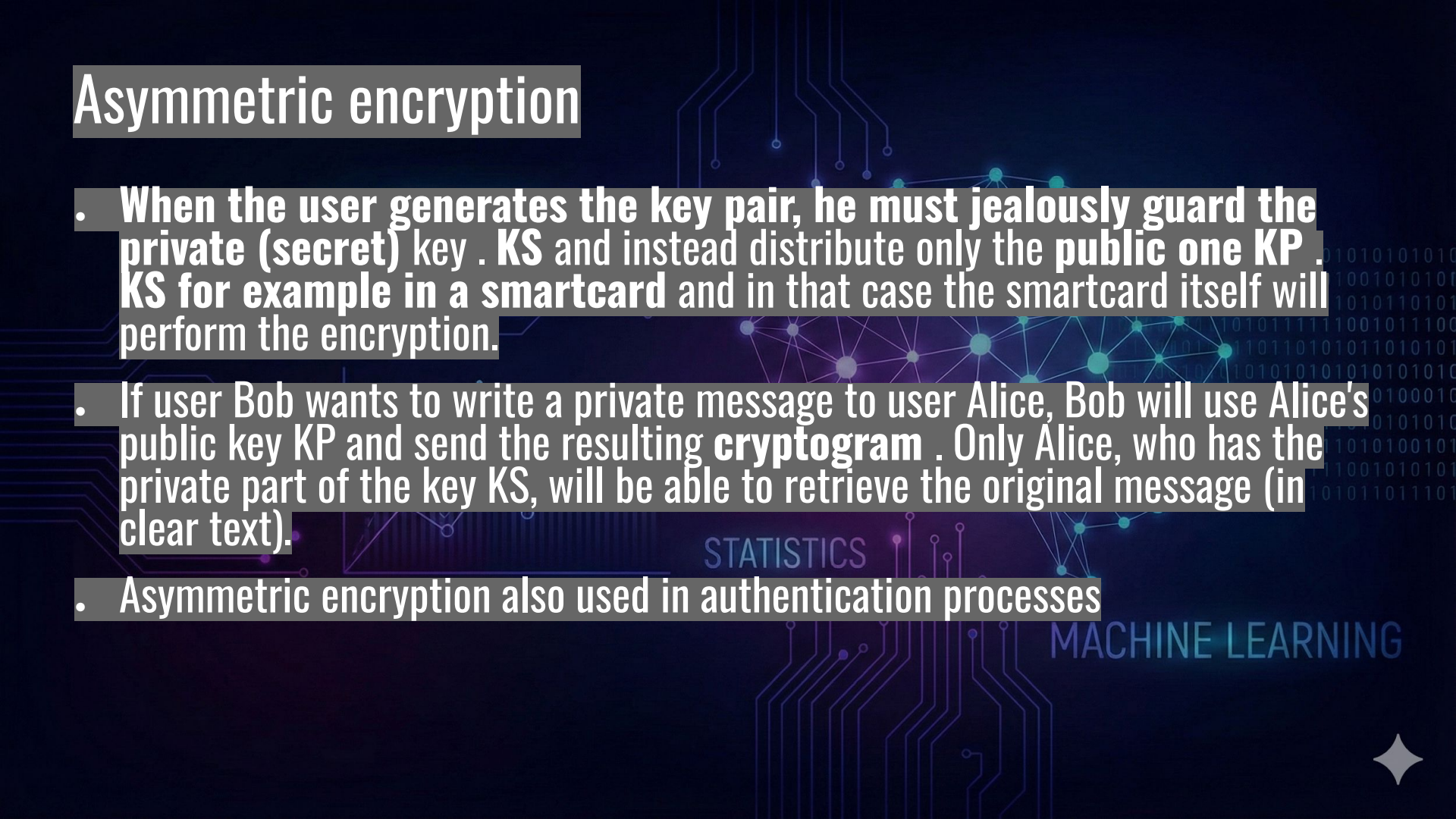
# Asymmetric encryption

**The keys used for encryption and decryption are different** . The private key, which must be kept secret, is used to decrypt, and therefore to recover the original message; the public key is used to encrypt.

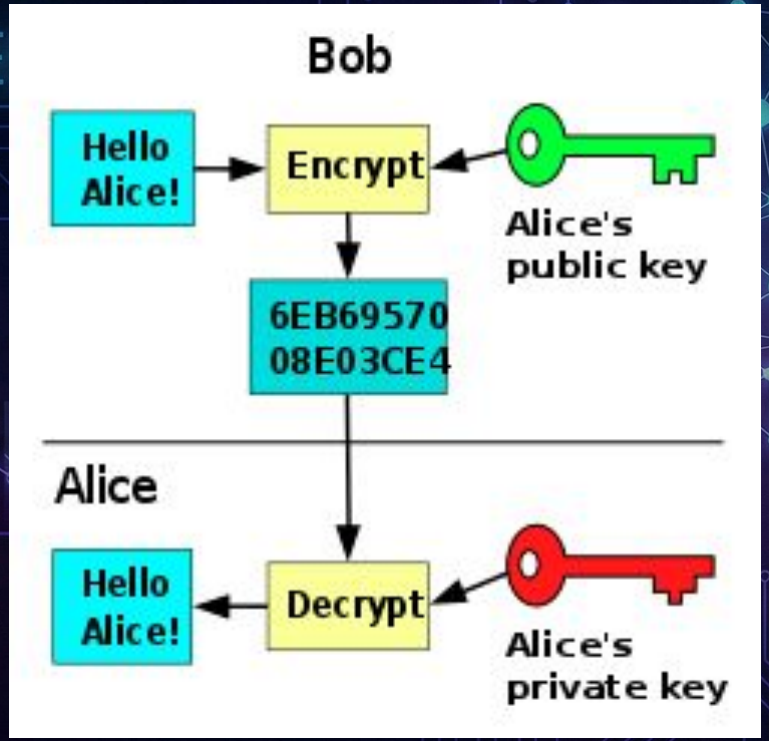


# Asymmetric encryption

- **When the user generates the key pair, he must jealously guard the private (secret)** key . **KS** and instead distribute only the **public one KP** . **KS for example in a smartcard** and in that case the smartcard itself will perform the encryption.
- If user Bob wants to write a private message to user Alice, Bob will use Alice's public key KP and send the resulting **cryptogram** . Only Alice, who has the private part of the key KS, will be able to retrieve the original message (in clear text).
- Asymmetric encryption also used in authentication processes

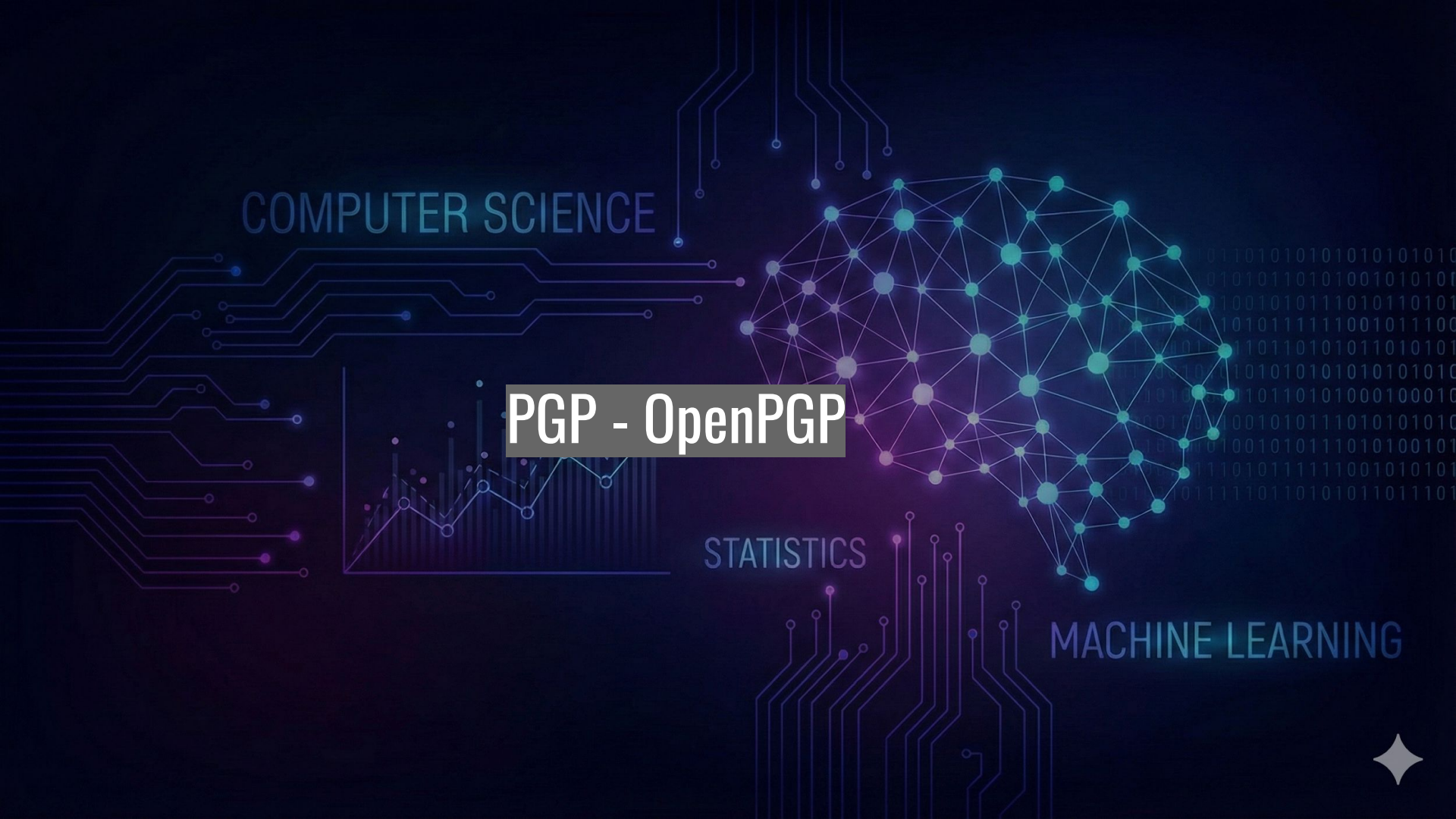


# Asymmetric encryption



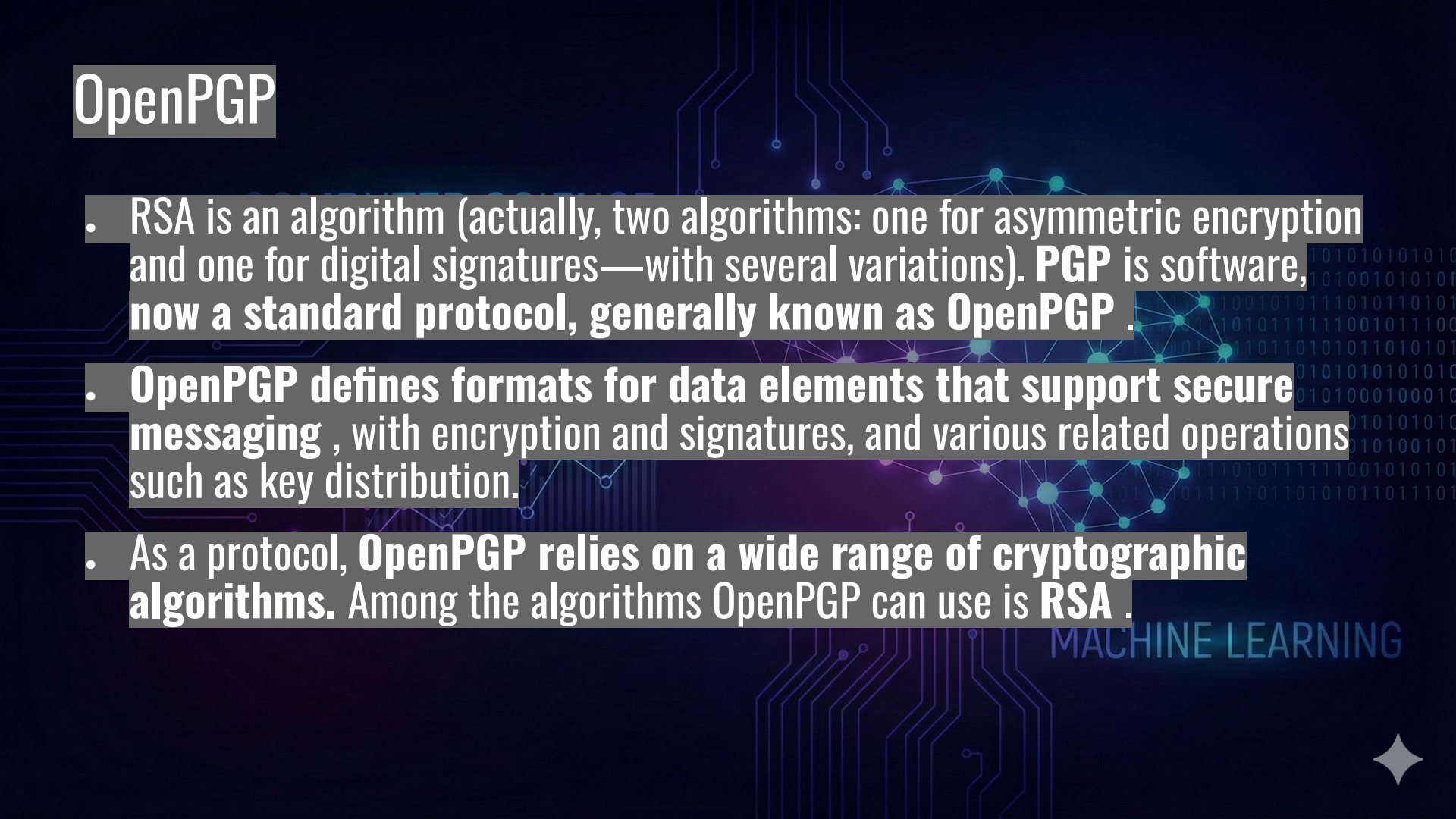
# Asymmetric encryption: RSA

- The most well-known and used algorithm is **RSA** (names of the inventors Rivest, Shamir, Adleman)
- Also for authentication or to guarantee the integrity of a document (including digital signature)
- **Based on prime numbers, that is, those natural numbers that are divisible only by 1 or by themselves (2, 3, 5, ..., 19249 · $2^{13018586} + 1$)**
- **In practice, the security relies on the difficulty of finding the prime factors of a large number (the modulus). KS and KP are mathematically derived from these factors.**
- Interest in prime numbers and factorization algorithms ( **Shor's quantum computer algorithm** )


COMPUTER SCIENCE
PGP - OpenPGP
STATISTICS
MACHINE LEARNING

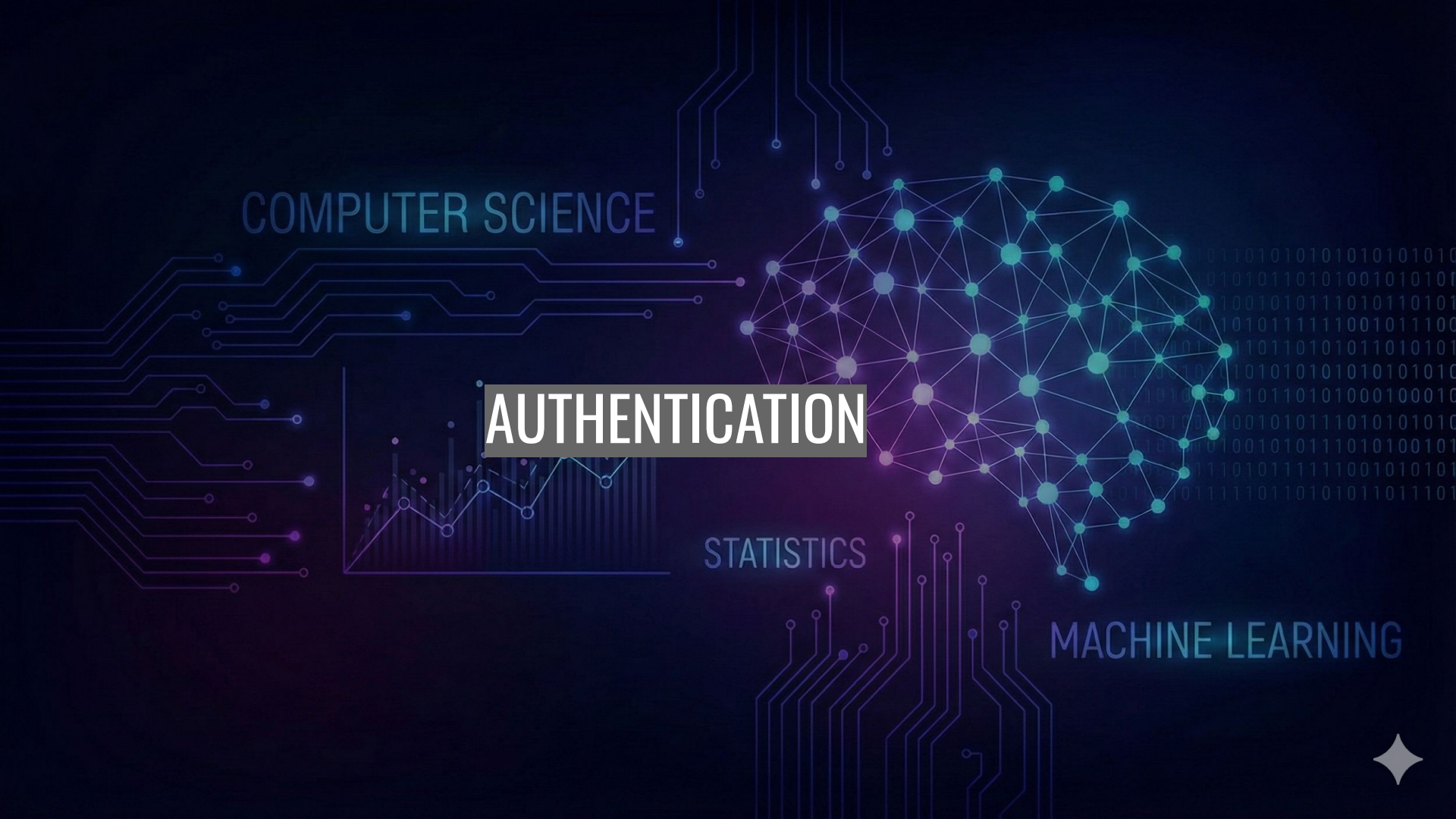
# OpenPGP

- RSA is an algorithm (actually, two algorithms: one for asymmetric encryption and one for digital signatures—with several variations). **PGP** is software, **now a standard protocol, generally known as OpenPGP** .
- **OpenPGP defines formats for data elements that support secure messaging** , with encryption and signatures, and various related operations such as key distribution.
- As a protocol, **OpenPGP relies on a wide range of cryptographic algorithms.** Among the algorithms OpenPGP can use is **RSA** .

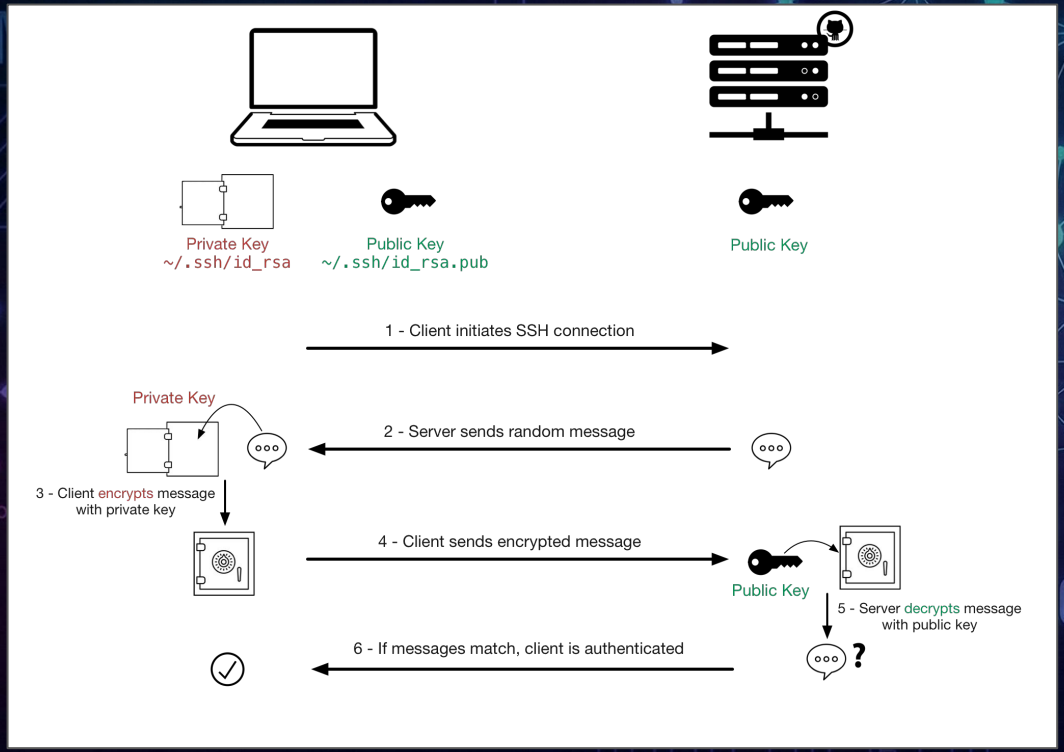


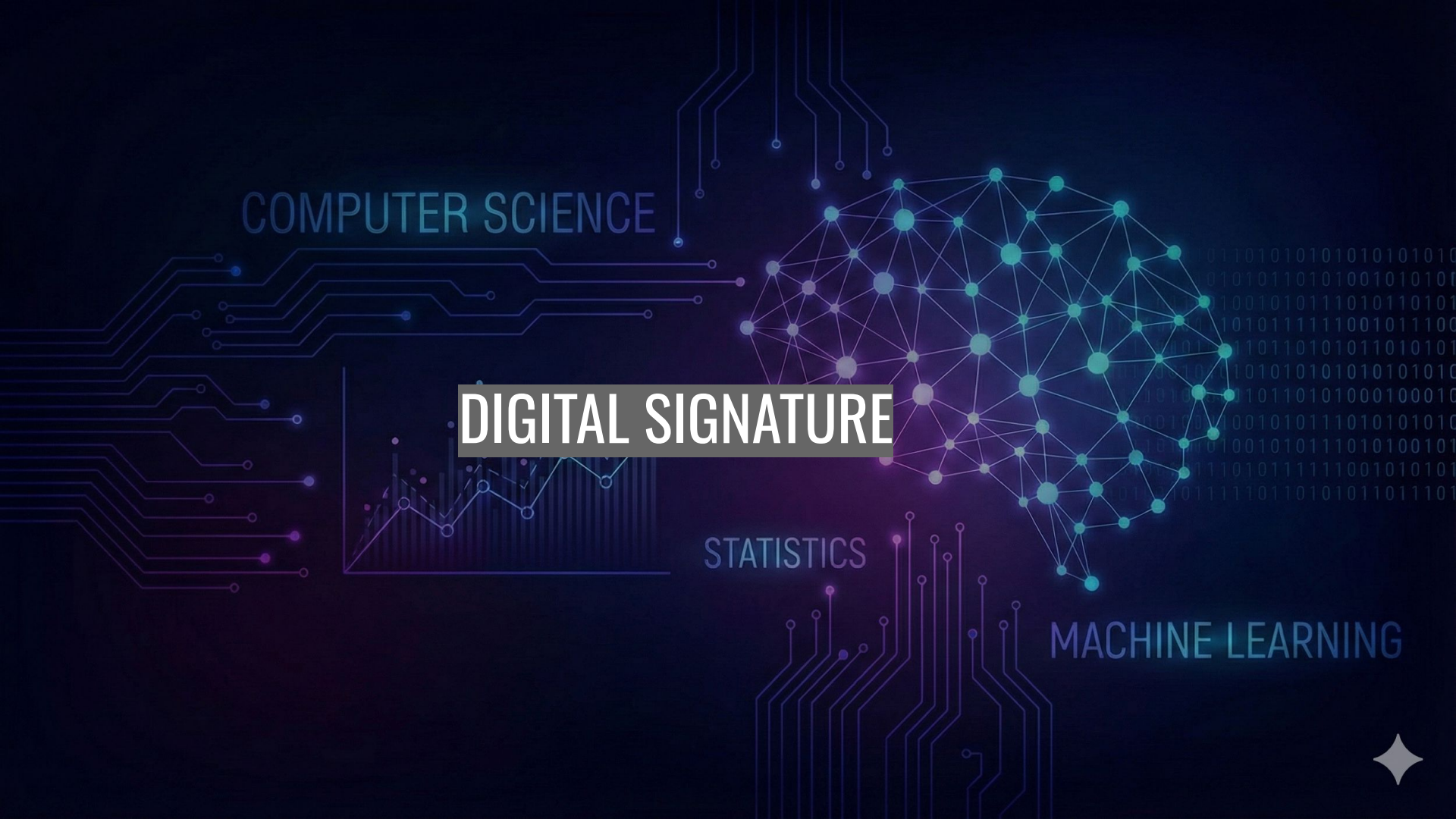
# OpenPGP

- **Philip R. Zimmermann** is the creator of **Pretty Good Privacy** , an email encryption software package. Originally conceived as a human rights tool, **PGP** was **released free of charge on the Internet in 1991** .
- This made Zimmermann the target of a three-year criminal investigation, because the government believed that U.S. export restrictions on cryptographic software were violated when PGP spread worldwide.
- **GNU Privacy Guard** (GnuPG or GPG) is free software designed to replace the PGP cryptographic suite.


COMPUTER SCIENCE
AUTHENTICATION
STATISTICS
MACHINE LEARNING

# SSH Authentication Example




COMPUTER SCIENCE
DIGITAL SIGNATURE
STATISTICS
MACHINE LEARNING

# Digital Signature

- It is affixed to digital documents in order to guarantee
  - **Authenticity** : therefore guarantee of the origin of the message
  - **Integrity** : The message has not been modified in any way
  - **Non-repudiation** : The source of the message cannot deny having signed it.
- Only the sender can add that particular signature
- Anyone can verify who signed the message (digital document, text, sound, image, video)
- **Basic ingredients of asymmetric encryption system and hash function**

# Digital Signature: Signature Methods

- **CASDES** : The file has the **pdf.p7m extension** and can be read with signature software (File Protector, DiKe, etc.).
- **PADES** : **PDF extension** : the file is signed with signature software and read with Acrobat Reader.
- **XADES : xml.p7m** extension , the xml file is automatically read by the software that receives it

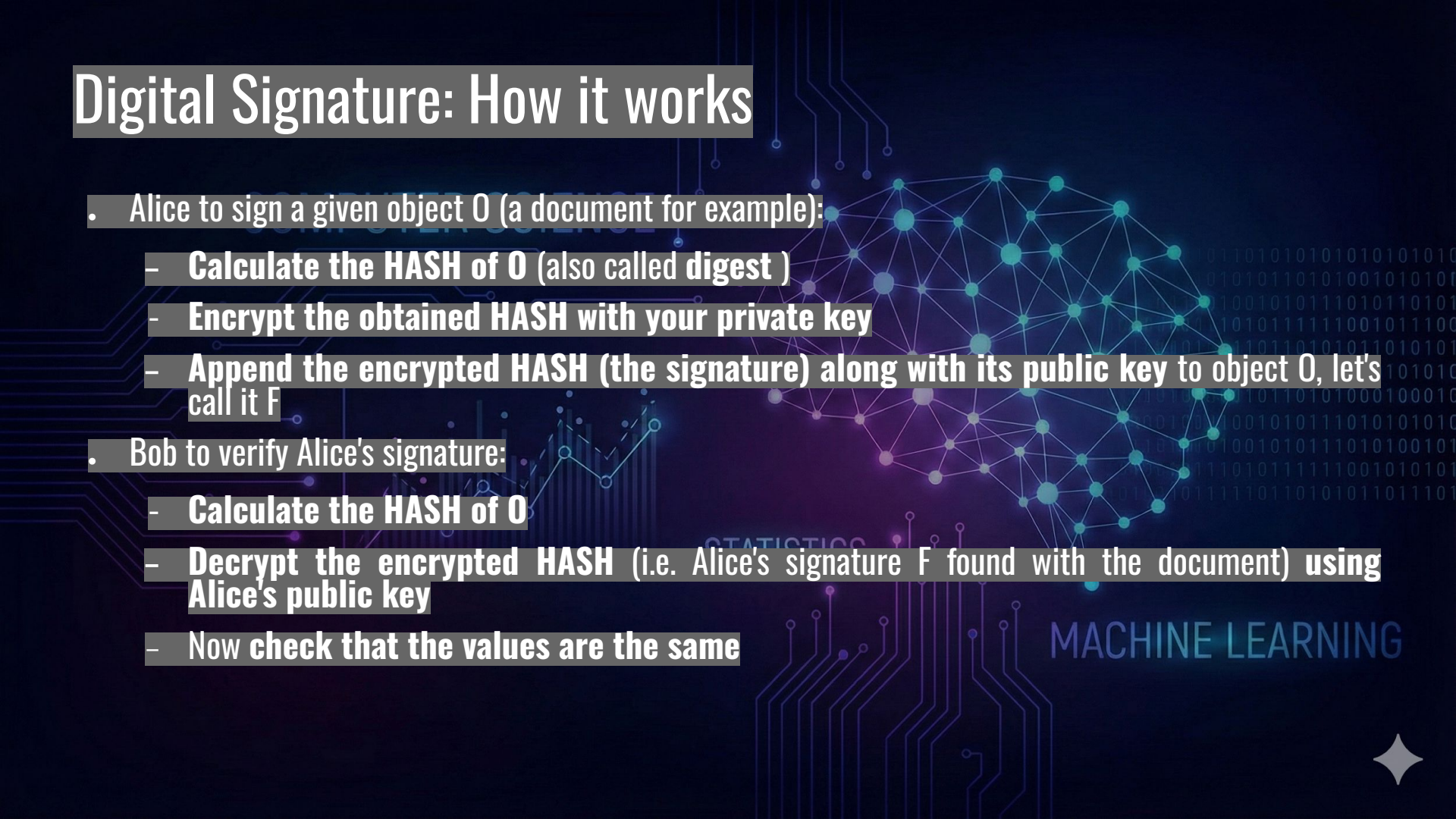
# Digital Signature: HASH

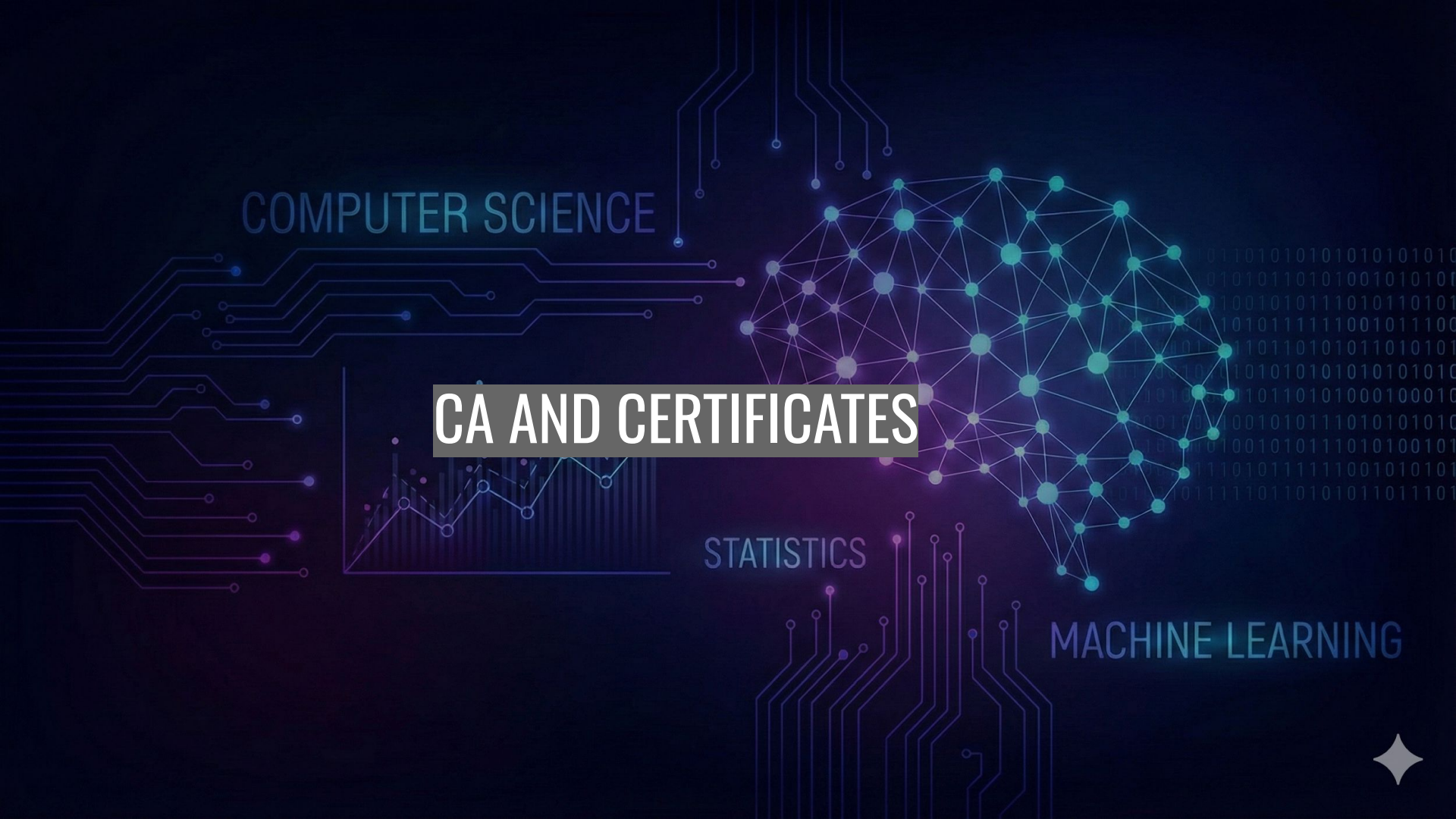
- **Hash** algorithm : **MD5, SHA-1, RIPEMD, SHA-256:**
  - A function that, given a variable-sized bitstream, returns a fixed-sized string of letters or numbers (a sort of one-time stamp)
  - The string is a unique identifier, changing just 1 bit of the source stream produces a different HASH
  - It is not invertible so it is not possible to determine the original flow from the returned string.

```
redo@raspberrypi:~$ date
Fri Nov  3 12:42:50 CET 2017
redo@raspberrypi:~$ date | md5sum
628a589b0ecb099db2cf4d9f4235f97f  -
redo@raspberrypi:~$ date | md5sum
4c0b372ca0fe4cd391d1eb02deaae7b8  -
redo@raspberrypi:~$
```

# Digital Signature: How it works

- Alice to sign a given object O (a document for example):
  - **Calculate the HASH of O** (also called **digest** )
  - **Encrypt the obtained HASH with your private key**
  - **Append the encrypted HASH (the signature) along with its public key** to object O, let's call it F
- Bob to verify Alice's signature:
  - **Calculate the HASH of O**
  - **Decrypt the encrypted HASH** (i.e. Alice's signature F found with the document) **using Alice's public key**
  - Now **check that the values are the same**



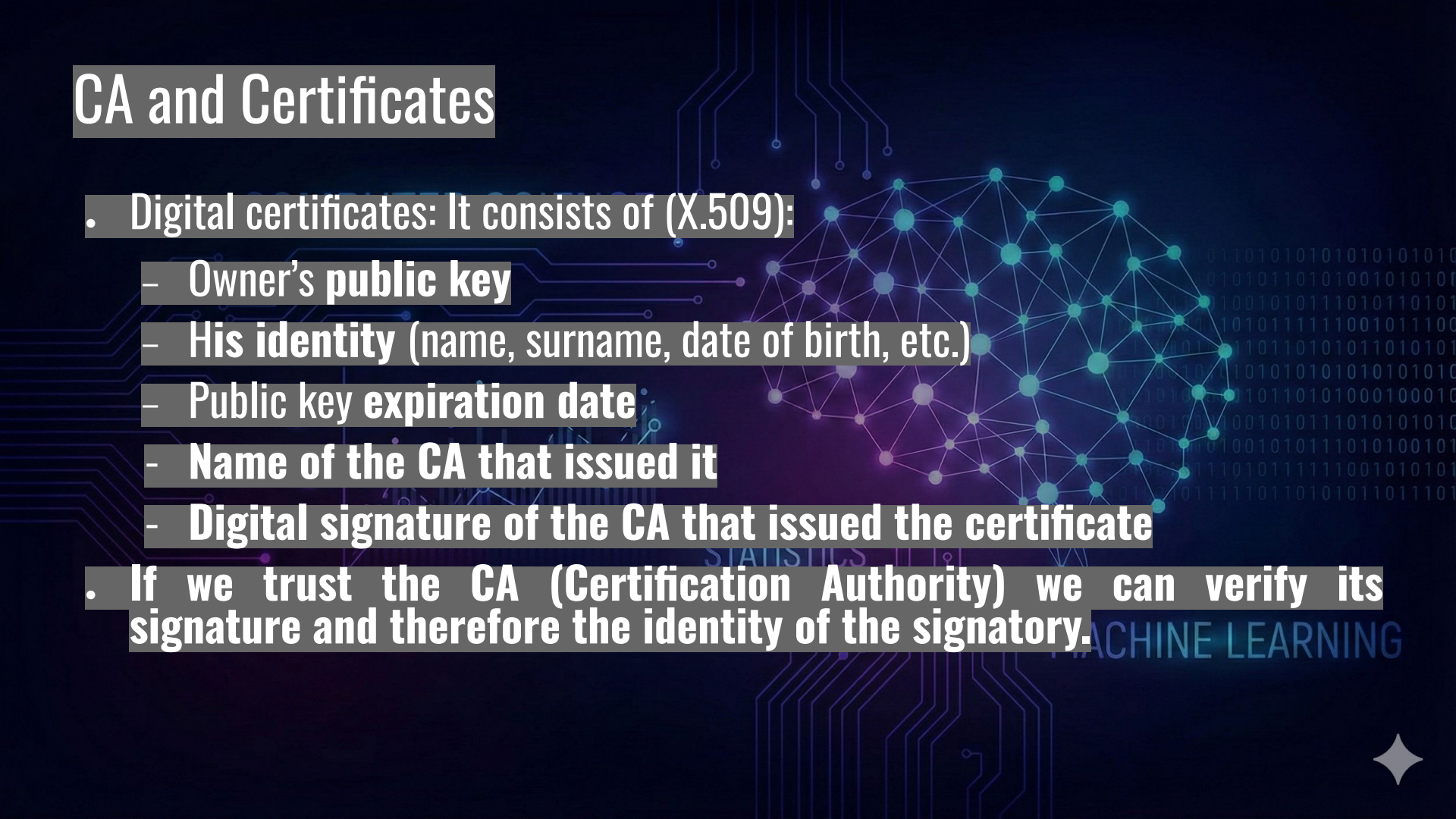

COMPUTER SCIENCE
CA AND CERTIFICATES
STATISTICS
MACHINE LEARNING

# CA and Certificates

- How can I be sure that the signature used actually belongs to the signer? To paraphrase, how can I be sure of the user-public key association?
- **Digital certificates serve this purpose. They contain a variety of information, such as the public key and user data.**
- Just as paper certificates allow us to have information about the user
- **CA, Certification Authority** , guarantees the association between the **Public Key** and the owner's identity

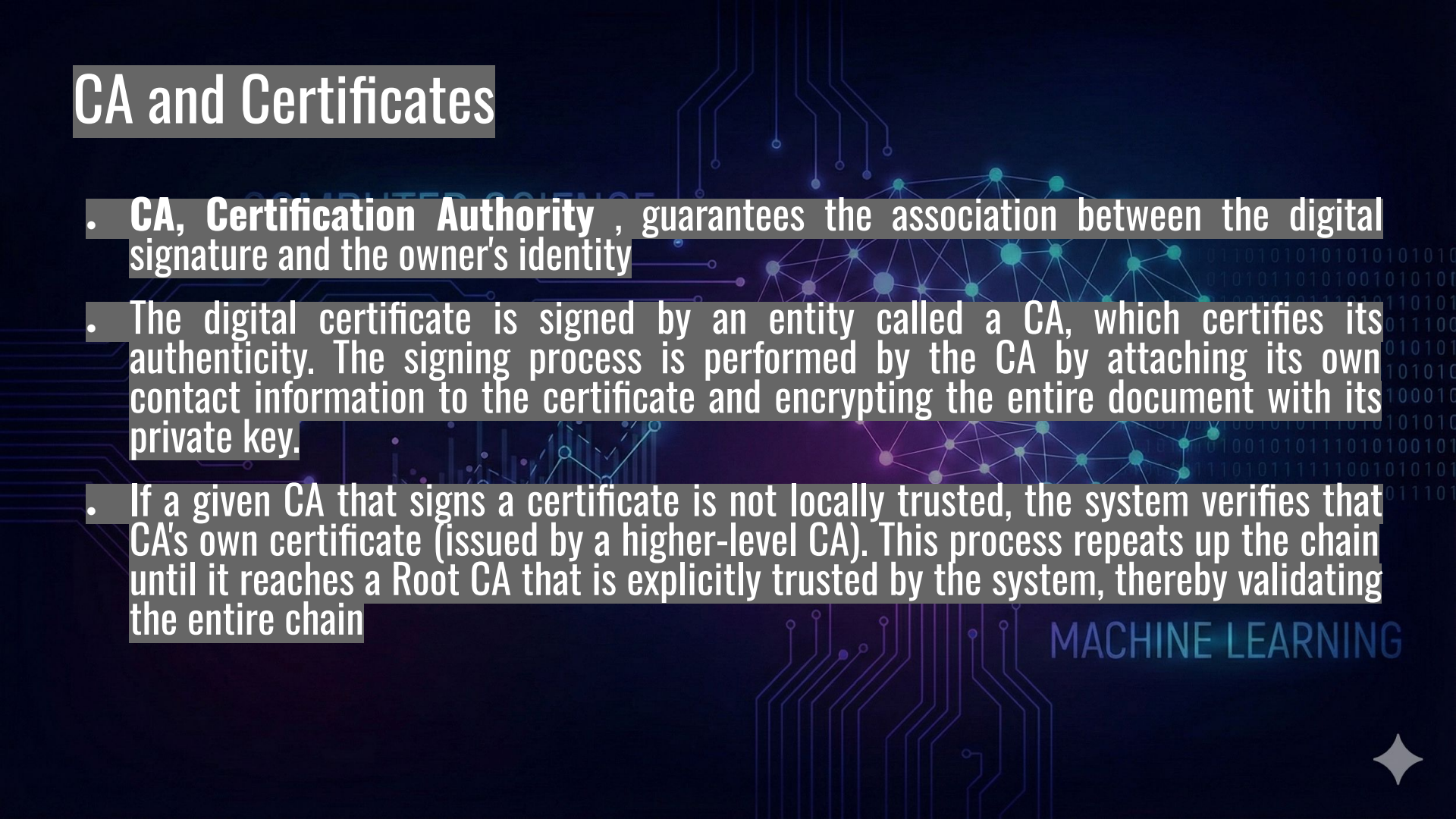
# CA and Certificates

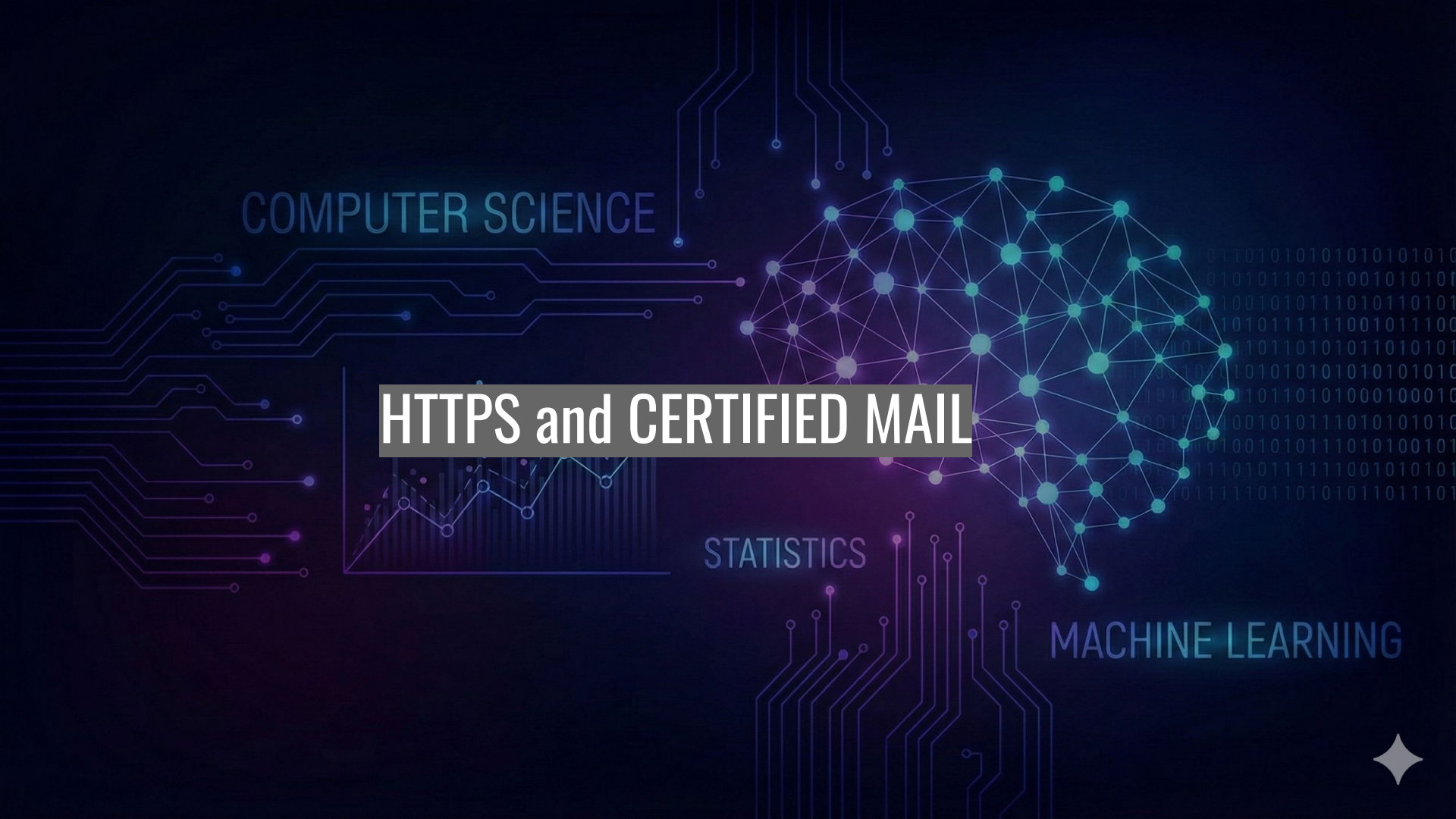
- Digital certificates: It consists of (X.509):
  - Owner's **public key**
  - **His identity** (name, surname, date of birth, etc.)
  - Public key **expiration date**
  - **Name of the CA that issued it**
  - **Digital signature of the CA that issued the certificate**
- **If we trust the CA (Certification Authority) we can verify its signature and therefore the identity of the signatory.**



# CA and Certificates

- **CA, Certification Authority** , guarantees the association between the digital signature and the owner's identity
- The digital certificate is signed by an entity called a CA, which certifies its authenticity. The signing process is performed by the CA by attaching its own contact information to the certificate and encrypting the entire document with its private key.
- If a given CA that signs a certificate is not locally trusted, the system verifies that CA's own certificate (issued by a higher-level CA). This process repeats up the chain until it reaches a Root CA that is explicitly trusted by the system, thereby validating the entire chain




COMPUTER SCIENCE
HTTPS and CERTIFIED MAIL
STATISTICS
MACHINE LEARNING

# HTTPS

- Fundamental protocol for example in e-commerce and home banking
- When I connect to a site via https:
  - **The server declares its identity by sending its** public key certificate **guaranteed by a CA**
  - **My browser (client) verifies Hostname/Domain matches the identity contained in the certificate**
  - **In modern HTTPS (TLS 1.3), the client and server use a mathematical method called Diffie-Hellman to generate the same key independently. The key never travels across the network, so even if the server's Private Key is stolen later, past conversations remain safe.**

# Certified mail

Unlike standard email where you just send data to a server, in the PEC system, both the Sender's Provider and the Recipient's Provider must be certified and accredited by a government body (in Italy, AgID). They act as the guarantors of the communication

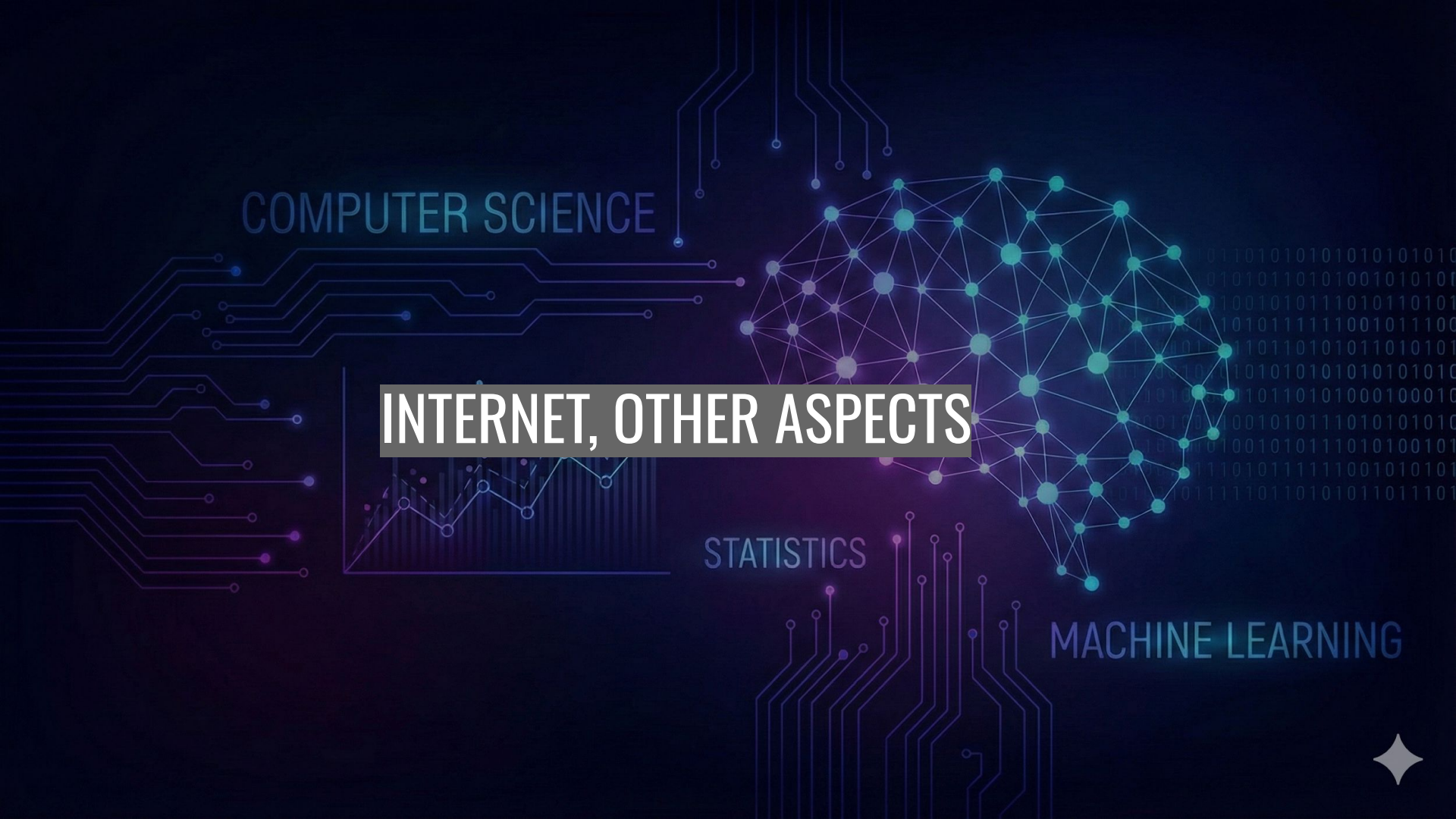
- Step A: Submission & Acceptance (Ricevuta di Accettazione)
  - You (Sender) send the email to your PEC provider.
  - Your Provider checks the email for viruses and compliance.
  - Your Provider issues a Ricevuta di Accettazione (Acceptance Receipt) back to you.
    - What it proves: This receipt is digitally signed by your provider. It legally proves when (timestamp) you sent the message and who you sent it to. Even if the recipient claims they never got it, you have proof you sent it.

# Certified mail

- Step B: The Transport Envelope (Busta di Trasporto)
  - Your Provider does not just forward your email. It wraps your original message inside a new digital package called a Busta di Trasporto (Transport Envelope).
  - Your Provider digitally signs this envelope.
    - What it proves: This guarantees Integrity. The recipient's provider can verify that the message has not been altered in transit (no Man-in-the-Middle attacks).

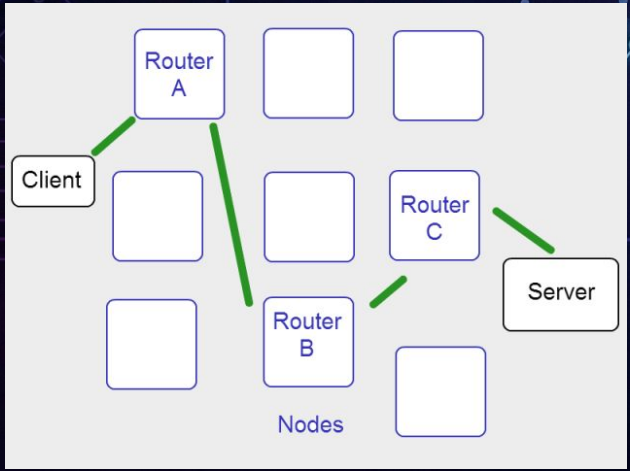
# Certified mail

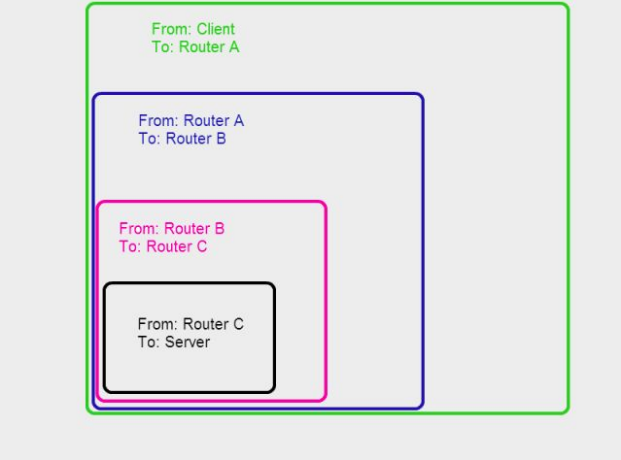
- Step C: Delivery (Ricevuta di Consegna)
  - Your Provider sends the envelope to the Recipient's Provider.
  - The Recipient's Provider verifies the digital signature on the envelope (checking it came from a valid PEC provider).
  - The Recipient's Provider puts the message into the recipient's inbox.
  - The Recipient's Provider issues a Ricevuta di Consegna (Delivery Receipt) and sends it back to You (Sender).
    - What it proves: This is the most critical receipt. It legally proves the message reached the destination's address. It is the legal equivalent of the "return receipt" (cartolina di ritorno) in traditional registered mail.
    - Note: It proves delivery to the inbox, not that the user actually read it. (Just like a registered letter delivered to your mailbox is considered legally served even if you don't open it).


COMPUTER SCIENCE
INTERNET, OTHER ASPECTS
STATISTICS
MACHINE LEARNING

# Deep Web, Tor

- Onion routing. The Tor network is made up of volunteers who use their own computers as nodes:
- I can reach “hidden” services , but also normal servers via exit nodes

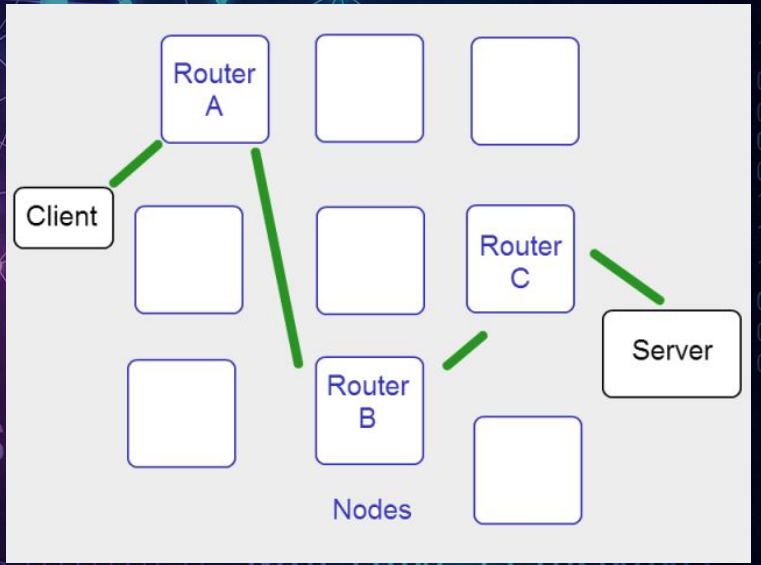


# Deep Web, Tor

Normal Servers via Exit Nodes: If you use Tor to browse the "Clear Web" your traffic leaves the Tor network through an Exit Node.

1. The Tor client constructs a random circuit through a series of volunteer nodes to connect to the destination server. Path Selection: Your client randomly selects three nodes from this list to form a "circuit":
   a. Entry Node (Guard): The first hop. It sees your IP address.
   b. Middle Node: The second hop. It acts as a buffer.
   c. Exit Node: The final hop. It will connect to the destination server for you.

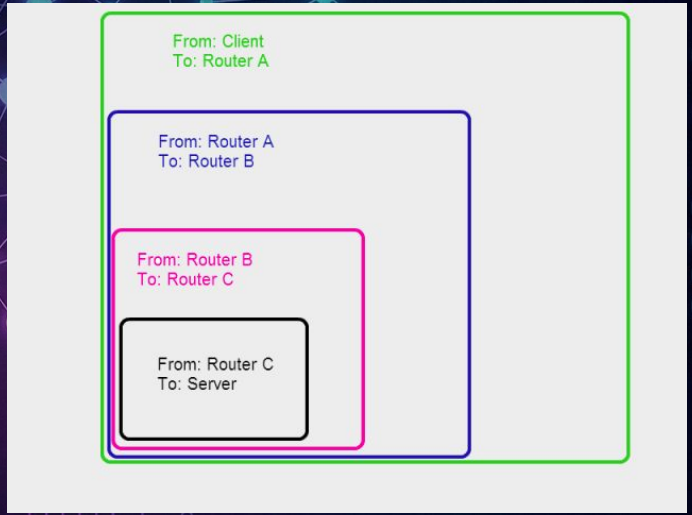


# Deep Web, Tor

Layered Encryption: Your client encrypts the data three times (like an onion):

1. Layer 1 (Outer): Encrypted for the Entry Node.
2. Layer 2 (Middle): Encrypted for the Middle Node.
3. Layer 3 (Inner): Encrypted for the Exit Node.

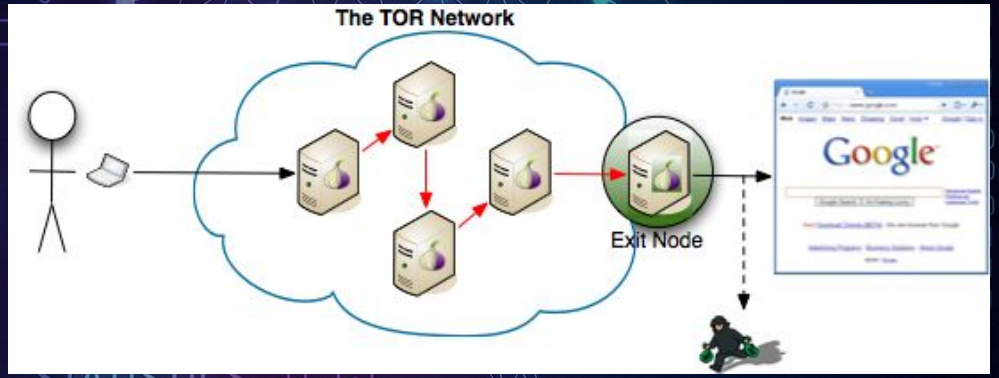
The data packet travels through the circuit, getting "peeled" at each step.



# Deep Web, Tor, Bitcoin

The Server's View: The website (Google) sees a request coming from the Exit Node's IP address. It has no way of knowing your IP address.

- Entry Node: Sees the Client (You) and the Middle Node. $\rightarrow$ It sees the Client!
- Middle Node: Sees the Entry Node and the Exit Node. $\rightarrow$ It sees neither Client nor Server.
- Exit Node: Sees the Middle Node and the Server. $\rightarrow$ It sees the Server!



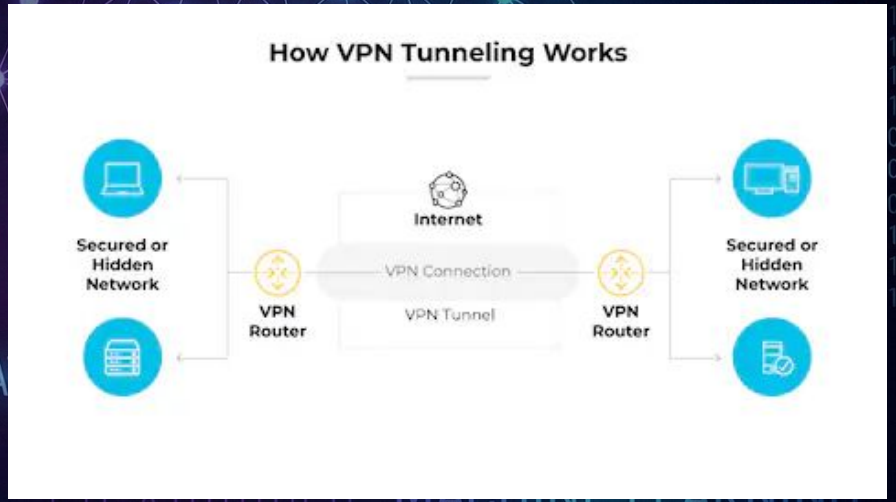
# Deep Web, Tor, Bitcoin

Tor Hidden Services (addresses ending in .onion).
The Onion Server wants to stay hidden. It doesn't publish its IP.

- Introduction Points: The server picks 3 random relays in the network to act as its "Introduction Points."
- It builds encrypted circuits to them and tells them: "If anyone wants to talk to me, forward their message here."

# VPN: Virtual Private Network

- The Secure Tunnel: Think of it as driving a private armored car through a public highway. The VPN creates an encrypted connection over the public Internet, isolating your data from other traffic
- .Encapsulation: Your original data packets (Letter) are wrapped inside new VPN packets (Envelope). The ISP only sees the outer envelope, not the letter inside.
- IP Masking: You connect to a VPN Server, and the Server connects to the website. The website sees the Server's IP address, not yours, effectively hiding your location.



# Bitcoin

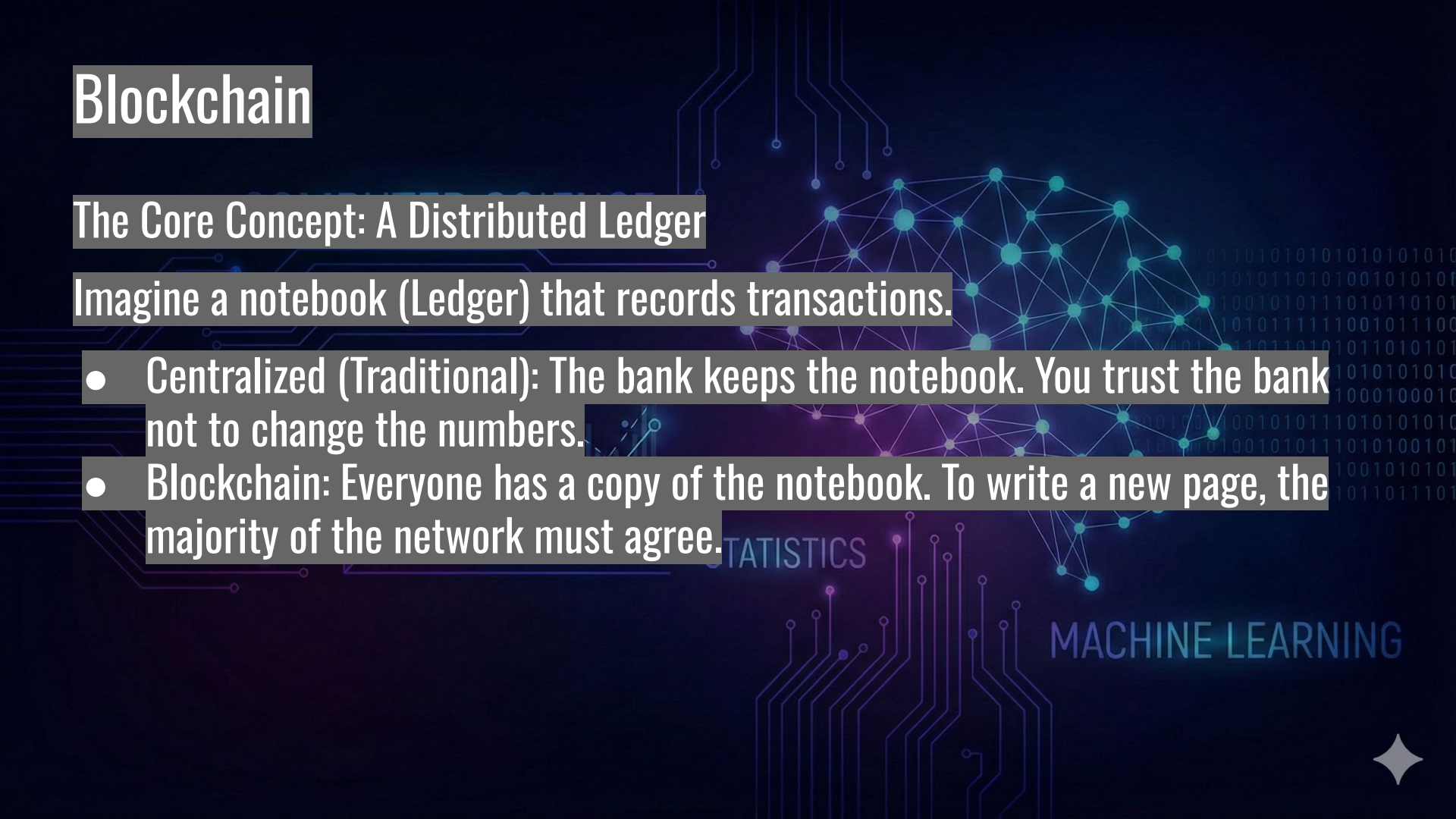
- **Bitcoin** (I quote Wikipedia) Unlike most traditional currencies, Bitcoin does not use a central authority: it uses a database distributed among the nodes of the network that keep track of transactions, but uses cryptography to manage functional aspects, such as the generation of new money and the attribution of ownership of bitcoins.
- Based on cryptography and hashing algorithms (SHA-256), Bitcoin uses the SHA-256 hash algorithm to generate verifiable "random" numbers in a way that requires a predictable amount of computation time. Generating a SHA-256 hash with a value lower than the current target solves a block.

# Blockchain

## The Core Concept: A Distributed Ledger

Imagine a notebook (Ledger) that records transactions.

- Centralized (Traditional): The bank keeps the notebook. You trust the bank not to change the numbers.
- Blockchain: Everyone has a copy of the notebook. To write a new page, the majority of the network must agree.



# Blockchain

## The Block (The Container)

- Each block contains three main things:
  - **Data**: The transactions (e.g., "Alice sends 5 BTC to Bob").
  - **The Hash**: The block's unique digital fingerprint (SHA-256).
  - **The Previous Hash**: This is the magic glue. **Each block contains the Hash of the previous block.**

# Blockchain

## The Chain (The Link)

- Because Block 3 contains the Hash of Block 2, and Block 2 contains the Hash of Block 1, they are mathematically linked.
  - Block 1 (Genesis): [Data | Hash: A123 | Prev: 0000]
  - Block 2: [Data | Hash: B456 | Prev: A123]
  - Block 3: [Data | Hash: C789 | Prev: B456]

# Blockchain

Why is this secure? (Immutability) If a hacker tries to change a transaction in Block 1:

- The data in Block 1 changes.
- Therefore, the Hash of Block 1 changes (from A123 to X999).
- Block 2 immediately becomes invalid because it says "Previous: A123", but the real previous block is now "X999".
- To fix this, the hacker would have to re-calculate the hashes for Block 2, Block 3, and every subsequent block