Sovereign Cognitive Intelligence for a Safer Europe

A Symbolic Cognitive Platform for Law Enforcement, Defense, and Critical European Organizations

Where are we today ?

In an era of increasing cyber threats, disinformation campaigns, and complex information warfare, organizations require reliable tools to transform raw textual data into actionable intelligence. Threat intelligence has become essential for anticipating and mitigating risks before they materialize.

Years ago, employees in organizations spent hours manually reading reports, emails, and messages to extract precise meaning before deciding and acting. Traditional technologies simply couldn't modernize these processes.

But today, in defense, law enforcement, and critical sectors, AI and Natural Language Understanding technologies (NLU) process vast volumes of text at unprecedented scale, flagging keywords, spotting patterns, and raising alerts faster than any human team.

But here is the problem. These solutions rely on foreign technical stacks that introduce unacceptable sovereignty and security risks, facilitating potential exfiltration of sensitive or classified data. They are black boxes, prone to hallucinations, vulnerable to bias poisoning and adversarial attacks. In high-stakes missions, unverifiable outputs and compliance gaps are simply not acceptable, costing lives, strategic assets, and European autonomy.

There is no sovereignty

Without going into too much detail, on the server side, the vast majority of Machine Learning (ML) or Large Language Models (LLM) solutions rely on a set of ready-to-use binaries and libraries, bundled together in a framework (a collection of packages) called PyTorch. This framework depends on the Python programming language, and the first thing to do is to install it.

When you install Python, you install a set of interconnected packages: the language interpreter, binaries, libraries, and the large standard library, for a total of typically several thousand files (exact numbers vary by version and installed packages, often around 2,000 to 3,000 or more for a full installation).

You can then install PyTorch, which adds a significant volume of files (typically tens of thousands depending on the configuration, with a full CUDA installation taking up several GB on disk). This installation contains Python code, compiled libraries, data, resources, and configuration files.

CUDA (Compute Unified Device Architecture) is a proprietary parallel computing platform and programming model developed by NVIDIA. It allows developers to use NVIDIA GPUs for general-purpose computing (not just graphics), dramatically accelerating complex mathematical operations such as matrix multiplications, which are at the heart of training and running Large Language Models. In simple terms, CUDA is the software layer that gives programmers direct access to the massive parallel processing power of NVIDIA GPUs. It is currently the dominant standard in the AI industry because most popular frameworks like PyTorch and TensorFlow are heavily optimized for it. This creates a strong dependency: if you want maximum performance for LLMs today, you almost always need CUDA-compatible NVIDIA hardware.

Among all these files, the "Python" part corresponds to code readable by developers (the so-called "open-source" part). The PyTorch repository contains roughly 1.5 million lines of Python code (plus hundreds of thousands of lines in C/C++), though exact installed figures vary. The majority of high-performance computations are handled by compiled native libraries (whose full source code is not always directly accessible), which can represent several GB of binaries depending on the setup (CPU-only vs. CUDA). The estimated size of the total source code (including dependencies and backends) is substantial and can reach tens of millions of lines depending on compilation options.

All this is so to say the standard software stack !

The main organizations involved in the development and maintenance of Python and PyTorch include the Python Software Foundation (PSF, United States), a non-profit organization that manages Python and is funded by major sponsors such as Anthropic, for example.

The PyTorch Foundation (United States), hosted by the Linux Foundation (United States), oversees PyTorch with major contributions from Meta (United States, the project's origin), NVIDIA, AMD, Intel, Microsoft, AWS (Amazon), and other members such as Carnegie Mellon University, Monash University, the National IT Industry Promotion Agency (NIPA, South Korea), Nota AI (South Korea), Clockwork.io, Emmi AI, yasp.ai, CommonAI CIC (United Kingdom), and the University of Leicester (United Kingdom). Its funding relies on donations, memberships, and events.

Triton, integrated into PyTorch for GPU kernels, originated from OpenAI but is now maintained by an open-source community with contributions from Meta, AMD, Intel, and NVIDIA, through standard open-source mechanisms.

Finally, NVIDIA (United States) is the most dominant player in hardware/software acceleration for PyTorch. It is a "Premier" member of the Foundation and is described as a principal technical collaborator and major financial sponsor. NVIDIA is injecting significant investments into the PyTorch ecosystem and open-weight models (for example, up to $26 billion over 5 years announced in 2025/2026 for model development and ecosystem support).

To conclude and be complete, as of March 2026, several Large Language Models (LLMs) explicitly present themselves as European, often emphasizing digital sovereignty, GDPR/AI Act compliance, training on European supercomputers, and a degree of independence from American Big Tech. These include Mistral AI (France), EuroLLM / OpenEuroLLM (pan-European consortium), TildeOpen, Aleph Alpha (Germany), and others. They all use PyTorch (and often CUDA backends). In summary, these companies that present themselves as "European" are essentially local operators within a globally American-dominated technical system.

There is no technological sovereignty here, and the entire system remains heavily dependent on primarily US-developed and maintained technologies (Python, PyTorch, CUDA, etc.).

Poor security

In a theoretical scenario where a corporate LLM is deployed on a machine exposed to the Internet, the exfiltration of sensitive data embedded in the model or the system represents a major security risk.

For example, an Internet-exposed machine may be vulnerable to attacks such as malicious prompt injections via an API or a poorly secured web interface. An attacker could theoretically force the LLM to regurgitate fragments of sensitive training data (through extraction or model inversion attacks).

LLMs based on PyTorch can unintentionally memorize and reproduce training data (a phenomenon known as memorization). In an exposed context, an attacker can use iterative extraction techniques (advanced prompt engineering, membership inference attacks, or model inversion) to attempt to recover sensitive information without directly compromising the machine itself.

Since PyTorch is open source, an attacker could theoretically exploit vulnerabilities in the framework or its dependencies (third-party libraries such as NumPy, CUDA, etc.), or through supply-chain attacks if updates are not properly verified.

If the machine is compromised, an attacker could access the model files (weights), residual datasets, checkpoints, or attempt to reconstruct data using inversion or extraction techniques.

These risks are greatly amplified by Internet exposure (which multiplies the attack surface: open ports, inadequate firewalls, exposed APIs, etc.) and by the very nature of LLMs, which “learn” data without explicitly forgetting it.

These risks are even higher in the cloud. Even if the hosting provider promises not to use the company’s data to train its own models, prompts, datasets, and results still pass through its servers. There is an inherent risk of cross-border data transfers (often to U.S. data centers), subject to extraterritorial laws such as the U.S. CLOUD Act. The company then largely depends on the provider’s good faith and controls !

Now let us assume the LLM is only accessible from within the company (internal network, no direct Internet connection). Even then, risks remain: the server is connected to other internal machines, and compromise can occur via an infected workstation, lateral movement attacks, insider threats, or human error. The attack surface is reduced but still exists.

In the most extreme case, the machine is “air-gapped”: no network connection at all (no Ethernet, no Wi-Fi, no Bluetooth, no cables to other systems). The machine is completely physically isolated. Data transfer can only occur via removable physical media (USB drives, external disks, etc.). Remote exfiltration becomes impossible without direct physical access.

However, physical access remains a risk vector. Any person with legitimate physical access (for maintenance, updates, training data transfer, backups, or normal use) could potentially copy the model files or extract data onto removable media. Additional controls (encryption, Hardware Security Modules, physical supervision, strict policies) can minimize this risk but cannot eliminate it entirely.

Finally, most of these issues would not exist (or would be far more limited) if the machine were not “learning.” By processing your documents, the model incorporates and memorizes part of the sensitive information, which can facilitate its later extraction.

It is also worth explaining what happens in Python when managing memory. Python uses automatic garbage collection and memory management optimized for performance. When a variable is created, Python allocates the necessary memory for the object but does not systematically zero out the raw memory area before writing the new data. As a result, this memory may contain remnants of previous objects. When the variable is freed (end of function, reference deletion), Python does not zero the freed memory by default either. The old bytes remain in memory until it is reallocated and overwritten by a new object. This is a performance optimization, but it creates a risk of “data remanence.”

To force secure wiping (zeroing) of sensitive data, Python alone is not sufficient. You must use specialized libraries (e.g., secrets, cryptography with manual wiping) or switch to lower-level languages like C/C++ where memory can be precisely controlled.

Thus, during the training or fine-tuning phase of an LLM, sensitive data from documents passes through many Python layers (tokenizers, dataloaders, etc.) before being integrated into the model. Residues may persist in RAM after processing. An attacker with access to memory (via process inspection, core dump, or compromise) could theoretically recover them. This risk is higher in shared environments (cloud, multi-tenant containers) or during long training runs. It is a well-known issue called memory remanence or data remanence.

That said, at runtime, an attacker with access to the process, even briefly, can use a single command (for example with tools like gcore, dd on /proc/<pid>/mem,

or memory dumping utilities) to take a full snapshot of the process memory and easily exfiltrate this data.

Another critical weakness is that Python-based applications frequently leave sensitive data unencrypted on the hard drive. Temporary files, cached models, serialized objects (pickle files), logs, or database dumps created during execution are often stored in plaintext. These files remain vulnerable to theft, forensic analysis, or unauthorized access long after the program has run, especially on shared or compromised servers.

Finally, another critical security risk lies in the ease with which malicious code can be introduced into Python-based LLM solutions, particularly those relying on PyTorch and its extensive ecosystem of libraries.

Because Python is an interpreted language, most dependencies are distributed as source code or pre-compiled binary wheels that can be relatively easily modified by an attacker.

A malicious actor with access to a supply chain, such as a compromised PyPI package, a forked GitHub repository, a modified wheel during the build process, or even a compromised continuous integration pipeline, can subtly alter a few lines of code in PyTorch, TorchVision, Transformers, or any supporting library to insert backdoors, data exfiltration routines, or hidden triggers.

These modifications can be made to look perfectly legitimate and are often difficult to detect through standard code reviews, especially when they involve low-level C++ extensions or optimized kernels. Once integrated into the solution, the malicious code can remain dormant until activated by a specific condition, allowing attackers to steal sensitive threat intelligence data, bypass security controls, or compromise the entire system.

This supply chain vulnerability is particularly concerning for sovereign applications, where trust in the entire software stack is essential.

Solutions that cannot be trusted

Large Language Models are often described as "black boxes" because their internal decision-making process is largely opaque and incomprehensible to humans. Even though we can observe the input and the output, we cannot clearly see or understand how the model arrives at a particular conclusion, which

neurons activated, which statistical patterns were used, or why one answer was chosen over another.

In the context of threat intelligence and sovereign applications, this lack of transparency represents a major risk.

Analysts and decision-makers cannot reliably verify the reasoning behind critical outputs, such as threat assessments, correlations, or alerts. This makes it extremely difficult to audit the system, ensure accountability, or provide legally defensible explanations, especially under the strict requirements of the EU AI Act. In high-stakes environments where mistakes can have serious national security or legal consequences, relying on an unexplainable black box is simply unacceptable.

Also, Large Language Models hallucinate primarily because they are not knowledge systems but sophisticated next-token predictors trained to generate the most statistically probable continuation of a text sequence.

During training, they learn to reproduce patterns from vast amounts of internet data, but they have no true understanding, no internal fact-checking mechanism, and no direct access to a verified source of truth while generating responses.

When the model encounters gaps in its statistical knowledge, ambiguous patterns, or conflicting information from its training data, it still produces fluent and coherent text by choosing the most plausible-sounding continuation, even if that continuation is factually incorrect. This probabilistic nature, combined with the autoregressive process where each new token must remain consistent with previous ones, often leads the model to confidently invent details, fabricate sources, or create entirely plausible but false information.

Additionally, the training objective rewards fluency and confidence more than strict accuracy, which further encourages convincing hallucinations rather than admitting uncertainty.

Large Language Models are also particularly vulnerable to data poisoning because they are trained on enormous quantities of uncurated text scraped from the open internet, where malicious actors can deliberately inject false, biased, or harmful content (no clear data pedigree).

Attackers can poison the training data by creating fake websites, editing public sources such as Wikipedia or forums, uploading manipulated datasets to open repositories, or even participating in crowdsourced data collection efforts. Once

this tainted information is included in the training process, the model statistically learns and internalizes the poisoned patterns, embedding them directly into its neural weights. This makes the malicious behavior extremely difficult to remove afterwards, as there is no simple "delete" function for specific knowledge in a trained model.

As a result, a poisoned LLM can reliably produce propaganda, spread disinformation, leak sensitive information, or exhibit hidden backdoors when triggered by certain inputs, all while appearing completely normal in other contexts.

Keep in mind, for all those reasons, that the same input can produce different outputs (lack of determinism and reproducibility). In threat intelligence, where auditability, forensic traceability, and legal admissibility are critical, this is a serious problem. Here, reliability and trustworthiness are critical.

Solutions that are not sustainable

A conventional server typically relies on one or two general-purpose CPUs and modest amounts of standard DDR memory. An AI-optimized server is very different.

An AI-optimized server is built around multiple high-power GPUs (often 4 to 8 or more), each consuming between 700 and 1,200 watts under load. These systems require massive high-bandwidth memory (HBM), ultra-fast interconnects such as NVLink, and advanced liquid cooling solutions to manage the extreme heat generated. As a result, a single AI server node can draw between 6 and 15 kilowatts or more during operation, compared to just 0.5 to 1.5 kilowatts for a traditional server. This architectural difference means AI infrastructure demands far more raw materials for manufacturing, including large quantities of silicon, rare earth elements, copper, gold, and specialized chemicals used in advanced semiconductor nodes and high-bandwidth memory stacks. Consequently, the production of AI servers carries a substantially higher environmental footprint in terms of resource extraction, energy-intensive fabrication, and overall material intensity than conventional IT equipment.

And the reason is obvious. Large Language Models consume enormous amounts of energy during computation because they rely on billions to trillions of mathematical operations, primarily massive matrix multiplications and attention calculations, performed repeatedly for every token generated.

On a typical AI server with powerful GPUs (such as NVIDIA H100 or B200), this energy demand is extremely high for several reasons.

First, the GPU's thousands of parallel cores run at near-maximum utilization to process these dense linear algebra operations as quickly as possible, drawing hundreds of watts per card (often 350–1000 W under load).

Second, the model's parameters and intermediate activations must be constantly loaded from high-bandwidth memory (HBM), creating heavy memory traffic that consumes significant power beyond pure computation.

Third, the entire server infrastructure, including multiple GPUs working in parallel, high-speed interconnects (NVLink), cooling fans or liquid cooling systems, CPUs, and power delivery components, adds substantial overhead, with a full 8-GPU node easily reaching several kilowatts under sustained inference or training workloads.

Unlike traditional software, LLMs have very little idle time during active use, and the autoregressive nature of generation (producing one token after another) keeps the hardware under constant heavy load. As a result, even a single inference request on a large model can consume far more electricity than conventional computing tasks, and at scale this quickly translates into massive operational energy costs and environmental impact.

Also keep in mind that Large Language Models degrade over time (concept drift), especially in fast-moving domains like cyber threats. Keeping them relevant requires frequent retraining or fine-tuning, which is extremely expensive in compute and energy, and reintroduces all the poisoning and sovereignty risks each time.

In addition, the portability of Large Language Models from CISC architectures (primarily x86 used by Intel and AMD) to RISC architectures such as RISC-V is highly complex and represents a major sustainability challenge.

These models are deeply optimized for the dominant CISC ecosystem, relying on highly tuned CUDA kernels, proprietary GPU libraries, and pre-compiled Python extensions (wheels) that are specifically engineered for x86 instruction sets and NVIDIA hardware. Moving them to RISC-V requires rebuilding the entire software stack from source, adapting or rewriting thousands of low-level mathematical operations for the RISC-V Vector extension (RVV), and re-optimizing memory access patterns, which often results in significant performance degradation.

Furthermore, the massive size of modern LLMs, combined with their heavy dependence on specialized GPU acceleration and high-bandwidth memory, makes efficient execution on current RISC-based systems extremely difficult and energy-intensive.

This lack of portability renders the technology unsustainable in the long term for Europe's sovereignty goals, as it perpetuates a strong dependency on non-European hardware vendors and ecosystems.

In a context where technological independence and resilience are priorities, relying on such architecture-specific, resource-heavy models creates ongoing risks related to supply chain vulnerabilities, high energy consumption, and limited adaptability to future open hardware standards.

In addition, one of the most significant challenges in adopting Large Language Models at enterprise scale lies in the unpredictability of costs.

Unlike traditional IT modernization projects, where expenses are largely fixed and foreseeable, LLM usage is billed on a consumption basis (pay-per-token). This creates a fundamental issue for business cases: the actual cost often deviates dramatically from initial projections. A seemingly minor increase in query volume, prompt complexity, or output length can multiply expenses by several factors within weeks. As a result, many organizations struggle to build reliable ROI calculations.

What appears economically attractive at pilot stage frequently becomes far more expensive than anticipated once deployed at scale, sometimes even exceeding the fully loaded cost of the human employees the AI was meant to replace. This cost unpredictability represents a major barrier to confident investment decisions and explains why many companies are now shifting toward hybrid approaches or specialized AI solutions with predictable, fixed-cost models for high-volume, repetitive tasks.

Solutions that are no legal to use

From a legal perspective, Large Language Model-based solutions pose significant compliance risks under the European Union AI Act, which entered into force in 2026.

Many applications in threat intelligence, cybersecurity, or decision-support systems are likely to be classified as high-risk AI systems due to their potential impact on public safety, fundamental rights, and national security.

The AI Act imposes strict requirements for transparency, explainability, risk assessment, human oversight, technical documentation, and conformity assessment. However, LLMs operate as black boxes, making it extremely difficult, and often impossible, to provide clear explanations of how a specific output or decision was reached.

This lack of transparency prevents proper auditing, accountability, and the ability to demonstrate compliance with the mandatory human oversight and risk mitigation obligations. The system frequently operates as a semi-autonomous “black box,” generating conclusions and recommendations with limited explainability. In such cases, the ultimate legal liability shifts almost entirely to the organization, as the AI itself has no legal personality. This creates significant compliance challenges under the EU AI Act, particularly for high-risk applications.

Furthermore, the stochastic nature of LLMs, combined with their propensity for hallucinations and vulnerability to data poisoning, increases the likelihood of erroneous or biased outputs that could lead to legal liability or regulatory sanctions. As a result, pure LLM-based systems are considered high-risk and poorly suited for deployment in sensitive sovereign contexts under the AI Act, as they struggle to meet the fundamental principles of trustworthiness, traceability, and explainability required by European law.

Another major concern is that LLM-based systems can be deliberately manipulated to discriminate against individuals or groups, directly violating the fundamental rights protected by the European Charter of Fundamental Rights. By injecting biased, discriminatory, or hateful content into the training data, even in very small quantities, malicious actors can poison the model so that it systematically produces unfair, stereotypical, or discriminatory outputs.

For instance, the system could unfairly associate certain nationalities, ethnic origins, religions, or social profiles with higher security risks, leading to biased threat assessments or automated decisions. Because of the black-box nature of LLMs and the difficulty in detecting such subtle manipulations after training, these discriminatory behaviours can remain hidden and persistent. This represents a serious risk of indirect discrimination and automated bias, which is explicitly prohibited under the EU AI Act and contradicts the core principles of equality, non-discrimination, and human dignity enshrined in the European Charter of Fundamental Rights. In high-stakes domains such as threat intelligence and law

enforcement, such vulnerabilities could lead to systemic injustices and legal challenges at both national and European levels.

Another important thing to be aware of is that the use of Large Language Models in critical sectors is particularly complicated under the NIS2 Directive, which entered into full application in 2025 and sets stringent cybersecurity and resilience requirements for essential and important entities across the European Union.

NIS2 mandates robust risk management, incident reporting, supply chain security, and the implementation of appropriate technical and organizational measures to ensure the confidentiality, integrity, and availability of systems and data.

LLM-based solutions struggle to comply with these obligations due to their inherent vulnerabilities, including susceptibility to data poisoning, prompt injection attacks, hallucinations, and unpredictable behavior.

Furthermore, their heavy reliance on opaque third-party components (such as foundational models, Python libraries, and CUDA-dependent GPU stacks) creates complex and difficult-to-audit supply chains that extend far beyond European control.

The lack of determinism and explainability also makes it extremely challenging to conduct proper risk assessments, demonstrate due diligence, or provide timely and accurate incident reporting when something goes wrong.

As a result, deploying LLMs in environments covered by NIS2 exposes organizations to significant compliance risks, potential regulatory fines, and increased liability, making them poorly suited for sovereign and critical infrastructure applications where cybersecurity resilience and accountability are mandatory.

In conclusion, while Large Language Models offer impressive capabilities in terms of speed and versatility, their numerous structural limitations make them poorly suited for sovereign and high-stakes applications such as threat intelligence in Europe.

Their heavy dependence on Python and CUDA, difficult portability to RISC-V architectures, massive energy consumption, lack of explainability, propensity for hallucinations, and vulnerability to data poisoning create significant technical, operational, and environmental challenges.

From a legal standpoint, these systems raise serious compliance issues under the EU AI Act and the NIS2 Directive, particularly regarding transparency, accountability, non-discrimination, and cybersecurity resilience. They also pose risks to the fundamental rights protected by the European Charter of Fundamental Rights through potential biases and discriminatory outputs.

Given Europe's strategic objective of technological sovereignty and digital autonomy, relying on such opaque, resource-intensive, and externally dependent technologies represents an unacceptable level of risk.

A more sustainable and responsible path lies in adopting transparent, deterministic, energy-efficient, and architecture-independent solutions, such as symbolic AI systems, that can be fully audited, controlled, and optimized for European standards.

First Humans, then machines

At Aeteos™, we believe technology must help, serve, and protect people first. That is why we build human-centered cognitive computing solutions to empower the heroes on the front lines who defend our freedoms and our way of life, creating a safer and more resilient Europe.

Since 2016, we build these solutions differently. Drawing from neuroscience, psychology, philosophy, and linguistics, we are inspired by how humans actually process symbolic information. This allows us to create technology that is ethical, sovereign, secure, trustworthy, and sustainable, always in service of those who protect our society, never replacing them.

We deliver Percipion™, a cognitive platform that turns sensitive textual data into actionable, human-like intelligence. It empowers law enforcement, defense, forensics, and critical organizations with clear, explainable inferences they can trust to make fast, informed decisions.

Built with transparency, security, and trustworthiness at its core, Percipion™ offers a fundamentally different approach to information processing, one that is deterministic, explainable, and fully controllable by humans.

Can we rely on a machine to help us ?

Humans have always sought to create tools, machines, or automatons to overcome or alleviate the limits of their own bodies and finite energy. The fundamental need has remained almost always the same: to reduce physical fatigue (fewer repeated muscular efforts), increase speed (to do things faster or for longer than a human naturally can), improve precision and repeatability (to avoid errors caused by fatigue or human variability), and finally to free up time for nobler, more creative, or strategic tasks.

From the Acheulean handaxe to the wheel, from the automobile to the computer, tools extend our capabilities. They allow us to delegate tasks to machines, and this delegation profoundly transforms our societies. It frees up time for some, but it can also create or exacerbate inequalities for others. It is therefore essential to ensure that this delegation serves human virtue rather than altering or replacing it.

The tool or machine is, in a sense, an extension or prosthesis of ourselves that compensates for our weaknesses. It is a millennia-old quest for delegation so that humans can devote themselves to other things: creating, thinking, exploring, or simply resting.

When humans delegate a skill to a machine, that machine must not operate without limits.

Human virtue is precisely what directs power (physical strength, intelligence, technology) toward good, rather than toward evil, excess, or indifference. Without virtue, delegation can generate massive inequalities: some gain time and wealth while others lose their jobs and existential meaning.

What distinguishes humans is not only their ability to create tools, but also their capacity to deliberate on what is good, to freely choose what is just, and to cultivate stable dispositions toward virtues. If technological delegation erodes these capacities, it ultimately attacks what makes us human.

In this context, human virtue refers to the set of moral and intellectual excellences that enable human beings to remain masters of their own technical power, rather than becoming its instrument or victim.

It is precisely because we can delegate so much that we need strong virtues to decide what to delegate, to whom, for what purpose, and with what limits. Without

these virtues, delegation risks producing a society that is more technologically powerful but less free, less just, and less human.

Ethics therefore intervenes as an essential normative framework to ensure that delegation truly serves humanity, by emphasizing transparency (understanding how the machine works), equity (avoiding discriminatory biases), autonomy (preserving human freedom), and responsibility (preventing the dilution of accountability).

Ethics as the Foundation

Ethics can be defined as the “science of morality.” It is a philosophical discipline concerned with moral judgments. It represents a fundamental reflection by every people to establish its norms, limits, and duties, and aims to determine a proper and good way of living.

Even though the term “ethics” is often used as a synonym for “morality,” this equivalence is not entirely accurate.

Morality consists of a set of “relative” rules and values, fictitiously elevated to absolute Good and Evil. It belongs to a specific individual, social group, or people at a precise moment in its history. Ethics, by contrast, is the reasoned pursuit of the good. It is a body of theoretical reflections on the value of human practices and the conditions under which they occur. We can say that ethics provides a rational framework for morality in the search for happiness for all.

Ethics is a rationally structured set of explicit values that define the good, the just, and the beautiful, through which a person accounts for themselves, for what makes them exist and act. It tells the individual how they ought to live and on what basis they should judge and decide. It is therefore an explicit and argued system of values that guide behaviors and social practices. Ethics can be compared to a code of universal principles applicable across a wide range of contexts.

Aristotle teaches us that ethics consists of a rationally structured set of explicit values defining the good, the just, and the beautiful. Acting with justice and moderation (virtue), performing noble actions based on true knowledge and guided by reason, leads to a fulfilled life. This is practical wisdom, and it is the path to true happiness.

It is therefore necessary for humans to be able to delegate activities to machines in a virtuous manner.

As Aristotle indicates, our actions must be guided by “practical wisdom” (phronesis) or prudence. The decision to delegate must also take this into account. Humans must deliberate on when and how to delegate so that the action remains ethical. It is up to them, and them alone, to decide when delegation is appropriate.

For example, using a tool to accelerate a repetitive task is virtuous if it frees up time for higher-value activities, but not if it creates dependency that erodes our capacity for judgment or exposes us to new risks.

It is ultimately the human who bears primary responsibility for the proper implementation of this delegation. It is the human who designs, builds, and decides to use the machine. They must ensure that it serves a real need, that the user remains fully free and autonomous, and that they understand its functioning sufficiently to maintain control and assume full responsibility.
Prioritize long-term resilience over hyper-innovation !

Our ancestors used to cook soup in a cast-iron pot that rested on a tripod in the fireplace or was suspended from a crane or chain above the fire. This cauldron was never washed, which caused it to develop a seasoned layer (culottage). Since soup was always cooked on top of the remains of the previous batch, the more seasoned the pot became, the more flavor the soup acquired. It is true that “the best soup is made in old pots.”

This “old-fashioned” approach also applies to the world of machines and software.

Let us take a few examples.

Horology is the art and science of measuring time. In the 13th–14th centuries in Europe, the first mechanical clocks with weights and gears appeared. A gear train counted the hours. They were imprecise but revolutionary for regulating social and monastic life. Miniaturized in the 15th century and then slipping into pockets by the 17th century, they became increasingly accurate. The first wristwatches appeared around 1810.

In 1848, 23-year-old Louis Brandt opened a family workshop in La Chaux-de-Fonds, in the Swiss Jura mountains, a few kilometers from the French border. All production was local. His sons joined the venture and in 1894 launched the

famous Omega caliber, a precise, robust, and easy-to-repair movement. Today, in 2026, these high-precision Omega watches remain largely mechanical and inherit centuries of Swiss watchmaking expertise.

The boxer engine (with opposed cylinders) was first patented in 1896 by Karl Benz. It features two opposing cylinders (flat-two) and was designed to reduce vibrations and improve balance, in contrast to the dominant inline engines of the time. This design aimed to create a smoother and more compact engine, ideal for early motorized vehicles.

Later, Ferdinand Porsche, strongly influenced by this concept, adopted it for Volkswagen in the 1930s with an air-cooled flat-four engine mounted at the rear for better balance and mechanical simplicity. This engine has a low center of gravity (improving road handling) and minimal vibrations. This flat-four became the technical foundation for the first Porsche models. The company was founded in 1948 in Gmünd, Austria, by Ferdinand and his son Ferry. It powered the very first Porsche and then evolved. With the arrival of engineer Hans Mezger, it became a flat-six in 1963 and equipped the first Porsche 911 from 1964 onward. A new engine known as the “Mezger Engine” was introduced in the 1980s and evolved into the flat-six engines that still power modern Porsche 911 GT3 RS models. The main characteristics remain unchanged: the boxer engine’s DNA has stayed simple, balanced, and high-performing.

Thus, whether for a mechanical watch or an automobile engine, this technical conservatism aims to prioritize reliability, durability, and incremental innovation rather than radical changes. Some manufacturers prioritize long-term resilience over hyper-innovation. They focus on small continuous improvements, give priority to stability and reliability rather than rapid growth, and seek above all to preserve and enhance the essentials.

When building a software, we should keep this in mind and prioritize reliability, stability and durability too.

Static and dynamical systems

Many machines or systems consist of interconnected elements which, once assembled, execute predefined functions according to internal rules, in an autonomous or semi-autonomous manner, to produce a specific result. These systems are essentially “automatons.”.

Here are a few concrete examples: the components of a mechanical watch assemble to measure time, the components of a rifle assemble to perform a ballistic function or the components of an engine assemble to produce motion.

Once assembled and put into use, these systems make physical parts interact with one another according to mechanical laws, autonomously producing a predictable functional outcome.

The principle behind a computer program or software product is identical, except that its components are immaterial. It results from the fixed assembly of abstract pieces (libraries and functions) “forged” using a programming language.

These pieces are executed by a processor to perform their function. Once launched, the program processes input data by applying programmed rules and produces an output.

When these systems or programs are assembled, tested, and adjusted to meet the expressed needs, they can be put into service.
The watch case is sealed, the engine cover is closed, the software is compiled. Everything is then under control, fixed, predictable, and immutable, with few external dependencies once built.

These systems are static and deterministic: each initial condition leads to one and only one final state. The same inputs always produce exactly the same outputs. They can be trusted because they exhibit reliable and predictable behavior.

But there are also dynamic systems, capable of variation, of transitioning from one state to another, while still performing their intended function by adapting.

In computing, a program can inherit either of these natures.

A program runs statically when the code that composes it is fixed at the time of compilation, before execution. The generated binary is self-contained (with no external dependencies) and deterministic. Once compiled, it is immutable. The result of each execution is perfectly predictable.

A program runs dynamically when a significant portion of the code (the software’s source code and the libraries it uses) is read and interpreted at runtime. No final binary is generated in advance. Its execution is therefore less deterministic, mutable, and able to adapt to the context.

Guided by a form of prudence that must prevail and having in mind that people who are using the system should remain responsible for it's usage. The more deterministic the system will be, the better.

Construction materials

Another important aspect worth mentioning is the "raw materials" used to build the system. The choice of these materials strongly influences the quality, robustness, and longevity of the final product.

Here are a few examples.

An engine block can be made of low-grade cast aluminum (3xx series alloy with few additives). It deforms easily at high temperatures and can crack under high pressure. It is used in some low-cost engines or prototypes. A temperature overrun of just 20 °C can render it unusable.
The same engine can be made of traditional gray cast iron: much more resistant to heat and mechanical stress than low-grade aluminum, but heavy, prone to rust, and subject to fatigue cracking after 150,000–200,000 km.

Finally, it can be made of Compacted Graphite Iron (CGI) or high-performance forged aluminum with heat treatment. Its resistance to heat, vibration, and pressure then becomes exceptional. It easily handles over 300,000 km without noticeable deformation, while being both lightweight and ultra-rigid. This engine remains performant even under extreme racing conditions or prolonged overheating.

The same logic applies to a precision mechanical watch. If its case and movement are made of low-grade plastic/polymer with ordinary steel springs (as in "fashion" watches or toys), it will be very fragile: easily scratched or broken by minor shocks, rapid loss of precision, and sensitivity to humidity, temperature, and magnetic fields. Its real lifespan will be only 1 to 3 years.

With a stainless steel case, a standard automatic movement, and a classic alloy balance wheel, it will offer good resistance to everyday shocks and water (up to 100 m), acceptable precision (±10–15 s/day), and gradual wear. Its lifespan will reach 10 to 20 years with regular maintenance.

Finally, with a grade 5 titanium or high-density ceramic case, and a high-end horological movement featuring a silicon balance spring and balance wheel, its

resistance to shocks, scratches, corrosion, and magnetic fields becomes exceptional. Its lifespan can exceed 50 years with no significant loss of performance.

The choice of the “base material” (alloy, watch component, or programming language) ultimately determines the final strength of the product and its level of sovereignty.

The same will apply to a software. Selecting the programming language which will be used to create it as well as the operating system on which it can be executed has to be done with great care.

Building Percipion™

At Aeteos™, we have deliberately chosen the C programming language to develop Percipion™, our sovereign symbolic cognitive computing platform dedicated to threat intelligence.

This choice is explained by the intrinsic advantages of C in a context where reliability, durability, and sovereignty are paramount.

It is essential for a critical system handling sensitive data, especially under the European AI Act, that controls and audits of the software’s operation be possible. The C language, with a limited number of instructions and great simplicity, offers total control over every line of code and every function.

Well-written C code can be compiled and is operational for decades (often with C89 or C99 standards). In cybersecurity, where systems must run without interruption for 10, 20, or 30 years, this temporal stability is irreplaceable.

Using C, the final program contains all the necessary code in the form of a self-contained binary with no external dependencies. If a change is needed, one must access the source code, modify it, and recompile. This is both a drawback (less flexibility) and a major security advantage: an attacker who gains access to the system cannot easily alter the program’s behavior. The program cannot dynamically import or execute additional code. Its attack surface is minimal not to say inexistent.

Portability is another major advantage: C programs compile and run with minimal modifications on a wide range of platforms, without depending on frameworks or

third-party libraries that may be vulnerable or under foreign control. This guarantees complete independence from suppliers and the ability to deploy Percipion™ in constrained or air-gapped environments.

Finally, the stability of the language itself, which has evolved very slowly since 1972 through conservative standards, ensures “mechanical” predictability and fully deterministic behavior.

In summary, C is not chosen out of nostalgia or to be “vintage,” but out of conservative pragmatism: it maximizes control, minimizes hidden risks, and guarantees uncompromising longevity, exactly what a sovereign platform demands. But that is not all.

A 2017 university study conducted by Portuguese researchers (University of Minho) and presented in Canada compared 27 programming languages executing 10 identical simple algorithms, measuring their performance in terms of execution time, energy consumption, and memory usage.

Regarding energy consumption: a program written in C consumes the least energy. By comparison, the same program written in Python consumed 75.88 times more energy.

Regarding execution time: a program written in C is the fastest. The Python version was 71.90 times slower.

Regarding memory consumption: a C program used about 17% more memory than the same program in Pascal. The Python program used 280% more memory.

The authors then combined these factors according to different constraint scenarios: For the fastest solution that also uses the least memory: C, Pascal, or Go. For the fastest that consumes the least energy: C. For the one that consumes the least memory and least energy: C or Pascal. For the fastest that also consumes the least memory and least energy: C, Pascal, or Go.

In all possible configurations, the C language is always one of the recommended languages, if not the only one.

In its standard ANSI C99 version, this language additionally offers:

- Absolute control over the machine, with no abstraction hiding anything. You see exactly what happens byte by byte, cycle by cycle.

- No imposed “runtime”: the code you write is (essentially) the only code that executes.

- Great stability: since the C99 standard, evolutions have been optional and conservative.

- Compilers (GCC, Clang) that produce highly optimized machine code, with optimizations that have been stable for over 20 years.

In summary, the C language created in 1972 is not “old”, it is mature and complete. It has never been replaced by any other language and has only been improved when truly useful.
When the priorities are total mastery, longevity, portability, raw performance, and independence from any third party, the C language remains in 2026 one of the best possible choices.

Therefore, in our context, “doing it the old-fashioned way” has its advantages and makes sense. It is a solid foundation for achieving total sovereignty, and it is the language we have chosen at Aeteos™ to build Percipion™ from scratch.

In computing, two other choices are essential: the hardware and the operating system on which the program will run.

Today, the vast majority of servers worldwide run on CISC architectures, primarily x86 processors from Intel and AMD. These architectures have dominated the data center and cloud landscape for decades, with mature toolchains, optimized libraries, and extensive hardware support.

For reasons of technological sovereignty, Europe has decided to accelerate its transition toward RISC architectures, and particularly the open and royalty-free RISC-V instruction set. This strategic shift aims to reduce critical dependencies on non-European suppliers. As a result, in the coming years, an increasing number of servers, edge devices, and high-performance computing systems in Europe will be based on RISC-V or other RISC designs.

Future-proof applications, especially in sensitive domains such as threat intelligence, cybersecurity, and AI, will therefore need to run efficiently and natively on both of these architectures.

This is why the choice of programming language made today is crucial. C is significantly better positioned than Python for this transition.

Adapting a Python-based program to RISC-V is much harder for several fundamental reasons. Python is an interpreted language that relies on the Python interpreter and a large ecosystem of pre-compiled binary extensions (wheels). Most core libraries used in AI and data processing, such as NumPy, SciPy, PyTorch, or Transformers, contain performance-critical parts written in C or C++ that are heavily optimized for x86 (and to a lesser extent ARM). Porting these to RISC-V requires rebuilding the entire dependency chain from source, adapting or rewriting vectorized code for RISC-V Vector extensions (RVV), and often losing the highly tuned performance gains that exist on x86. Many wheels are simply not available yet for RISC-V, forcing slow compilation steps and frequent compatibility issues. Additionally, Python's Global Interpreter Lock, dynamic typing, and high-level abstractions make it difficult to achieve low-level control and optimal performance on a new ISA without deep modifications to the runtime itself.

In contrast, a program written in C is compiled directly to machine code. It offers fine-grained control over memory, registers, and instructions, making it far easier to port, optimize, and generate efficient code for any new architecture like RISC-V using standard compilers such as GCC. This results in better performance, smaller binaries, lower resource consumption, and much greater long-term portability, exactly what is needed for a sovereign European stack. Choosing C today therefore provides a decisive advantage in terms of resilience, performance, and independence in the face of the architectural shift ahead.

Regarding the operating system, the user should have the freedom to select the operating system that he would like to use and Percipion™ should run on it. That said, we should prioritize sovereignty and in 2026, the only operating system that truly enables technological sovereignty is Linux and the first thing to do when preparing a server is to install it. This installation places numerous software components on the machine that are necessary for it to function. Every installed piece of software or component can then be executed. If it contains no vulnerabilities, all is well. Otherwise, any malfunction or vulnerability in that software can provide an attacker with a potential entry point into the system, toward business applications or sensitive data.

Debian stands out as the strongest choice for European sovereignty in threat intelligence computing because it is the only major Linux distribution that operates as a truly independent, community-driven project with no single corporate owner or controlling shareholder. For us at Aeteos™, it was the best possible choice.

In a high-risk domain governed by the AI Act, where systems must be fully auditable, free of hidden dependencies, and resistant to data exfiltration, Debian's

minimal-install philosophy, open governance, and absence of vendor lock-in deliver unmatched control. You can build every package from source, run air-gapped deployments, and avoid the “sovereignty-washing” trap seen with Mistral or SUSE, where a French or European facade still hides heavy reliance on US-controlled frameworks like PyTorch, NVIDIA CUDA, or a private-equity-owned company that could be sold tomorrow.

Its main long-term support comes from European entities (Freexian in France, Infomaniak in Switzerland, Proxmox in Austria), its user base and contributors are overwhelmingly European, and it has already been chosen for sovereign French government systems.
For threat intelligence workloads that demand transparency, reproducibility, and zero hidden US backdoors, Debian gives you the closest thing to genuine digital sovereignty: a rock-solid, auditable foundation that you truly own, rather than rent from a marketing narrative.

Getting to where we want, it is essential to install the absolute minimum number of applications and packages in order to reduce the attack surface as much as possible. This is precisely the goal of a minimal Debian installation, which aims to include only the packages strictly necessary for the server to boot and run at a basic level, typically around 60 to 90 essential packages depending on the configuration.

A package is an archived file that bundles everything needed to install and run a piece of software or a software component in a clean and controlled manner. This compressed archive has a precise structure and usually contains:

- Binary files (pre-compiled executable programs);
- Shared libraries required by the software;
- Default configuration files;
- Documentation;
- Scripts that run automatically during installation;
- A metadata file describing the package, its version, architecture, and dependencies.

If needed, this minimal installation can later be supplemented by adding only the truly useful missing packages.

This approach drastically reduces the number of binaries, libraries, and services that could potentially be vulnerable on the server. Less code means fewer risks of exploitable flaws, whether remotely or locally. The attack surface is thus significantly reduced.

Once this base system is installed and hardened, additional software solutions can be installed in a controlled manner.

We now have all we need to start building a software !

Let us do Cognitive Computing

At the heart of Percipion™ lies its Cognitive Computing Engine, a sovereign and deterministic approach to information processing that fundamentally differs from statistical or probabilistic AI systems.

The engine works by first encoding information through clean and simplified language, transforming raw data into structured, high-quality input. It then leverages VirtualBrains™ (knowledge schema), a powerful categorized symbolic object structure analog to the human long-term memory and based on hyponym-superordinate relationships, enabling true symbolic representation and understanding of natural language. This foundation allows Percipion™ to genuinely understand text, perform deductive reasoning, compare documents with precision, and search either by exact words or by semantic meaning. The system can memorize new concepts by processing semantic noise to detect emerging words and expressions, while offering rich communication capabilities through multiple user-ready reports and instant messaging. Finally, it can act autonomously by launching scripts or external applications based on reasoned outcomes.

This integrated cognitive cycle (Represent, Understand, Reason and Communicate) delivers transparent, explainable, and fully controllable intelligence that organizations can trust for critical decision-making.

Percipion™ is a software product entirely written in the C programming language (ANSI C99 standard). It is compiled statically and installs on the server “as is,” without depending on any third-party software products or external libraries.

The C header files and system libraries compliant with the ISO C99 standard used by GCC during the compilation of Percipion’s™ services are those provided natively by the Debian operating system (7 standard libraries in total).

Version M.26 of Percipion™ (latest version as of April 2026) consists of 29 distinct software services and 26 knowledge schemas. It was developed using

exclusively these system libraries. This version contains 99,259 lines of source code. The code is available for audit upon request.

The only company involved in the development and maintenance of Percipion™ is Aeteos™. Everything was coded in France, in Picardy, in the Somme department. From 2016 to 2026, Aeteos™ has remained 100% independent, with no fundraising and no external dependencies.

Orchestrator

Complementing the Cognitive Computing Engine is Percipion's™ powerful real-time orchestrator, a lightweight software daemon designed to seamlessly connect multiple software services and create sophisticated business scenarios.

Operating continuously in real time, the orchestrator enables organizations to define and automate complex assessment scenarios tailored to their specific needs.

Whether linking threat intelligence feeds (documents, websites, social networks, instant messaging, mailboxes, ...) or internal business applications, it provides the flexibility to build custom workflows that transform raw data into business reports and actionable decisions.

Fully aligned with Percipion's™ principles of sovereignty, security, and trustworthiness, the orchestrator remains transparent, auditable, and free from third-party dependencies, ensuring that automated business processes stay under complete human control while delivering speed, reliability, and operational efficiency.

Advanced textual analysis

Percipion™ provides powerful advanced analysis capabilities for textual content, going far beyond traditional keyword or sentiment detection.

The Cognitive Computing Engine performs deep psychological assessment by identifying affective states, primary and secondary emotions, human needs, psycho-social risks, and motivational drivers. It analyzes semantic and

psychological valences, takes into account grammatical context, and accurately transcribes Emojis and Algospeak into clear, usable language.

This advanced analysis relies on a neuroscience-inspired component called the SmartNeuron™. When Percipion™ processes textual information, it will evaluate whether a word in the incoming data is semantically close to concepts stored in one available knowledge schemas. This SmartNeuron™ perceptual matching layer emulates the brain's natural tolerance to imperfect input. It allows machines to correctly interpret and link misspelled words, typos, informal variants, or abbreviated forms, exactly as a human reader would, without any prior training or machine-learning retraining. The result: robust, real-world performance on noisy, real-life textual data (forensic extractions, social media, internal communications, etc.) while remaining fully explainable and sovereign.

The system detects semantic signals, including strong and weak signals as well as words carrying multiple or ambiguous meanings, delivering nuanced understanding of intent and underlying messages.

In addition, Percipion™ is capable of evaluating a wide range of sensitive and strategic knowledge areas, including disinformation, conspiracy theories, corruption, conflicts of interest, workplace satisfaction, soft skills, client experience, hate speech, scamming, bullying, human trafficking, self-harm, sexual offenses, child grooming, false documents, illegal drugs, as well as specific high-risk domains such as violence and ultra-violence, weapons and ammunition trafficking, explosives, and general content moderation.

Threat Intelligence

Threat intelligence is the systematic process of collecting, analyzing, and transforming raw data about potential and emerging cyber threats, including threat actors' motives, tactics, techniques, and procedures, into actionable, contextual insights that enable organizations to anticipate, prevent, and respond effectively to attacks.

Rather than reacting to incidents after they occur, it empowers proactive defense by providing the "who, why, and how" behind evolving risks.

Percipion™ supports this critical need through its sovereign, minimalist, and fully transparent architecture combined with its advanced Cognitive Computing Engine.

By operating with zero dependencies, static compilation, and a deterministic knowledge-based approach, it delivers explainable and auditable analysis of qualified data without any black-box mechanisms, learning, or data memorization. This ensures that threat intelligence remains accurate, ethical, and protected against poisoning or alteration, while its low attack surface, quantum-resistant encryption, and total resilience allow organizations to maintain continuous, trustworthy situational awareness, even in offline or high-security environments.

Percipion™ is particularly suited for critical organizations and high-stakes environments, where it delivers tangible value across a wide range of strategic and operational use cases. Some examples :

Disinformation and Cognitive Warfare : Percipion™ detects weak semantic signals, psychological manipulation tactics, disinformation narratives and conspiracy theories in real time on social media and messaging platforms. It assesses whether content is harmful, triggers early warnings and supports swift reasoned counter-responses, a vital pillar of cognitive resilience and democratic defense in Europe.

Proactive Threat Detection : Leveraging advanced weak-signal detection, psychological profiling and contextual knowledge schemas, Percipion™ anticipates emerging threats, grooming, scams, bullying, radicalization, hate speech, terrorism planning and hybrid attacks, before they materialize. It tracks subtle shifts in semantics, emotion and intent across public and internal sources, delivering early warnings and powering total resilience strategies.

Insider Threats : Percipion™ identifies grooming, coercion, psychological manipulation, and gradual disloyalty through deep analysis of internal communications. It detects rising negativity, emotional drift, and suspicious behavioral patterns, allowing security teams to intervene early and prevent data leaks, sabotage, or espionage from within.

Industrial Espionage & IP Theft : Percipion™ monitors internal-supplier communications for signs of data exfiltration, IP theft, and foreign espionage. It uncovers subtle linguistic indicators and psychological coercion patterns, while its advanced document comparison capabilities help protect Europe's strategic technologies and industrial sovereignty.

Forensic Analysis : In post-incident or judicial contexts, Percipion rapidly processes content of seized devices and communications to reveal evidence of corruption, bribery, fraud, ransomware negotiations, or crime. Its explainable

symbolic analysis and evidence report delivers court-admissible, traceable inferences in hours instead of weeks.

Talent Attrition & Workforce Morale : Percipion™ detects early signs of disengagement, frustration, or external recruitment pressure in employee communications. In critical sectors (defense, aerospace, energy), early identification of attrition risks protects sensitive knowledge, reduces insider threats, and maintains operational continuity.

Psychosocial Risks Assessment : Percipion™ analyses internal communications to identify burnout, bore-out, brown-out, and psychosocial risks. By tracking emotional valence and psychological drift at scale, it helps organizations prevent mental health crises, reduce absenteeism, and strengthen workforce resilience, a key component of total organizational resilience.

Supply Chain Risk Monitoring : Percipion™ monitors supplier and partner communications for early signs of compromise, coercion, bribery, or sabotage intent. It detects psychological pressure, anomalous phrasing, and weak signals of supply-chain attacks, protecting Europe's critical infrastructure from hidden vulnerabilities.

E-reputation Analysis & Crisis Monitoring : Percipion™ continuously tracks brand perception, competitor narratives, and emerging crises across social media and news. It identifies coordinated attacks, disinformation targeting reputation, and sentiment shifts, enabling rapid crisis response and strategic communication decisions.

Behavioral Assessment & Psychological Matrix : Percipion™ builds dynamic psychological profiles based on affective states, primary/secondary emotions, and human needs. It delivers deep behavioral inferences for threat assessment, radicalization detection, grooming identification, and personnel risk evaluation.

Content Moderation : Percipion™ provides precise, context-aware moderation of internal and external content, detecting hate speech, violent extremism, grooming, and other inappropriate content with high accuracy and full explainability, essential for platforms, institutions, and organizations operating in regulated environments.

By supporting this comprehensive spectrum of scenarios through its sovereign, secure, and explainable architecture, Percipion™ empowers organizations to anticipate risks, protect sensitive assets, and maintain operational resilience in an increasingly complex threat landscape.

Beyond threat intelligence, Percipion™ also supports:

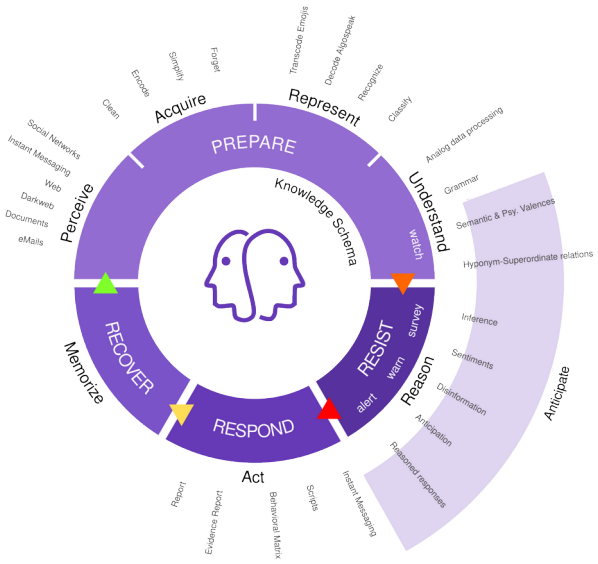
HR and talent teams by analyzing employee communications and feedback to detect low morale, uncover root causes of attrition, identify psychosocial risks, and evaluate motivation, job satisfaction, and transformation initiatives, helping organizations proactively safeguard workforce well-being and retention.

Marketing & sales, protecting brand reputation, uncovers purchase motivations and behavioral profiles, analyzes campaigns and competition in real time, and identifies ambassadors vs. critics, helping organizations prevent client loss, outmaneuver competitors, and drive more targeted, resilient growth.

Customer experience and support, analyzing interactions in real time to identify dissatisfied customers early, monitor complaints, evaluate overall satisfaction, optimize ticket routing, and decode tone/emotions, helping teams reduce churn, boost resolution speed, and deliver more empathetic, proactive service.

Resilient

Percipion™ delivers total resilience by acquiring textual content, representing it within its internal memory, searching for related engrams across contextually connected knowledge schemas, and linking those that are relevant to their parent symbolic object. This process enables the system to understand situations deeply, better evaluate threats, and act with anticipation.



The resilience capability is structured around four key stages.

In the Prepare phase, Percipion™ continuously monitors the environment, identifies emerging patterns, and flags potential threat actors.

During the Resist phase, it evaluates threat levels and provides appropriate responses to survey, warn, or alert.

In the Respond phase, the system rapidly extracts relevant evidences to support and accelerate human intervention.

Finally, in the Recover phase, Percipion™ assesses relevant contextual semantic noise to uncover overlooked patterns and build long-term resilience through bias-free processing.

This comprehensive Prepare-Resist-Respond-Recover cycle makes Percipion™ a truly dependable foundation for critical operations in uncertain and high-threat environments.

Sovereign

Percipion™ is designed from the ground up to deliver complete technological and operational sovereignty.

Built exclusively in Pure C (ANSI C99) and running natively on GNU Linux with Debian, the solution supports both RISC and CISC architectures while maintaining zero dependencies on any libraries or third-party components.

It can be deployed on-premise or in the cloud and requires no GPU, enabling efficient operation on standard, low-cost hardware.

This minimalist, self-contained architecture ensures full control: every line of code is auditable, execution is fully transparent, and there are no black-box mechanisms.

Entirely coded in France by Aeteos™, Percipion™ maintains complete independence from any supplier or third-party provider, guaranteeing that organizations retain full ownership and mastery over their data-processing environment without external influence or hidden dependencies.

Secured

Percipion™ is engineered with security at its core, delivering an exceptionally small attack surface by relying on a limited, minimalist software stack with no dependencies on external libraries or third-party components.

Fully statically compiled in Pure C, the solution eliminates risks of source poisoning and drastically reduces the number of potential vulnerabilities compared to traditional systems composed of thousands of components.

On the exfiltration front, Percipion™ ensures strict data protection through a no-learning, no-reuse architecture that prevents any form of model or data memorization.

It features no memory leaks or unauthorized copies, incorporates quantum-resistant encryption for both stored data and data in transit between hosts, and guarantees no IP loss by never copying or storing user data. With no persistent storage of sensitive information and minimal exposure to external networks for updates and upgrades, Percipion™ can operate fully offline when required, providing organizations with robust, future-proof protection against both current and emerging cyber threats.

Trustworthy

Percipion™ is designed to deliver the highest level of trustworthiness, ensuring full compliance with the EU AI Act and the NIS 2 directive. The solution is inherently reliable: it performs no prediction and no recommendation, remains non-autonomous, and operates under permanent human control.

Every process is fully explainable, deterministic, and protected against poisoning, whether at the binary or data level. As a knowledge-based system, Percipion™ relies exclusively on accurate, qualified data with a verifiable pedigree, guaranteeing ethical outcomes that are free from any discriminatory bias.

It uses an encrypted knowledge schema that cannot be poisoned or altered, providing organizations with a transparent, auditable, and responsible foundation for critical operations.

Sustainable

Percipion™ embodies a strong commitment to sustainability through its exceptionally low resource consumption. Designed with a minimalist architecture, the solution requires very low energy (2.37Wh per CPU Core) while running and maintains minimal memory usage thanks to its drastically reduced number of software components and memory optimization.

As an example, the difference in energy consumption between Percipion™ and large language models becomes striking when performing emotion analysis on short texts such as tweets (approximately 159 characters, or around 40 tokens).

Percipion™, a single-threaded symbolic AI system, processes one tweet in about 20 milliseconds while consuming 2.37 watts per active CPU core. This results in an extremely low energy usage of approximately 0.00001317 Wh per tweet (calculated as 2.37 W × 0.02 s = 0.0474 J, then 0.0474 / 3600 = 0.00001317 Wh).

In contrast, even the most efficient large language models require significantly more energy for the same task. A realistic request includes not only the tweet (40 tokens) but also a detailed fixed prompt explaining Robert Plutchik's model (8 primary emotions and 24 dyads), analysis rules, and the desired output format (1,200 tokens), plus the generated response (around 400 tokens). This brings the total to approximately 1,640 tokens per query.

While raw estimates for LLM inference range from 50–150 Wh per million tokens for optimized "Fast" models and up to 200–800 Wh per million tokens for larger models (such as GPT-4o or Claude Sonnet 4.6), modern serving optimizations such as quantization, continuous batching, speculative decoding, prompt caching, and hardware-aware tuning are substantially reducing actual consumption for short, high-volume queries. In practice, this brings the effective energy usage down to roughly 0.02–0.05 Wh per such request (on small models), which is still 1,500 to 3,800 times higher than Percipion™.

This massive efficiency gap underscores the strong advantage of symbolic AI systems like Percipion™ for large-scale, repetitive text analysis tasks where energy consumption, operational costs, and environmental impact are key considerations.

This efficient design not only lowers operational costs and environmental impact but also contributes to superior stability. By avoiding complex dependencies and unnecessary layers, Percipion™ ensures outstanding technological longevity,

allowing organizations to deploy a future-proof solution that remains reliable and efficient over the long term without frequent upgrades or hardware refreshes.

Legal

Aeteos™ provided for Percipion™ a “Declaration of Conformity” which is fully relevant and accurately reflects the EU AI Act’s core provisions (Regulation (EU) 2024/1689), including the precise definition of an AI system in Article 3(1), the exclusion of non-adaptive knowledge-based systems, the prohibitions on biometric emotion recognition and discriminatory categorization in Annex II, the high-risk obligations for law-enforcement tools under Annex III point 6, and the narrowly permitted uses for substantial public interest in criminal investigations as outlined in Articles 5 and 6.

As a threat-intelligence solution expressly positioned for high-risk use cases under the EU AI Act, Percipion™ respects the regulation in full because it operates as a purely symbolic, knowledge-based cognitive computing platform that performs deterministic semantic analysis of provided natural-language text against fixed, high-quality internal reference databases and semantic graphs, without any machine-learning components, training data, adaptability after deployment, predictive inference, content generation, recommendations, or autonomous decision-making.

This design ensures it falls outside the statutory definition of an AI system while simultaneously satisfying every high-risk requirement, human oversight remains absolute, results are transparent and fully explainable, accuracy and robustness derive from validated domain-specific reference frames that cannot be altered by third parties or external inputs, cybersecurity is inherent through offline operation and the absence of exploitable models or datasets, and no biometric data, emotion inference from physiological signals, individual profiling, social scoring, or discriminatory categorization of persons (by gender, origin, political orientation, or any other protected characteristic) ever occurs.

When deployed in the strictly delimited law-enforcement contexts permitted by the Act, such as forensic text analysis to locate missing children, detect terrorist threats, identify victims of the 32 catalogued serious criminal offences under Council Framework Decision 2002/584/JHA, or support the detection and prosecution of those offences, Percipion™ processes only lawfully obtained content under human supervision, serving overriding public-interest objectives without infringing fundamental rights protected by the EU Charter, thereby

delivering compliant, non-discriminatory, and societally beneficial threat intelligence that authorised users can interpret and act upon entirely within the legal framework.

Benefits

Percipion™ delivers immediate and tangible value to organizations by providing instant operational capability with no learning phase: simply install and start assessing critical situations right away.

It brings high-level expertise directly to environments and teams that lack specialized analysts or dedicated threat intelligence resources.

The solution operates on a transparent flat-rate model with a monthly fixed price per server that includes all features, unlimited usage (files, users or operations), updates, and support, eliminating hidden costs and unpredictable expenses.

Furthermore, Percipion™ is deployed as a fully secured platform that can operate standalone or within a collaborative environment such as NextCloud, giving organizations the flexibility to choose the deployment model that best matches their security and teamwork requirements.

These benefits combine to make Percipion™ a fast-to-deploy, cost-predictable, and highly sovereign solution for critical organizations seeking trustworthy cognitive intelligence.

Towards a Sovereign and Human-Centered Future

In an era marked by escalating cyber threats, disinformation campaigns, and sophisticated information warfare, organizations in defense, law enforcement, and critical sectors can no longer afford to rely on opaque, resource-intensive, and externally dependent technologies. Large Language Models, while powerful in scale, introduce unacceptable risks to European technological sovereignty, data security, explainability, legal compliance under the EU AI Act and NIS2 Directive, and long-term sustainability.

The dominant technical stack, built on Python, PyTorch, CUDA, and foreign hardware ecosystems, creates hidden dependencies that undermine autonomy,

amplify attack surfaces, and conflict with the principles of transparency, human oversight, and non-discrimination that European law demands.

At Aeteos™, we chose a fundamentally different path. Since 2016, guided by neuroscience, psychology, philosophy, and linguistics, we have built Percipion™ as a sovereign symbolic cognitive platform that faithfully reproduces human mechanisms of information processing.

Developed entirely in pure ANSI C99, statically compiled on a minimal Debian foundation, and free from any third-party dependencies, Percipion™ delivers deterministic, explainable, and auditable intelligence without hallucinations, data memorization, or poisoning risks.

With its extremely low energy footprint (2.37 Wh per CPU core), full offline capability, quantum-resistant encryption, and native support for both CISC and RISC-V architectures, Percipion™ offers not only immediate operational value but also true technological independence and environmental responsibility. It empowers human decision-makers rather than replacing them, turning sensitive textual data into clear, trustworthy insights for threat intelligence, cognitive warfare defense, insider risk detection, forensic analysis, and organizational resilience.

Percipion™ proves that reliable, ethical cognitive computing is possible when we prioritize humans first: control over speed, transparency over black-box fluency, and European sovereignty over convenience.

The time has come to move beyond fragile probabilistic systems and reclaim control over the technologies that protect our freedoms and way of life.

If your organization operates in high-stakes environments where trust, auditability, and autonomy are non-negotiable, Percipion™ provides the sovereign, secure, and sustainable alternative you need.

We invite you to discover how Percipion™ can strengthen your resilience against cyber, cognitive, and informational threats while ensuring full compliance with European regulations.

Contact us directly to learn more, request a personalized demonstration, or explore a proof-of-concept deployment tailored to your needs. Our team is ready to support you with a human-centered approach that respects your security and privacy.

First Humans, then Machines. Together, let us build a safer and more resilient Europe, one where technology truly serves and protects those who defend our shared values.