ChatGPT

QFaaS Triple-Transformer Systems in Sovereign AI Computing

The rise of **sovereign AI computing** – where individuals and organizations run advanced AI entirely on local infrastructure – has spurred new system designs that integrate classical and quantum elements. A prominent example is **Quantum Function-as-a-Service (QFaaS)** architectures built around a *Triple-Transformer Loop* pipeline. This deep research report explores the design and implications of **QFaaS Triple-Transformer systems**, focusing on ten key areas: (1) the Triple-Transformer Loop architecture (Analysis–Mapping–Validation) and its hardware decomposition (CPU–GPU–NVMe) on a local AI workstation; (2) low-level optimizations (CPU core pinning, GPU scheduling, NVMe tuning, container orchestration) for performance; (3) agentic software frameworks (Google **Antigravity** IDE, Conductor planning protocol, **ATHENA** agent hierarchies) and local vs cloud trade-offs; (4) multi-agent collaboration via the **Model Context Protocol (MCP)**, including context sharing, latency, and safety; (5) hybrid quantum-classical systems, e.g. Quantum Virtual Machines (QVMs), **AlphaQubit-2**, and reinforcement learning methods like **VQR-DQN** for resource orchestration; (6) comparisons of open-source vs proprietary large language models (LLMs), such as DeepSeek, including FP8 training, Mixture-of-Experts (MoE) architectures, safety alignment and censorship leakage, and context window lengths; (7) emerging use cases in the creator economy and enterprise (AI-driven DevOps, compliance automation, video editing, research assistants like NotebookLM); (8) macro infrastructure trends (data center capacity constraints, power demand growth, on-premise vs hybrid deployments, tech firm energy investments); (9) foundational physics relevant to QVMs (structured light with OAM and toroidal vortices, computer-generated holography with Fresnel zone plates, quantum mechanics fundamentals); and (10) open challenges and future directions for scaling sovereign AI (reproducibility, trustworthiness, regulation, hardware limits, cultural implications, and economic models).

Throughout, we cite recent peer-reviewed studies, technical documentation, and industry reports (2022–2025) to provide evidence for claims. We also include **claim–evidence tables** and structured subsections for clarity. The goal is to present a comprehensive, academic-style synthesis of the architecture, optimizations, multi-agent software paradigms, quantum integration, model landscape, applications, and socio-technical impacts of QFaaS Triple-Transformer systems. The **academic tone** is maintained with precise technical detail, acknowledgement of limitations, and references to foundational concepts. We conclude by summarizing key findings and outlining future research directions for sovereign AI systems that are performant, trustworthy, and aligned with human values.

1. Triple-Transformer Loop Architecture on Local AI Workstations

Triple-Transformer Loop refers to a three-stage AI processing pipeline comprising **Analysis**, **Mapping**, and **Validation** transformers. In a QFaaS context, these stages correspond to distinct functional roles that together enable complex tasks to be autonomously analyzed, executed, and verified on a local AI

workstation. Each stage is associated with a different hardware component – **CPU, GPU, and NVMe storage** respectively – effectively acting as "Transformer 1–3" in the pipeline.

- **Analysis (Transformer 1, CPU):** In this initial stage, the system *analyzes* an input problem or request using classical computation. The CPU (central processing unit) handles control logic, branching, and serial tasks such as parsing instructions, preliminary planning, and heuristic reasoning. As the most general processor, the CPU excels at orchestrating the overall workflow. It may run a smaller transformer model for reasoning about the task or use symbolic logic to outline a solution approach. This stage outputs a high-level plan or decomposition of the task for the next stage. For example, in an agentic coding scenario the Analysis transformer might interpret a user's goal (e.g. "build a web scraper") and break it into sub-tasks or specifications [1] [2]. The CPU's strengths in branching and system calls are leveraged here to prepare the problem for efficient mapping onto accelerators.

- **Mapping (Transformer 2, GPU):** The second stage *maps* sub-tasks or computations to be executed, typically on a **GPU (graphics processing unit)**. The GPU is adept at parallel numerical computation and is used for heavy model inference or training tasks. In this context, the "Mapping Transformer" could be a large neural network (e.g. an LLM or vision model) running on the GPU to generate results for each sub-task identified in Analysis. For instance, if Analysis determined multiple function implementations needed, the Mapping stage might run a code-generation model for each function on the GPU. The GPU's thousands of cores perform matrix and tensor operations for the transformer's attention mechanisms in parallel. The mapping from sub-problems to GPU kernels is orchestrated by the CPU, but once launched, the GPU handles the bulk of computation. This division aligns with real system design: high-performance serverless frameworks often offload compute-intensive parts to GPUs for efficiency [3] [4]. In a QFaaS, the GPU might also simulate quantum circuits or run hybrid algorithms during this stage (discussed further in §5).

- **Validation (Transformer 3, NVMe Storage):** The final stage *validates* and integrates outputs, often involving large data retrieval or storage – tasks suited to high-speed NVMe SSD storage. This "Validation Transformer" is not a neural network per se, but rather the **I/O and data layer** that checks results, fetches any additional context, and writes outputs. NVMe (Non-Volatile Memory express) drives provide ~GB/s throughput and low latency, enabling rapid reading of reference data or prior context (e.g. vector databases, logs) and writing of results. In the Triple-Transformer loop, NVMe might be used to store intermediate results from the GPU and then feed them back into an Analysis process for verification. For example, after the GPU generates candidate solutions, the CPU might ask the NVMe-backed database or cache to *validate* correctness (such as running test cases or retrieving known good outputs for comparison). This stage closes the loop by returning either a verified result or feedback for re-analysis. By treating the storage subsystem as a third "transformer," the architecture highlights the active role of data subsystems in AI workflows (caching, filtering, merging results). Modern NVMe drives are so fast that storage can effectively act as another processing layer for certain tasks, like k-nearest-neighbor lookups for memory or batching large datasets for the GPU. Techniques like key–value (KV) storage on NVMe can further accelerate retrieval by bypassing file system overhead [5]. In fact, the NVMe 2.0 standard introduced a KV command set where data is accessed directly via keys, avoiding costly block-index translation [6]. Such NVMe-KV features allow *Validation* to query large knowledge bases at near-memory speeds, supporting real-time fact-checking or constraint validation in the workflow.

Hardware Decomposition and Local QFaaS: In a sovereign AI workstation, these three "transformers" – CPU (Transformer 1), GPU (Transformer 2), and NVMe storage (Transformer 3) – operate in concert to implement QFaaS entirely on-premise. This contrasts with cloud FaaS where the service might invoke distributed functions; here the analogous "functions" are the pipeline stages on local hardware. The QFaaS research by Nguyen et al. (2024) emphasized vendor-agnostic integration of classical and quantum tasks via a serverless model [3]. A local QFaaS Triple-Transformer system follows a similar philosophy: *each stage is loosely coupled*, communicating via data files or shared memory, which mimics stateless functions passing outputs to the next stage.

<table> <tr><th>Transformer Stage</th><th>Role in QFaaS Pipeline</th><th>Hardware & Function</th><th>Example Operations</th></tr> <tr><td>Analysis
(Transformer 1)</td> <td>Interpret goal, plan tasks, orchestrate workflow (logic & control)</td> <td>CPU (general-purpose cores)
– executes planning code, small reasoning models</td> <td>Parse user request; determine needed steps; allocate tasks to GPU or disk; e.g., parse code specs, plan API calls. [7] [2] </td> </tr> <tr><td>Mapping
(Transformer 2)</td> <td>Execute compute-heavy tasks, generate outputs for each sub-task</td> <td>GPU (thousands of parallel cores)
– runs neural network inference/training, parallel math</td> <td>Run LLM to produce code or text; perform matrix ops or simulation; e.g., generate model outputs for each planned sub-task. [3] [8] </td> </tr> <tr><td>Validation
(Transformer 3)</td> <td>Check and integrate results, perform I/O intensive operations</td> <td>NVMe Storage (SSD with ~GB/s throughput)
– high-speed reads/writes, KV store queries</td> <td>Fetch ground-truth data or prior results for comparison; store outputs; run tests or consistency checks using data on disk. [9] [10] </td> </tr> </table>

In practice, these stages may iterate in a loop (hence *Triple-Transformer Loop*): after Validation, if a result fails checks, the system can re-enter Analysis with that feedback. For instance, if generated code fails a unit test (Validation using NVMe to store test cases and outcomes), the CPU Analysis stage can adjust the prompt or logic and the GPU Mapping stage can regenerate a solution, until the output passes validation. This loop echoes the *analyze–act–verify* cycles seen in autonomous agents. A recent retrosynthesis AI offers an analogy: it used a "triple transformer loop" (TTL) to propose chemical steps (T1, T2) and then validate them with a forward prediction model (T3) [11]. By ensuring a forward validation, the TTL improved reliability. Similarly, in QFaaS computing, the Validation transformer provides a forward-check that the combined classical/quantum action achieved the desired outcome.

Local vs Cloud Considerations: Mapping the pipeline to local hardware confers both benefits and challenges. On one hand, data stays local (important for privacy and sovereignty) and latency between stages is minimal (CPU, GPU, SSD are all on high-speed PCIe interconnect). This tight integration can be more efficient than cloud FaaS which might incur network overhead between functions. On the other hand, the local workstation has finite resources, so the pipeline must be optimized (see next section) to fully utilize hardware and avoid bottlenecks. For example, saturating a modern NVMe (e.g. 7 GB/s read) requires issuing enough parallel I/O requests; if the CPU Analysis stage waits synchronously for one GPU task, the NVMe might be underutilized. Therefore, pipelining and concurrency are key: the CPU can prepare next tasks while GPU runs, and NVMe can stream in the next batch of data in parallel. This is analogous to techniques in high-performance computing where overlapping computation and I/O (with **asynchronous pipelines**) yields stall-free throughput [12] [13].

Within each hardware component, **transformer models or modules specialized to that stage** can reside. For example, the CPU might run a lightweight transformer (like a code analyzer or small GPT) to decide on a

plan, whereas the GPU runs a large transformer (like CodeT5 or a 70B LLM) to generate content. The NVMe stage might use learned Bloom filters or vector indexes stored on disk to validate (one could even consider these as *frozen transformers* in the form of databases). This specialization aligns with efficient use of each component's strengths. Research in heterogeneous ML serving suggests splitting model execution across CPU–GPU can improve efficiency when done carefully [14] [15]. Here we go further by attributing entire sub-tasks to the device that handles them best.

In summary, the Triple-Transformer Loop architecture in sovereign QFaaS systems distributes AI workloads across **local CPU, GPU, and NVMe** in an *analysis* → *mapping* → *validation* loop. This design mirrors a serverless function pipeline but contained within a single machine. The CPU orchestrates and plans, the GPU executes core neural computations, and the NVMe-based storage verifies and feeds data. Together, they form an **autonomous closed loop** that can take a high-level request (e.g. "solve this complex problem") and cycle through solution generation and checking until a valid result is obtained – all without cloud dependencies. The following sections delve into how to optimize such a system for performance (Section 2), how software frameworks enable the planning and orchestration (Section 3–4), and how quantum resources and advanced ML models integrate into this paradigm (Sections 5–7), before examining broader implications (Sections 8–10).

2. Low-Level Optimizations: CPU Pinning, GPU Scheduling, NVMe Tuning, Containers

Building an efficient Triple-Transformer system on local hardware requires careful **low-level optimizations**. Each component (CPU, GPU, storage) must be tuned to deliver maximum throughput and minimal latency, and their interactions orchestrated via containerization and scheduling. We examine key optimizations:

- **CPU Core Pinning and Isolation:** To ensure consistent low-latency performance of the CPU-bound Analysis stage, it's common to *pin* certain processes or threads to specific CPU cores (also called setting CPU affinity). Core pinning keeps a thread on one core, improving cache locality and avoiding costly context switches [16] [17]. This is crucial for real-time analysis and coordination tasks, as it reduces jitter. For example, one can reserve a high-performance core for the main orchestrator thread (Analysis transformer) and isolate it from the OS scheduler's interference. Ampere Computing notes that core pinning yields **more predictable latency and higher throughput** for HPC, embedded, and low-latency apps by keeping threads on the same core to exploit cache warmth and avoid migration overhead [18] [19]. In a Docker container environment, one can use `--cpuset-cpus` to restrict a container to specific cores or use Kubernetes CPU Manager with static policy for guaranteed cores [20] [21]. For instance, the Analysis agent's container might be pinned to core 0-1 (for hyper-threading pairs) to ensure it's always readily scheduled, while other background tasks run on separate cores. Additionally, isolating those cores from general OS tasks (using Linux `isolcpus` or `nohz_full` for no-scheduler jitter) can further guarantee consistent performance for time-sensitive loops [22] [23]. Pinning can also help *reduce non-determinism* in multi-agent timing, aiding reproducibility by providing similar execution ordering across runs.

- **GPU Scheduling and Partitioning:** The GPU-bound Mapping stage can be optimized via smarter scheduling of GPU workloads and even partitioning GPUs for concurrent use. Modern NVIDIA GPUs allow techniques like *Multi-Process Service (MPS)* and *Multi-Instance GPU (MIG)* to improve utilization. **MIG**, introduced with the Ampere architecture (A100/H100), lets one physical GPU be split into up to

7 fully isolated GPU instances, each with dedicated SM cores, memory, and cache [24]. This is useful if the Mapping stage involves multiple concurrent models or tasks – e.g. if our pipeline needs to run several medium-sized transformer models in parallel (each can be assigned to a MIG partition). By using MIG, the system can ensure each task's GPU portion gets guaranteed resources, eliminating interference and queue delays. NVIDIA's documentation emphasizes MIG's ability to securely partition GPUs so that multi-tenant or multi-agent workloads **run simultaneously with QoS isolation** [25]. For example, an agent might run a vision model on one MIG slice while another runs a language model on a separate slice, both on the same physical GPU but without contending for the same memory. Alternatively, **CUDA MPS** allows multiple processes to share a GPU with a common scheduler – this can reduce context switch overhead when many small kernels are launched by merging them. A Medium case study found that combining MIG (for coarse isolation) and MPS (for fine sharing within a slice) yields safe GPU sharing with lower tail latencies in production, compared to naive time-slicing [26]. Beyond MIG/MPS, one can adjust the GPU's scheduling policy (on Linux via `nvidia-smi` settings) – e.g. use **compute exclusive mode** to prevent context thrashing, or utilize **stream priorities** if certain inference tasks must preempt others. For latency-critical validation steps on GPU (if any), one could assign higher priority CUDA streams to those kernels so they run before lower-priority background work [27]. In sum, *the GPU should be viewed as a schedulable resource like a mini cluster*: by partitioning and prioritizing tasks on it, the system maximizes throughput and responsiveness.

- **NVMe Tuning (Scheduler, Read-Ahead, TRIM/KV):** NVMe SSDs are a backbone for the Validation stage and any large data handling. Tuning the Linux I/O stack for NVMe can significantly improve performance. First, the **I/O scheduler** for NVMe is often best set to `none` or `mq-deadline`, since NVMe devices handle internal scheduling well. Using the noop scheduler avoids extra queuing latency. A recommended practice is `echo none > /sys/block/nvme0n1/queue/scheduler` to disable Linux scheduling for that device [28]. Next, adjusting **request queue affinity** can reduce CPU overhead: setting `rq_affinity=2` pins completion interrupts to the core that submitted the I/O, improving cache locality (the NVMe driver supports this) [28]. In fact, a sample udev rule for NVMe tuning sets `scheduler=none` and `rq_affinity=2` automatically on device add [29]. **Read-ahead** is another tunable: for predominantly random I/O, a small read-ahead (e.g. 128 KB) is better to avoid wasting bandwidth, whereas for streaming large files, a larger read-ahead (e.g. 4 MB) can boost throughput [30] [31]. One can dynamically set `blockdev --setra` or via sysfs; for instance, an NVMe used as a random-access key-value store might use 128 KB read-ahead [30], while one used for sequential logging might use more. **TRIM scheduling** is also important: periodic TRIM frees deleted blocks, maintaining SSD performance over time. It's recommended to use a weekly `fstrim.timer` rather than mounting with continuous discard, to avoid runtime overhead [9] [32]. Enabling `systemctl fstrim.timer` will schedule efficient weekly TRIMs [33]. For QFaaS workloads that create and delete many temporary files (e.g. caching intermediate results), this ensures the NVMe doesn't bog down due to fragmented free space. If using the NVMe **Key-Value (KV) command set** (available on some drives and enabled in NVMe 2.0 spec), one might consider *KV-specific optimizations*: for example, Samsung's KV SSDs allow direct object delete commands which internally handle garbage collection in a KV-optimized way [6]. Although KV support is still emerging, it could accelerate retrieval in the Validation stage by bypassing the filesystem entirely – essentially treating the NVMe as a giant dictionary. Finally, monitoring NVMe latency and throughput via tools like `nvme smart-log` helps validate that tuning is effective (one should see lower average and tail latencies after tuning [34] [35]).

- **Docker Container Orchestration and Cgroup Tuning:** The Triple-Transformer loop is likely deployed in containers for modularity (e.g. separate containers for Analysis agent, GPU inference service, and a database or filesystem watcher for Validation). Using Docker or Kubernetes, one can assign resource limits and scheduling settings to each container to reflect its role. For instance, the Analysis container can be given a **real-time scheduling class** or higher CPU share (using `--cpu-rt-runtime` and `chrt` within container) if ultra-low latency is needed. The GPU container (Mapping) can be given access to GPU devices via NVIDIA Docker, and if MIG is used, one can directly allocate a specific MIG instance to the container (using the MIG UUID to avoid the container seeing full GPU) [36] [24]. This ensures the GPU container only uses its partition. For NVMe access, using `--device` flags with Docker gives direct host device access (bypassing the overlay filesystem for data volumes, which is preferable for high I/O). Additionally, cgroup I/O throttling can be turned off for the Validation container if it needs full NVMe bandwidth (cgroups v2 IO controllers allow weight or max limit; by default, all containers share fairly). We would assign, say, high `io.weight` to the validation container if its disk checks are critical. Container orchestration also aids in *affinity*: one can deploy the Analysis container on a specific NUMA node or socket if the system has multiple CPU sockets, to be closer to the NVMe controller or GPU (leveraging Kubernetes topology hints or `numactl` inside the container) [37]. This avoids cross-socket memory traffic.

Another aspect is **process priority**: the OS can prioritize the Validation stage if needed (for example, if writing results to disk should not block too long, the I/O scheduler can favor those writes by using `ionice` with high priority). Conversely, one might deem the background storage of logs as low priority (using `ionice -c3` best-effort).

A final note on **network** (though everything is local, loopback networking may be used if microservices communicate): one should use shared memory or UNIX domain sockets for inter-container comms to cut latency (Docker allows containers to share IPC namespaces or use volumes for memfd). If using gRPC/HTTP between Analysis and Mapping components, enabling TCP fast loopback and increasing OS socket buffers can help throughput.

Collectively, these optimizations aim to **eliminate bottlenecks and unpredictable delays** in each stage:

- CPU pinned -> consistent orchestration timing [16] [17].
- GPU partitioned/scheduled -> high utilization and concurrency with bounded latency [25] [26].
- NVMe tuned -> minimal I/O overhead, maximum throughput (no kernel queuing, appropriate read-ahead, periodic trim) [9] [38].
- Docker/cgroup configured -> each component gets appropriate resources and isolation, aligning with its needs (preventing e.g. the GPU container from starving the CPU one).

These hardware/software tuning techniques are validated by extensive experience in high-performance computing and server tuning literature. For example, a Linux NVMe optimization guide demonstrates that setting `scheduler=noop`, `rq_affinity=2`, and moderate read-ahead on NVMe yielded lower 99th percentile latencies and higher IOPS in microbenchmarks [34] [35]. Similarly, core pinning is a standard in financial trading systems to cut down unpredictable latency spikes [39] [23]. GPU MIG is a newer but increasingly adopted practice; cloud providers have begun offering MIG slices so that smaller models can run in parallel, showing better overall throughput per GPU for mixed workloads [36] [26].

In implementing a sovereign QFaaS workstation, one might script these tunings as part of the deployment (e.g. using an Ansible playbook or Dockerfile commands). Ensuring these low-level details are handled provides a solid foundation so that higher-level agent frameworks (next section) can operate smoothly without worrying about thread jitter or I/O hiccups. The result is a *deterministic and efficient execution environment* for the triple-transformer pipeline, which is particularly important as we layer on complex multi-agent and even quantum components.

3. Agentic Development Frameworks: Antigravity, Conductor, ATHENA, and Cloud vs Local

Designing software that can effectively leverage the Triple-Transformer architecture requires advanced **agentic development frameworks**. These frameworks provide high-level tools for building autonomous agents that plan, execute, and verify tasks – aligning well with the Analysis–Mapping–Validation stages identified above. We examine several state-of-the-art frameworks and protocols: Google's **Antigravity IDE**, the **Conductor** planning protocol (with its Model Context Protocol integration), the **ATHENA** agent hierarchy for research tasks, and we discuss the trade-offs between local (on-premise) vs cloud implementations of such agent systems.

3.1 Google Antigravity IDE: Agent-First Software Development

Google **Antigravity** is a recently introduced "agentic development platform" (launched Nov 2025) that exemplifies the cutting edge of AI-assisted software engineering [40]. Unlike traditional IDEs that are passive tools, Antigravity integrates multiple AI agents into the development workflow. It provides two main interfaces: the **Editor View** and the **Manager Surface** [41]. In Editor View, developers write and edit code with AI assistance (tab completions, inline suggestions). More novel is the Manager Surface, which allows spawning and orchestrating *multiple agents working asynchronously* across the IDE, terminal, and browser [42] [43].

This design directly corresponds to our triple-loop: one can imagine an **analysis agent** in Antigravity that interprets a feature request and breaks it into tasks (e.g., update backend API, adjust database schema, then modify frontend), then delegates each to specialized agents (coding agent, database migration agent, test agent). Antigravity's philosophy is that *agents shouldn't just live in chat sidebars but have their own workspace* to carry out complex plans [44]. For example, an agent could autonomously create a new source file, run `git` commands in the terminal, or open relevant documentation in a browser tab to gather context [45] [41]. The developer can oversee and intervene via the Manager Surface, approving or adjusting the plan.

In practice, Antigravity demonstrates how an agent loop can implement Analysis–Mapping–Validation: the agent might *plan* (analysis) a multi-step refactor, *execute code edits and run tests* (mapping on the coding task, using underlying GPU-powered models for code generation), and *verify the results* (validation by running test suites). Google's introduction of Antigravity highlights use cases like delegating multi-tool tasks to an agent (e.g., "Set up OAuth for my app" – the agent can edit code, run commands, and verify it works) [46]. By doing so, Antigravity moves developers to a higher abstraction, focusing on **task orchestration** rather than line-by-line coding.

One important aspect of Antigravity is its support for *multiple asynchronous agents*. This inherently requires a robust context-sharing and scheduling mechanism – likely leveraging protocols like those in Section 4. The Manager Surface likely coordinates context so that agents are aware of the project state and each other's progress. It also embodies a *mixed-initiative* paradigm: agents do a lot autonomously, but the human developer remains in control, reviewing plans before code is written and keeping "the human developer firmly in the driver's seat" [7] [47]. This is crucial for safety and correctness; the agent proposes but the human disposes (at least in oversight capacity).

From a sovereign computing perspective, one might run a local version of such an agentic IDE (Google's is presumably cloud-powered via their Gemini 3 model and others [48] [49]). If implementing Antigravity-like capabilities locally, an organization might deploy an open-source IDE plugin that uses local LLMs as the agent brains. The architecture would align well with our pipeline: the planning agent's reasoning (maybe smaller model) could run on CPU, the code generation model runs on GPU, and the validation agent uses local test databases on NVMe.

Antigravity's emergence signals a **new era of software development** where AI agents actively collaborate with developers. It provides a concrete example of how the *Triple-Transformer concept maps to real tools*: planning agents (Analysis) and coding agents (Mapping) with verification loops (running tests, Validation). Moreover, it highlights interface design – giving agents a "space" to work (the Manager view) acknowledges they are more than just chatbots; they are co-workers managing sub-tasks.

3.2 Conductor Planning Protocol and Model Context Protocol (MCP)

While Antigravity focuses on the IDE experience, **Conductor** is a framework and protocol aimed at planning and controlling AI agent workflows, especially with context management. Conductor was introduced as an extension for Gemini CLI (Google's AI command-line interface) in Dec 2025, emphasizing *context-driven development* [1]. The core idea of Conductor is to formalize specs and plans in Markdown files that live alongside code, which the AI agents then follow [50] [47]. This approach shifts from ephemeral chat to persistent plans (e.g., a `PLAN.md` that outlines steps to implement a feature, which an agent then executes).

Conductor provides a **planning protocol** where developers or product managers write structured task descriptions that the AI uses to generate and coordinate actions. It ties into the **Model Context Protocol (MCP)** – an open standard spearheaded by Anthropic (and supported by others) that allows AI assistants to securely connect to tools and data [51] [52]. Specifically, Conductor offers a *Conductor MCP server*, which exposes all of Conductor's API endpoints (the tasks, specs, etc.) as MCP tools [53]. This means an AI agent (like Claude or ChatGPT) can query the Conductor server to get project context or update plans, using a unified protocol for context exchange.

One way to think of Conductor is as a *conductor–creator architecture*: a "conductor" agent plans high-level objectives and delegates subtasks to "creator" agents that generate outputs (code, content) [54] [55]. The conductor ensures consistency with specs and keeps track of progress. The Conductor CLI extension encourages developers to "measure twice, code once" by having an explicit planning step [56]. For example, instead of immediately prompting "build this feature" into an LLM, a developer writes a spec in Markdown. Conductor then uses that to steer the LLM – ensuring it knows the acceptance criteria, style guides, etc., which are now *part of the persistent context* [57] [2].

The **Model Context Protocol (MCP)** piece is crucial for multi-agent collaboration (further discussed in Section 4). MCP standardizes how tools (like Conductor, or a code repository, or a database) are exposed to AI, and how multiple AI models can share context. Anthropic's vision for MCP is to be a *"USB-C for AI applications"*, enabling a plug-and-play way for models to access organization-specific data securely [51] [58]. Conductor uses MCP so that, for instance, an AI agent can directly ask Conductor "what are the open plans?" or "create a new plan with these steps" – effectively controlling the software development workflow through an API rather than via freeform chat only [59]. The **Conductor MCP Server** specifically lets agents create, run, and review orchestrated workflows [60]. It turns a conceptual plan into actual CI/CD pipeline actions.

One tangible example: Suppose we have a repository with a `FEATURE_request.md` describing a new feature. Conductor reads it and generates a plan with steps (update schema, modify API, write tests). Each step might correspond to a script or command. Through MCP, an AI agent (Claude with Conductor's MCP integration [61]) can sequentially call tools: e.g. call "Conductor.RunStep('update_schema')", then call a GitHub MCP tool to open a PR, etc. Thus, *Conductor + MCP* enables **agentic CI/CD** where multiple tools are orchestrated via a standard protocol. In the absence of MCP, such orchestration often relied on ad-hoc prompt engineering ("You have tool X and Y, respond with JSON to call them"). MCP formalizes this as a two-way API.

From a local vs cloud perspective: Conductor and MCP can be run locally. Indeed, Conductor open-sourced a portion on GitHub [1], and MCP servers are available for self-hosting (Anthropic released connectors for Git, Slack, etc. that one can run [62] [63]). A sovereign AI stack could deploy an internal Conductor server plus local MCP connectors to internal data (like a private Confluence or JIRA). The benefit is that even if the language model itself is local, it can use the same open protocol to fetch context or perform actions on local systems securely. The alternative is being locked to one vendor's tool integration, but MCP is an **open standard (Nov 2024)** backed by multiple companies [51] [58].

Conductor's emphasis on *"plan-first, keep human in loop"* also addresses a key safety concern: by reviewing AI-generated plans before execution, developers can catch missteps early [64] [65]. It's essentially inserting a human validation between our triple-transformer stages, which is often wise in high-stakes coding (the developer acting as a final Validation). As an example, Keith Ballinger (Google VP) said Conductor "allows you to plan before you build, review plans before code is written" – bringing **control and predictability** to AI coding [1] [2]. This resonates with ensuring trust: in critical systems, autonomous agents must present their reasoning (plans) for approval, akin to how autopilot systems still allow pilot override.

In summary, **Conductor and MCP** provide the glue for complex multi-agent systems in development workflows. Conductor turns specifications into structured plans that an agent can execute, and MCP ensures that agents can communicate with tools and each other in a secure, standardized way [58] [62]. These frameworks reduce the "black box" nature of LLM-driven development by externalizing memory (Markdown specs) and standardizing tool use. For a sovereign AI developer workstation, adopting Conductor's methodologies means more reproducible and governable agent behavior, since everything the agent does is anchored to persistent context files and tool API calls rather than ephemeral prompts. Section 4 will delve deeper into how MCP enables multi-agent collaboration beyond just development use cases.

3.3 ATHENA: Hierarchical Agent Teams for Complex Problem Solving

While Antigravity and Conductor focus on software development tasks, the **ATHENA** framework (Agentic Team for Hierarchical Evolutionary Numerical Algorithms) exemplifies agentic AI applied to scientific research and problem-solving [66] [67]. ATHENA (introduced on arXiv in Dec 2025) is essentially an **Autonomous Research Lab**: a system of multiple agents with a hierarchical structure, designed to tackle complex scientific computing tasks end-to-end [68].

ATHENA's core is the **HENA loop** – a knowledge-driven diagnostic loop framed as a contextual bandit problem [68]. In essence, the system iteratively proposes actions (e.g., try a certain numerical solver or neural network architecture), executes them, and observes rewards (like solution accuracy) to inform the next iteration [69] [70]. This is very similar to an analysis-mapping-validation cycle: *analyze past trials, map an action to try, validate via reward*. Over many iterations, ATHENA converges on solutions like discovering new algorithms or tuning models to super-human performance (reportedly reaching errors of 1e-14 in some tasks where human-designed solvers struggled) [67].

What makes ATHENA notable is its **hierarchical team of agents**. It doesn't rely on a single monolithic agent; instead, it's organized into logical groups with specific roles [71] [72]. For example, some agents might propose candidate equations (creative generation), others verify stability or compliance with physics (critics), and others manage the overall experiment pipeline (a meta-controller). This hierarchy maps well onto the triple-transformer idea: a top-level agent (analysis) decides which approach to pursue (e.g., "try a PINN approach on this PDE problem"), lower-level agents (mapping) carry it out (e.g., train a Physics-Informed Neural Network on GPU), and validation agents check results (e.g., compare to known symmetries or conservation laws, potentially using cached data) [73] [74]. If validation fails, the system adapts: maybe next iteration it chooses a different approach (like a finite-element solver) or tweaks hyperparameters.

ATHENA's **performance and approach** highlight a few important concepts for sovereign AI:

- **Hierarchical Planning:** Complex tasks are broken down across multiple agents that operate at different levels of abstraction. This is akin to a manager (high-level strategy) and workers (specific tasks). In ATHENA's case, the manager agent selects structural actions (like what method or what form of equation) while worker agents implement details [75]. This avoids overwhelming a single agent with both high-level creativity and low-level execution.

- **Learning from Iteration (Evolutionary Loop):** The system improves over time by learning what worked. This is facilitated by retaining a **knowledge base** of prior trials and outcomes (which can be stored on NVMe – e.g., a database of experiments and their rewards). Each loop is essentially an experiment logged and fed back. This resonates with the *Validation stage using past data* idea: by storing each attempt's result, the system's Validation agent can inform the Analysis agent which strategies pay off [69] [70]. It's a form of automated *scientific method*: hypothesis, experiment, observe, refine.

- **Autonomy with Human Oversight:** ATHENA reportedly achieved super-human performance but also allowed "collaborative human-in-the-loop intervention" to bridge stability gaps, further improving results by 10× in some cases [67]. This underscores that even autonomous agent labs benefit from occasional human insight, especially for corner cases or to inject domain knowledge. In practice, ATHENA might flag when an agent is stuck (no progress in reward) and ask a human

mentor agent for input – analogous to how we would step in if our autonomous coding agent is confused.

- **Agent Safety and Specialization:** Each ATHENA agent has a clear scope (e.g., one agent's job is to ensure numerical stability, another's to find symmetries) [67]. This specialization can act as a safety mechanism because agents check each other's outputs. For example, a *physics-checker agent* might veto a solution that violates energy conservation. In multi-agent setups, such cross-checks are crucial to avoid runaway errors. It's similar to having a Validation agent that's distinct from the generation agent, providing an independent audit.

The ATHENA framework's success in scientific domains suggests a template for **agent hierarchies** in other domains. One could envision an ATHENA-like system for enterprise tasks: e.g., an autonomous business analyst that tries strategies to improve a metric (sales, efficiency) by launching various sub-agents (marketing plan generator, supply chain optimizer, etc.), measuring results, and iterating. The key is breaking the problem into a hierarchy and learning from feedback.

Technically, implementing ATHENA on a local workstation would require running multiple agents (some possibly large models) in parallel. This could stress our CPU–GPU pipeline, but optimizations from Section 2 (like MIG partitioning GPUs or scheduling tasks) could help. For example, each agent might run on a schedule: the analysis agent runs on CPU and issues tasks for mapping agents on GPU slices, which produce outputs that validation agents (maybe CPU or smaller GPU tasks) then evaluate. Given ATHENA was shown to manage complex numeric experiments, it likely involves orchestrating classical computations too (like solving equations with traditional methods) – hence integrating classical code execution with learning agents.

In summary, **ATHENA demonstrates the power of structured multi-agent hierarchies**. It confirms that having an *agentic team* with a well-defined loop (HENA) can outperform singular approaches in complex tasks [67]. For sovereign AI, it means that users can deploy not just one generalist LLM, but an ensemble of specialist agents that collaborate. The architecture might involve an overseer agent (for high-level planning and ensuring alignment with user goals) supervising expert agents (for coding, for math, for compliance checks, etc.). Google's Antigravity takes a step in this direction by allowing multiple agents in a dev environment, but frameworks like ATHENA push it further by adding learning and adaptation over repeated trials.

3.4 Local vs Cloud Implementation Trade-offs

Given these frameworks (Antigravity, Conductor/MCP, ATHENA), a critical question is: **Should the agentic system run locally or in the cloud?** There are trade-offs:

Cloud Implementation (managed services, online APIs): - *Pros:* Access to extremely powerful models (like OpenAI GPT-4, Google Gemini) that far exceed what can run locally. Cloud agents can tap into cloud-only data or services (if allowed) and scale dynamically (spin up more instances for parallel tasks). The maintenance of the AI models (updates, fine-tuning) is handled by providers. For example, Google Antigravity presumably uses cloud-hosted Gemini 3 and maybe Anthropic Claude or other APIs [49]. This gives users top-tier model performance without local hardware requirements. Cloud also simplifies multi-agent coordination across devices – e.g., Conductor's cloud backend can connect an IDE, a web app, and a database simultaneously via shared cloud context. - *Cons:* Dependency on internet connectivity and third-

party services. There are latency penalties (each agent action might have to call an API, which could be 100–500 ms each way). Data privacy is a concern – sending code or proprietary specs to a cloud LLM risks leakage or compliance issues. There's also **cost**: running large models via API is expensive at scale (token-based pricing can add up). And customization is limited to what the provider offers (maybe you can't easily fine-tune GPT-4 on your internal codebase, whereas a local model you could). Additionally, cloud models may update or change behavior without notice, hurting reproducibility [76] [77].

Local Implementation (on-premises/sovereign): - *Pros:* Full control over data and model behavior. All code, plans, and context remain on the user's machines (addressing sovereignty and IP concerns). One can run open-source models and even tune them to match organizational needs or cultural values (ensuring the AI's outputs align with local norms, see Section 6 on alignment). There's potentially lower latency for each model inference (no network hop; though if models are smaller or quantized to fit on local GPUs, their quality might be a bit lower than the largest cloud ones). Local frameworks can be deeply integrated with local dev tools, e.g., instant file system access, zero-trust issues (the agent can use shell commands freely on the machine, which cloud AI cannot for security). Also, cost may be lower in the long run: once hardware is purchased, using it for unlimited inferences can be cheaper than per-call cloud fees, especially given surveys showing **85% of enterprises are trying to rein in AI cloud costs** [78] [79]. A 2024 Lenovo study on generative AI TCO found that beyond a certain scale, on-prem GPU clusters have lower hourly cost than paying usage fees for equivalent cloud compute [80]. - *Cons:* Requires significant up-front investment in hardware (GPUs, high-performance CPUs, large NVMe storage as we assumed). Not every organization can afford an 8×A100 server for AI. Local models may lag in capability behind state-of-the-art closed models – though the gap has been closing (e.g., open DeepSeek models claim to approach closed GPT-4 quality [81]). Maintenance and engineering overhead is higher: you need teams to manage model updates, optimize runtime (the stuff in Section 2 needs expertise). Also, scaling out is limited by hardware; a local workstation might handle a few agents concurrently but not hundreds of parallel processes that a cloud backend could if needed.

A **hybrid approach** is also possible: where the orchestrating logic and some agents run locally, but heavy lifting is offloaded to cloud models. For instance, one could run Conductor locally (so all plans/specs stay internal), but have the Mapping stage call a cloud API for actual code generation. This keeps the sensitive specs in a structured format (only selectively sending what's needed to the cloud model). However, this still shares data and is subject to the cloud's reliability.

Trends indicate some *repatriation* of AI workloads from public cloud to on-prem private clouds as organizations seek more control and predictable costs. A Nutanix survey in 2024 found **85% of organizations moving at least some cloud workloads back on-prem** for AI, citing cost and security [79]. Moreover, national regulations (data residency laws, etc.) sometimes demand local processing.

From a technical lens, frameworks like Antigravity and Conductor could be deployed in both modes. Google's official Antigravity uses their cloud, but one could conceptually replicate it with local models (using, say, VS Code + open-source agent backends). Conductor is just a planning layer – it doesn't mandate where the model runs. In fact, Conductor's integration with MCP means it could coordinate either a local LLM (if one implements an MCP client around it) or a remote one.

Which mode is better? It often comes down to use case: - For personal or small team use (e.g., an indie developer or a researcher wanting maximum privacy), local might be favored. The cost and scale needs are manageable (a single high-end PC with an RTX 4090 can run a 20B parameter model at decent speed, and

4-bit quantization can push that to maybe a 30B model). - For enterprise at massive scale, cloud can offer elasticity – you can ramp up agent usage during the day and down at night. However, large enterprises also invest in their own datacenters (e.g., banks or healthcare might use **hybrid cloud**: on-prem for sensitive tasks, cloud for general ones). - Cloud models currently still have an edge in raw capability (e.g., GPT-4's complex reasoning). If a specific task truly needs that edge, a sovereign setup might incorporate a call to GPT-4 only for that piece and do the rest locally, balancing trade-offs.

To illustrate trade-offs with an example: Consider an **intelligent compliance agent** that scans internal documents for regulatory risks (see Section 7). If run on cloud, all those documents (which might be confidential) have to be uploaded to the AI – unacceptable for many compliance departments [82]. Running it locally with an open model fine-tuned on legal text is more private, but maybe that model is slightly less accurate than a cutting-edge cloud one. Many will choose privacy over a small accuracy gap, especially since domain-specific fine-tuning can mitigate quality differences (open models can be specialized to an organization's jargon, which a generic closed model might misunderstand).

Connected source citations back this: a **Linux Foundation Europe** report on the EU AI Act notes that open-source developers are concerned about sending data to big foundation models and are pushing for more local AI to maintain **AI sovereignty** [83] [82]. That phrase – AI sovereignty – echoes the sentiment that controlling the AI (in terms of deployment and value alignment) is as important as controlling one's data. Indeed, as one article put it, "AI sovereignty is the new data sovereignty... now we're arguing not just about who owns your data, but who gets to define reality" [82].

Thus, *cultural and strategic factors* also play a role: countries or organizations may prefer local AI to avoid dependence on foreign cloud providers, ensuring their AI systems operate under their laws and norms. This is leading to some **Balkanization of AI** (discussed in Section 10) where we'll see localized models for different regions and ideologies [84] [85].

In conclusion, the choice of local vs cloud is a balancing act between **performance/capability (cloud advantage)** and **control/privacy (local advantage)**. The trend in frameworks is to be *agnostic* to this: Conductor+MCP, for instance, can work with either, and A2A (Agent-to-Agent protocol, see Section 4) is about interoperability regardless of underlying model origin [86] [87]. In practice, a sovereign AI solution might start leveraging cloud for what's hard to do locally, but progressively shift on-prem as open models improve – thereby incrementally gaining sovereignty. With models like DeepSeek V3 (671B parameters) claiming to rival closed models [88] [81] and hardware like H100 enabling FP8 to train such models efficiently (37% less time, 42% less memory for a 175B model [89] [90]), the gap is closing faster than anticipated. This suggests that by 2025–2026, fully local agentic systems that are competitive with cloud offerings will be feasible, given sufficient investment in hardware and engineering.

4. Multi-Agent Collaboration via Model Context Protocol (MCP)

As AI systems grow from single agents into *societies of agents*, effective **multi-agent collaboration mechanisms** become critical. The **Model Context Protocol (MCP)** has emerged as a key enabler for such collaboration, by providing a standardized way for agents and tools to share context and communicate. In this section, we delve into how MCP facilitates multi-agent systems, including context sharing across agents, managing communication latency and efficiency, and maintaining safety at the agent level (ensuring agents do not behave malignly when collaborating). We will also touch on complementary

protocols like Google's **Agent-to-Agent (A2A)** communication standard, which addresses direct agent interoperability [91] [92].

4.1 Model Context Protocol (MCP): A "USB-C for AI" Enabling Shared Context

MCP, introduced by Anthropic in late 2024, is described as a **universal open standard for connecting AI systems with data sources and tools** [51] [58]. Think of MCP as a language that both AI assistants and external services speak, so they can exchange information systematically. For multi-agent setups, MCP serves as a *shared memory bus* and tool interface combined.

Key features of MCP relevant to multi-agent collaboration:

- **Unified Context Space:** MCP allows different agents (which could be different LLM instances, possibly from different vendors) to maintain a shared context about a task. For example, if Agent A finds some relevant data in a PDF via an MCP document connector, it can store that info on an MCP server; then Agent B, working on the same task, can retrieve it. This is done without hacky prompt copy-pasting but via explicit calls. Anthropic likens this to breaking down information silos – with MCP, context moves with the agent and between agents rather than being stuck in one session [93] [58]. In practice, an MCP **server** holds certain state or provides an interface, and agents as MCP **clients** connect to it. Early implementations (Claude Desktop, Cloudflare's agents, etc.) show that multiple tools or agent processes can connect to one local MCP server, effectively syncing their knowledge [62] [63].

- **Tool Access as API:** As mentioned in Section 3, MCP standardizes tool use – e.g., an agent can say "call tool X with these params" in a JSON/HTTP format rather than natural language guessing. This reduces misinterpretation and makes multi-step tool use more reliable. For multi-agent, one agent might expose itself as a tool via MCP. Indeed, the *Agent2Agent (A2A) protocol* from Google explicitly allows an agent to advertise its capabilities to others via an "Agent Card" (a JSON descriptor of what it can do) [94]. This is analogous to tools in MCP. In essence, one agent can call another as easily as calling a calculator API, if they both follow MCP/A2A standards.

- **Secure, Two-Way Connections:** MCP is designed with security in mind, given the concerns of letting AI loose on data. Agents authenticate to MCP servers (Anthropic built support for Claude to connect to local MCP servers with user approval) [62]. Data flows both ways: agents can *retrieve* context (like documents, database queries) and can *push* context or results (like writing a summary back to a knowledge base). This two-way flow is crucial for collaboration: an agent doesn't have to do everything itself; it can delegate subtasks to the environment (e.g., store intermediate results via MCP for others to pick up, or ask another agent for help via an MCP call).

Context Sharing and Latency: One big challenge in multi-agent systems is keeping agents on the same page (literally, the same context). If each agent had its own separate prompt history and no shared memory, they might duplicate efforts or even give inconsistent outputs. MCP addresses this by letting them synchronize through a common repository of context. For instance, consider a team of agents solving a research problem (like ATHENA or a NotebookLM scenario): one agent focuses on literature review, another on experiment design, another on writing the report. Using MCP, the literature agent can deposit key findings into a context store; the writing agent can query that store to incorporate those findings into the report. This is far more efficient than trying to prompt one giant agent with everything or manually copy

outputs between them. It also means each agent can work *in parallel* on its part, then merge contexts, rather than strictly sequentially waiting on each other (which would be slower).

Communication latency is a consideration: if agents constantly ping a context server or each other, could it become a bottleneck? MCP is usually implemented over localhost or internal network if local – that's very fast (microseconds to low milliseconds). The overhead of MCP calls is much smaller than the time it takes for a model to generate text. For example, Cloudflare measured that using MCP to let an AI agent retrieve data from a local source had negligible overhead compared to the model's processing time [95] [96]. The heavier cost is often *embedding or converting context* into a form the model can use (like vector embeddings for semantic search). But MCP can even aid retrieval efficiency by connecting to vector DBs directly as tools, rather than forcing the agent to hold huge context windows.

Indeed, the use of **vector databases and retrieval augmentation** is complementary to MCP. Instead of giving an LLM a 100k token context window with all possibly relevant info (which is memory and compute heavy), one can store documents in a database and let agents query via MCP (e.g., "find me info on X in our docs"). This means context is *pulled on demand* and not always stuffed into the prompt, improving both latency and correctness (since retrieval can be up-to-date and targeted). Deep retrieval systems like NotebookLM use similar strategies – NotebookLM supports adding multiple sources (Google Docs, PDFs, Sheets) to a notebook and the AI dynamically fetches relevant bits when answering questions [97] [98]. If NotebookLM's agents were built on MCP, one agent might be responsible for retrieving from Drive, another for summarizing, etc., coordinating via a shared notebook context.

Agent-Level Safety and Coordination: Multi-agent systems raise the question: how to ensure agents do not conflict or go off track? Several approaches facilitated by MCP and related protocols:

- **Role Definition and Capability Discovery:** A2A (Agent2Agent protocol) emphasizes that agents should advertise their capabilities and roles up front (via an Agent Card) [94]. This allows a client agent to know what a remote agent can do and choose the best one for a sub-task. For safety, if an agent knows that another agent is an expert in compliance checking and always follows certain rules, it can defer to that agent for those questions. Conversely, it might know not to use an agent outside its expertise (e.g., don't ask the image generator agent a legal question). By dividing responsibilities clearly, you avoid situations where agents attempt tasks they shouldn't (reducing errors).

- **Chain-of-Thought Transparency:** When agents share context via MCP, it's possible to log not just final answers but intermediate reasoning. For instance, Anthropic's Claude and others can produce a **chain-of-thought (CoT)** if allowed. In a multi-agent setting, one could design it such that each agent's CoT (or summary of it) is stored on an MCP server accessible to a supervisor agent or the user. This means we can audit why agent A recommended action B. A study on censorship in DeepSeek AI found that examining the model's chain-of-thought can reveal suppressed content that didn't appear in the final answer [99] [100]; similarly, in an agent team, having CoTs accessible can help diagnose misalignment or biases – e.g., a safety agent could scan another agent's reasoning for red flags (the analog of "Thought Police" agent ensuring no unethical plan is being formed). MCP could enable such cross-agent introspection by treating one agent's reasoning as data that another can analyze, if configured.

- **Sandboxing and Authority Control:** Because MCP defines explicit channels for actions (rather than freeform natural language behind the scenes), it is easier to put guardrails. For example, an agent might have the ability to propose deleting a file, but that request goes through an MCP tool which checks if this is allowed (maybe requiring a human confirmation if trying to delete more than 10 files, etc.). Agents can be sandboxed by limiting which MCP tools they have access to. If we have a database tool and a shell tool, we could let a data-analysis agent use the database tool but not give it shell access, preventing it from doing harm outside its domain. Such fine-grained permissioning is part of MCP's design – each connector can require certain auth, and presumably agents can be given scoped credentials. As RedHat's security brief notes, MCP is powerful but one must manage keys and scopes to prevent an agent from abusing connections [101] [102].

- **Latency vs. Autonomy Trade-off:** There is a design decision on how frequently agents synchronize context. Too frequent (every tiny step) could slow progress (agents spend more time talking than doing). Too infrequent and they may diverge. A solution is to adopt a **coarse plan, fine steps** approach: e.g., the conductor agent (using Conductor protocol) might lay out a plan with 5 steps. Then a creator agent executes step 1 to completion without needing to check in until done (unless it hits a problem). Only after completing step 1 does it update context and proceed. This batching of autonomy keeps latency reasonable. Human teamwork analogies apply: you don't call a meeting for every line of code changed; you implement a chunk then integrate. Similarly, agent collaboration works best with defined intervals of synchronization (like at plan boundaries or after a subtask is done). MCP enables that because the context store can act as the meeting point – e.g., an agent publishes "Step 1 result ready" to the context, triggering the next agent to pick it up.

The **Agent2Agent (A2A) protocol**, announced by Google in 2025, complements MCP by focusing on inter-agent communication across organizations and platforms [103] [104]. It's essentially an extension of the ideas in MCP to multi-party scenarios, with principles: - Agents collaborate even if built by different vendors/frameworks [86] [105]. - Security with enterprise authentication (so an agent from company X can talk to Y's agent with proper auth) [106] [107]. - Support for long-running tasks with state updates (similar to MCP's persistent context, A2A emphasizes real-time feedback and state streaming for tasks that take long) [108] [109]. - Modality-agnostic (A2A can handle not just text but audio, etc.) [110] [111].

A2A explicitly is meant to complement MCP: "A2A focuses on agent-agent collaboration... complements Anthropic's MCP (connecting to tools)" [112] [87]. In practice, A2A might ride on top of MCP or integrate – e.g., two agents might share context via MCP but also have a direct messaging channel via A2A for negotiation.

Agent-Level Safety in multi-agent systems is a rich area of research. One scenario often discussed is emergent behavior: will agents forming a loop produce unintended strategies (the classic example: agents in a competitive game finding a loophole or in collusion)? Ensuring safety might involve: - Using **Constitutional AI** principles where certain agents in the mix are explicitly devoted to upholding constraints. For example, Anthropic might deploy a "Constitution enforcer" agent that reads all outputs and ensures they follow a given constitution (some initial works have agents that self-critique or peer-critic outputs along ethical guidelines [113]). - Encouraging **multi-agent debate**: have two agents with opposing viewpoints discuss and then let a judge agent (or human) decide. This can reduce biases and catch errors because one agent might point out flaws the other missed (if set up adversarially in a productive way). MCP can facilitate debate by having a shared log where agents put arguments and counter-arguments.

Finally, regarding **efficiency**: retrieval efficiency improves as more context can be stored externally rather than each agent having to re-fetch or regenerate knowledge. If one agent reads a 100-page document and summarizes key points to context, others don't all need to re-read those 100 pages – saving compute. The **MCP connectors** can also index data for fast search (e.g., a Git MCP connector might have already cached repository contents, so any agent query about code is O(1) from that cache). In essence, MCP can serve as a collectively built **episodic memory** for the agent team, improving with use. Early adopters like Block (Square) have praised open standards like MCP for making AI context access more *"accessible, transparent, and rooted in collaboration"*, enabling removal of mechanical burdens so people (and by extension, agents) can focus on creativity [114] [115].

The outcome of applying MCP in multi-agent sovereign AI is a system where agents are not isolated minds but members of a shared workspace, communicating through well-defined channels. This markedly improves their ability to handle complex, multi-faceted tasks and to do so safely. As A2A and MCP mature, we can expect multi-agent ecosystems that are **interoperable** (agents from different makers working together), which is crucial to avoid one company's AI monopolizing or fragmenting the space. Instead of "Agent from vendor A only works with tool by vendor A", we'll have an internet-like environment of agent services speaking a common protocol – much like applications on the web follow HTTP/REST. This open collaboration ethos is likely to accelerate innovation and also allow more oversight: it's easier to monitor a standard channel for misuse (like we have security tools for network protocols) than to deal with opaque monolithic AI.

Next, we turn to hybrid quantum-classical aspects (Section 5), which introduce another dimension of complexity where these agent frameworks have to schedule quantum tasks. We will see how concepts like QVMs tie in and how multi-agent orchestration extends to managing quantum resources via deep reinforcement learning strategies like VQR-DQN.

5. Hybrid Quantum-Classical Systems: QVMs, AlphaQubit-2, and Local Quantum Simulation

Quantum computing is becoming an integral part of advanced AI workflows, promising speed-ups for certain tasks. **Hybrid quantum-classical systems** combine classical processors (CPUs/GPUs) with quantum processing units (QPUs) to tackle problems. In a sovereign AI context, one may not have a physical quantum computer on-premises (as these are still rare and delicate), but can use **Quantum Virtual Machines (QVMs)** – simulators or emulators that mimic quantum computation – often accelerated by classical hardware. This section examines how QVMs fit into the Triple-Transformer pipeline, the role of advanced quantum algorithms like **AlphaQubit-2** and reinforcement learning methods (**VQR-DQN**) for resource orchestration, and the feasibility of performing quantum simulations on consumer-grade GPUs.

5.1 Quantum Virtual Machines (QVMs) and Hybrid Workflow

A **Quantum Virtual Machine (QVM)** is essentially software that emulates a quantum computer's behavior. Google's Cirq framework, Rigetti's Quil, IBM's Qiskit all come with simulators often called QVMs [116] [117]. The QVM allows developers to run quantum circuits as if on a real QPU, but using classical compute. QVMs are essential for development, testing, and even executing small-scale quantum algorithms when hardware isn't available.

In a QFaaS Triple-Transformer system, a QVM can be invoked at the Mapping stage (Transformer 2) if a sub-task is designated as quantum. For example, suppose our Analysis stage (CPU) determines that a certain optimization sub-problem could benefit from a quantum algorithm (like QAOA for a combinatorial optimization within a larger pipeline). It could offload that sub-problem to a QVM instance which runs on either CPU or GPU. Once the QVM produces a result (like the optimal bitstring solution), the classical pipeline resumes and the Validation stage checks that solution in the broader context.

Modern QVMs can leverage GPUs for speed using techniques like state vector simulation with massive parallelism. NVIDIA even has a platform (cuQuantum) with libraries to accelerate quantum circuit sim on GPUs. For instance, simulating 30+ qubits with full state vector (2^30 amplitudes) is memory-heavy (~8GB for double precision), but doable on a high-end GPU, and simulation speed can be 10-100x faster than CPU. Some QVMs use advanced math like *tensor network simulators*, which can simulate certain large circuits by exploiting structure, also GPU-accelerated [118].

Feasibility of local quantum simulation on consumer GPUs: It depends on the problem size. Up to ~30 qubits, a single GPU with enough memory can handle full simulation. Beyond that, memory demands grow exponentially if simulating arbitrary states. However, researchers have managed to push this by distributing across multiple GPUs or using approximate methods. A USENIX 2023 paper presented a "Quantum Virtual Machine" that allowed scalable execution of large quantum circuits via *gate virtualization* – basically slicing circuits into smaller pieces run sequentially or in parallel, improving fidelity and capacity [119]. In essence, they virtualize qubits somewhat like an OS virtualizes memory, to handle larger logical circuits on limited hardware [118].

For a sovereign AI practitioner, running a QVM locally for tasks like quantum machine learning or quantum optimization is plausible for moderate sizes. For example, an engineering firm might run a QVM to simulate quantum chemistry problems with ~25 qubits (maybe representing a small molecule) entirely on an AI workstation with an RTX 6000 GPU. Google's QVM via Cirq can emulate Sycamore processors (their quantum chips) – this is how they trained **AlphaQubit** (DeepMind's AI decoder for quantum error correction) on simulated Sycamore data [120] [121]. They generated "hundreds of millions of examples" using a quantum simulator to train AlphaQubit's transformer [122]. They likely used Google's cloud or supercomputers for that volume, but it shows that simulation was effective enough to feed an AI training pipeline.

A **Quantum Access Infrastructure Management (Q-AIM)** framework proposed in mid-2025 envisions something relevant: a lightweight containerized stack to host quantum hardware access on devices from personal computers to cloud servers [123] [124]. It specifically mentions running on "small personal computing devices" up to clouds, exposing a single entry point to quantum resources [125] [126]. This implies even local quantum *hardware* or simulated hardware can be integrated uniformly. Q-AIM's goal was to let research groups who get their own quantum hardware (or QVMs) operate them easily [127] [128]. For a sovereign AI workstation, one could deploy Q-AIM with a QVM backend to manage quantum tasks seamlessly.

Hybrid Orchestration: Running hybrid algorithms means the classical and quantum parts must alternate and share data. Many quantum algorithms are hybrid (variational algorithms like VQE, QAOA involve a classical optimizer updating quantum circuit parameters iteratively). In such cases, the CPU orchestrator (Analysis transformer) is heavily involved: it sets circuit parameters, calls QVM to get result, computes gradients or updates, repeat – effectively forming an inner optimization loop. This can be slow if the QVM is heavy, so techniques like parallelizing multiple circuit evaluations or using surrogate models can help. But

an emerging approach to optimize hybrid orchestration is using AI planning or RL to schedule quantum jobs – which brings us to **VQR-DQN**.

5.2 AlphaQubit-2 and Variational Quantum Reinforcement (VQR-DQN)

AlphaQubit-2 likely refers to a second-generation quantum-enhanced AI system following DeepMind's original AlphaQubit (2024). The original AlphaQubit was a *recurrent transformer* model trained to decode quantum error syndromes on Google's Sycamore processor, achieving 6% better error correction than previous methods [129] [121]. It was a big achievement in applying AI to quantum reliability. AlphaQubit-2 could plausibly extend this to dynamic error correction or control of quantum systems in real-time, or perhaps to different error-correcting codes or larger logical qubits. We don't have direct references to "AlphaQubit-2", but given the naming it's likely a refined model or approach that maybe uses **AlphaQubit + additional capabilities** (like reinforcement learning to actively choose error correction actions, or improved architecture scaling to bigger code distances). If AlphaQubit-1 was published in Nature 2024 [129], a hypothetical AlphaQubit-2 in 2025 could incorporate new ideas such as: - Integrating **quantum physics knowledge** (e.g., symmetry constraints) into the model training. - Adapting to hardware changes: maybe it's trained to generalize across different quantum chip calibrations or noise patterns. - Possibly coupling with small QVMs on GPUs to simulate corrections beyond current hardware and practice controlling them.

For our context, AlphaQubit-2 hints at using AI (transformers) to optimize quantum tasks. So in a sovereign AI pipeline, one might use an AlphaQubit-like model as an *agent to manage quantum resources* locally. For example, if one runs multiple quantum circuit simulations, an AlphaQubit-inspired agent could predict which circuits are likely to fail or have high error and adjust them or allocate more shots (repetitions) accordingly. This overlaps with **resource orchestration**.

VQR-DQN (Variational Quantum Rainbow DQN) is a concrete algorithm that blends quantum computing with deep reinforcement learning. According to a Dec 2025 preprint [130] [131], VQR-DQN integrated *ring-topology variational quantum circuits* with the Rainbow DQN algorithm to tackle an NP-hard resource allocation problem (human resource allocation) [130] [132]. Essentially, they replaced parts of a classical RL agent with quantum circuit components that exploit superposition/entanglement for richer function approximation. The results: VQR-DQN achieved notable improvements (e.g., up to ~13% better than classical Rainbow DQN) on those problems [133] [134].

This suggests a pattern: *using quantum-enhanced models (variational circuits) in reinforcement learning to schedule or allocate resources*. For QFaaS orchestration, one can imagine a VQR-DQN agent scheduling tasks across classical and quantum resources in a data center. In fact, Nguyen et al. (2025) pointed out that QFaaS (quantum FaaS) had been using mostly heuristics for scheduling quantum jobs across cloud backends [3] [135], and that more adaptive approaches (like DRL) were needed [4] [136]. VQR-DQN is exactly such an adaptive approach: modeling task scheduling as an MDP, with a hybrid quantum-classical agent finding policies that classical ones couldn't [130] [137].

Concretely, imagine a sovereign AI workstation with limited quantum simulation capacity (say it can simulate at most N qubits reliably at a time, or it can send small jobs to a cloud quantum service occasionally). It has to decide which tasks to run quantumly vs classically to maximize performance or fidelity. This is a scheduling decision under uncertainty (since quantum tasks have some probability of giving better results but also might fail if too noisy or large). A VQR-DQN agent could observe state (queue lengths, noise levels, etc.), and choose actions (schedule this task on QPU vs CPU, or in what order to run

jobs) to optimize throughput or accuracy. Over time it learns, potentially finding non-intuitive strategies that heuristics miss. The quantum part of the agent (variational circuit) might capture complex correlations in the state that help decision-making.

Local Feasibility of VQR-DQN: The paper's scenario was human resource allocation (like assigning police officers to events) [132], but the method is general. Running a small variational quantum circuit can be done on a local QVM – indeed, they used a ring topology circuit which could be simulated for the scale needed (and possibly even run on an actual small quantum device). The classical DQN part runs on CPU/GPU. So a sovereign setup could incorporate a VQR-DQN agent for optimizing internal scheduling of both classical jobs (like GPU tasks) and any quantum simulations. If one extended this idea, one might consider *Quantum-enhanced Transformers* for system management – e.g., a transformer controlling resources but with certain layers replaced by quantum layers to encode more complex patterns (some research explores quantum-inspired attention mechanisms [138] [139]).

Practical Local Quantum Simulation: Are we close to doing significant quantum computing locally? It depends. For algorithms that need dozens of qubits with entanglement, simulation gets heavy. But one strategy is to combine **partial quantum hardware** with simulation. For instance, **Tensor Processing Units (TPUs)** or FPGAs could possibly accelerate certain quantum circuit computations. There are also analog approaches (like photonic simulators). However, on commodity hardware, techniques like dynamic circuit cutting, approximation, etc., will extend reachable problem sizes gradually. Also, if a problem's quantum aspect is limited (like a variational circuit of depth D and certain structure), classical simulation can exploit that.

It's noteworthy that HPC centers are now building "quantum simulators" using classical supercomputers for up to 40–50 qubits brute force (with ExaFLOP machines, etc.). A local workstation won't reach that, but can still meaningfully use quantum algorithms at small scale.

Hybrid Example: Consider a chemistry simulation – trying to find ground state of a molecule (a typical quantum algorithm application). A sovereign AI system could use a quantum algorithm (VQE) with a QVM for the molecule's electronic structure, while using classical AI to guide the search or provide initial guesses. Maybe an ATHENA-like agent oversees this: one agent suggests an ansatz (quantum circuit structure), QVM runs it, another agent evaluates error, then they adjust – a bit like ATHENA's loop but in quantum chemistry. This saves classical compute possibly by leveraging quantum subroutines for the hard part (computing molecule energy), combined with classical intelligence for strategy (deciding which ansatze to try or how to update parameters). DeepMind's AlphaFold (classical) revolutionized protein folding; one could envision a future AlphaFold-Q that uses quantum submodules for parts of the physics, if/when quantum advantage appears.

However, a **realistic near-term view** is that local quantum simulation will remain a supportive tool (for design, validation, training quantum-savvy AI like AlphaQubit) more than a main workhorse for solving production problems better than classical. The QFaaS concept, though, is preparing for when small/medium quantum accelerators are standard parts of computing environments (like GPUs are today). In an intermediate scenario, maybe **Quantum Processing Units** become available as PCIe cards (some startups are exploring small quantum annealers or ion-trap on a PCIe board). Then a sovereign AI could literally have a QPU in the workstation. Q-AIM's notion of hosting your own quantum hardware means one might plug a 50-qubit cryo-free quantum module or an optical QPU into a desktop by late 2020s. In such a case, QVM and orchestrations we discussed seamlessly extend to actual quantum hardware usage.

Summary for Section 5: Hybrid quantum-classical sovereign AI involves: - Using **QVMs** to simulate quantum circuits within the AI pipeline [140] [118]. - Incorporating **AI models (AlphaQubit-like)** to manage or mitigate quantum errors, making quantum steps more reliable [121] [122]. - Employing **quantum-enhanced RL (VQR-DQN)** to optimally allocate tasks across classical/quantum resources [131] [134]. - Recognizing the limitations (simulate relatively small qubit counts) but leveraging them where beneficial. - Preparing for future local quantum hardware by building the integration framework now (as Q-AIM suggests, treat quantum resource as a service in your stack) [124] [128].

With hybrid computing tools in place, sovereign AI systems could tackle certain problems faster or with higher fidelity than purely classical ones, even before large-scale quantum computing fully arrives. In doing so, they remain *sovereign* – not reliant on cloud quantum services (which are emerging, like Amazon Braket or Azure Quantum) – which is good for organizations that cannot send sensitive workloads to external quantum providers due to security/regulation.

Next, we will examine open-source vs proprietary LLMs (Section 6) which is another axis of sovereignty (model weights availability, customization) and discuss aspects like FP8 training, mixture-of-experts, and safety alignment.

6. Open-Source vs Proprietary LLMs: DeepSeek, FP8, MoE, Safety, and Context Windows

Large Language Models (LLMs) form the backbone of many agentic AI systems. A key question for sovereign AI is whether to use **open-source models** (which can be run and modified locally) or rely on **proprietary models** (often via cloud APIs). In this section, we compare these model ecosystems across several dimensions: model capabilities (using **DeepSeek** as a representative open model series), training efficiency innovations (like **FP8 precision** and **Mixture-of-Experts (MoE)** architectures), safety alignment and the risk of **censorship leakage**, and the practical limits of **context window** lengths.

6.1 DeepSeek Models: Pushing Open-Source Frontiers

DeepSeek is a family of open-source LLMs that emerged from a Chinese company and garnered attention for achieving cutting-edge performance in 2024–2025 [141] [142]. The DeepSeek series (versions V2, V3, etc.) exemplifies how open models have rapidly advanced: - **Scale:** DeepSeek-V3 is a *Mixture-of-Experts (MoE) model with 671 billion parameters*, though only 37B are active per token (i.e., each inference uses a subset of experts) [88] [143]. This sparsely activated design allows a huge knowledge capacity without proportional increase in runtime – a technique pioneered by Google's Switch Transformers and GShard [144] [145]. - **Training Efficiency:** DeepSeek models have employed various efficiency tricks. The team's technical analysis (blogs by Jinpeng Zhang) outline "10× efficiency improvements" through methods like MoE, **Multi-Head Latent Attention (MLA)** to compress the KV cache, **Multi-Token Prediction** to generate more than one token at a time, **DualPipe** for overlapping communication and computation in multi-GPU training, and crucially **FP8 Training** [146] [147]. FP8 (8-bit floating point) training reduced DeepSeek's training memory and time significantly [148] [90] – by ~37% time and 42% memory in a Microsoft paper's experiments for 175B models [148]. DeepSeek adopted FP8 from version V3 onwards [149] [150]. This allowed them to train massive models on given hardware budgets, essentially putting open community within reach of model scales previously exclusive to companies with fleets of A100s. - **Performance:** According to DeepSeek's technical report and benchmarks, DeepSeek-V3 *outperforms other open models and is comparable to leading closed*

models on many tasks [81] [151]. Figure 1 in the report shows DeepSeek-V3 near GPT-4 level on various evaluation metrics [152]. They achieved this with careful data curation (14.8 trillion tokens of diverse, high-quality data) and multi-stage fine-tuning (including **Supervised Fine-Tuning and Reinforcement Learning** in post-training) [153] [154]. They also introduced an *auxiliary-loss-free load balancing* for MoE to maximize expert utilization without additional MoE losses [143] [88], and multi-token training objective for better performance [88] [153].

In short, DeepSeek showcases that open models can incorporate state-of-the-art research (MoE, FP8, etc.) quickly and achieve near parity with proprietary giants. Notably, they did all this with *2.788 million GPU hours on H800 GPUs* (comparable to A100) which cost an estimated \$5.6M [155] [156] – a large but not astronomical sum, suggesting a well-resourced team but within reach of national labs or coalitions of universities, not just a Google-scale entity.

For sovereign AI adopters, DeepSeek or similar models (like Meta's LLaMA2, MosaicML's MPT, etc.) provide an opportunity to run high-quality LLMs entirely on-premises: - DeepSeek-V3's 37B active parameters per token means its runtime is akin to a 37B model in speed, albeit memory usage is higher due to storing many experts (they mention special memory-saving strategies in Section 3.3 of report, like storing experts in low precision when not in use, etc.) [157] [158]. A 37B model can be served on a single high-end GPU with 48GB memory (especially with 4-bit or 8-bit quantization for inference). For training or fine-tuning DeepSeek, one might need multiple GPUs, but not unthinkable – 8×80GB A100 could likely fine-tune 37B active model on specific tasks. - If open source licensing allows (some DeepSeek versions were under a custom license requiring a usage request; but generally open models allow use within certain guidelines), an organization can *customize* it. They could fine-tune on their proprietary data, add domain-specific experts to the MoE, or prune out parts not needed. - Importantly, open models avoid vendor lock-in and allow internal audit. One can inspect the weights for biases, run reproducibility tests, and ensure no hidden instructions (whereas closed APIs might have hidden system prompts or filters one doesn't see).

6.2 FP8 Training and Other Optimizations

FP8 (8-bit Floating Point) Training – why is this significant? Because training LLMs is memory-bandwidth bound, and reducing precision from 16-bit (FP16/BF16) to 8-bit cuts memory and communication in half. NVIDIA H100 introduced FP8 tensor cores and transformer engine support [159] [160], enabling stable FP8 training. The challenge is that naive FP8 can lead to under/overflow and loss of precision for certain gradients or weights [161] [162]. The *FP8-LM* paper (Microsoft, 2023) solved this with techniques: - **Precision Decoupling:** Keep certain sensitive parts in higher precision (like layer norms, embedding weights, etc.) [161] [163]. - **Automatic Loss Scaling:** Dynamically adjust scale of tensors to keep values in representable range [161] [163]. DeepSeek adopted similar strategies for their FP8 training [164] [165] – they mention doing GEMMs (matrix mult) in FP8 for forward and backward passes, but still keep some parts in BF16/FP32 for stability (embedding layer, output layer, MoE gating, normalization and attention softmax likely in higher precision) [165] [166]. They also applied fine-grained per-128-element scaling to reduce outlier impact [167] [168] (tiling the matrix into 128-sized chunks for separate scaling factors). The results were dramatic: training a 175B model was 37% faster and 42% less memory vs BF16, with no accuracy loss [89] [90]. DeepSeek's own results presumably mirrored that (they cite FP8-LM and clearly were motivated by it).

For sovereign AI, FP8 means one can train larger models on given hardware, or conversely, fine-tune using half the GPUs. Some open libraries (NVIDIA's Transformer Engine in PyTorch, and bitsandbytes library for quantization) already support FP8 for certain operations on H100 and even some approximations on A100

(though A100 doesn't have FP8 hardware, one might simulate or use int8 quantization with calibration). As hardware moves forward, FP8 (and even 4-bit for inference) will become standard.

Mixture-of-Experts (MoE) – we touched on this with DeepSeek. MoE allows scaling model capacity (number of parameters) without proportional scaling in computation. E.g., DeepSeek-V3 with 671B params uses maybe ~37B *inference FLOPs per token, not 671B. MoE does this by having many expert submodels and a gating network that activates top-K experts per token [169] [170]. Google's Switch Transformer used K=1 (one expert per token) for simplicity and got big efficiency gains, though needed to solve load-balancing (some experts would get way more data than others if not encouraged to even out) [169] [171]. DeepSeek's innovations like auxiliary-loss-free balancing [172] [173] suggest they found a way to avoid the extra loss term and still balanced usage (maybe by batch-wise or sequence-wise balancing) [174] [175].

The takeaway: **Open models with MoE** can achieve *super high parameter counts cheaply*, which closed models historically didn't exploit as much in final products (OpenAI and others seemed to favor dense models for consistency or simplicity; though Google did have MoE research and products like GLaM). This means open community might overcome the "size" advantage of closed models. Indeed, DeepSeek-V3 at 37B active presumably contains knowledge competitive with dense 175B+ models because of MoE's larger total parameters (671B). This might be why they claim comparable performance to closed GPT-4 etc. The downside of MoE is more complex serving (need to handle sparse routing) and more memory (store all experts, though not all used at once). But frameworks like FastMoE, DeepSpeed-MoE exist to ease that.

For a sovereign AI user, running an MoE model is a bit more involved (one must run the gating and fetch appropriate expert weights). However, if implemented well, you might only incur say 2–3x memory overhead for the experts but keep compute similar to a smaller model. Systems with lots of RAM or SSD (maybe memory-mapped weights) could handle the storage of many experts, pulling in ones needed for a given query. If an organization has a specialized knowledge domain, they could even add new experts trained on that domain without affecting the rest of the model, an appealing modularity.

6.3 Safety Alignment and Censorship Leakage

Safety alignment refers to fine-tuning models with human feedback or other techniques (RLHF, constitutions, etc.) to ensure the model follows instructions helpfully and avoids harmful content. Proprietary models like ChatGPT, Claude, etc., have intensive alignment – often making them refuse disallowed queries or moderate outputs. Open-source models historically might be either unaligned (just raw pre-training) or partially aligned via smaller-scale instruction tuning or community RLHF efforts.

One phenomenon of interest is **censorship leakage**, which is when a model's internal biases or moderation rules inadvertently surface or affect outputs even in unintended ways. The phrase came up in context of DeepSeek: a 2026 study found that DeepSeek AI suppressed certain politically sensitive content in its outputs, despite that content appearing in its chain-of-thought reasoning [99] [100]. Essentially, DeepSeek would "internally think" about e.g. Tiananmen Square if asked, but not mention it in the final answer because presumably it had some alignment to Chinese censorship guidelines [176] [177]. This is a form of *institutionalized bias* – the model learned or was fine-tuned to avoid topics sensitive in China (transparency, democracy, etc.) [176] [177]. The researchers termed it an "information suppression apparatus" embedded in the model [177]. They highlight that when such a model is used outside its original environment, these hidden rules cause *censorship leakage*: the model might omit facts or skew answers even where that is not

desired [178] [179]. And because weights are open, any downstream user or model inheriting from DeepSeek would unknowingly propagate that bias if not audited [180] [179].

This underscores that **open-source doesn't automatically mean unbiased or safe** – it depends on how it was trained. DeepSeek's open release ironically contained state-aligned biases. On one hand, this is a caution for sovereign users: if you pick up an open model from somewhere, be aware of possible hidden alignment (be it political bias, or it might be encoded to refuse certain content). The audit by Qiu et al. showed a method: compare the model's reasoning vs response on sensitive vs non-sensitive prompts [181] [100]. They found semantic-level censorship – certain terms and criticisms were systematically dropped from answers on sensitive topics [99] [100]. They even note that "DeepSeek has been banned in some jurisdictions over privacy and security concerns" [182], implying it was recognized to have issues.

Censorship leakage can also refer more broadly to unintended filtering. Another scenario: if a model is fine-tuned to avoid hate speech, sometimes it might refuse benign queries that just contain certain keywords out of context. Users have noticed e.g. ChatGPT refusing medical info because it triggers some heuristic. Or the opposite: a model might have a secret list of disallowed outputs but under certain prompt conditions it reveals them (like the DAN jailbreak phenomenon where you could trick ChatGPT to ignore its content filter by role-playing, leaking its "true" uncensored self). That's leakage of alignment – the alignment not fully holding or it being circumventable.

Open vs closed approach to safety: Closed models often have reinforcement learning with human feedback (RLHF) to align with a set of guidelines. This yields a relatively *consistent style* – e.g., ChatGPT has a known voice (polite, a bit verbose, with certain phrasings) and a list of refusals. Open models, if just instruction-tuned on public datasets (like ShareGPT dialogues, OIG, Dolly dataset, etc.), have some alignment but may be less strict or polished. E.g., they might be willing to answer slightly more controversial questions, or they might not have as solid a refusal mechanism (leading to more likely to produce disallowed content if prompted). Also open models can be deliberately fine-tuned to be **uncensored** (some communities did that with LLaMA to produce models that ignore moral constraints – these exist on torrent more than official hubs due to obvious reasons).

For sovereign AI, safety alignment is double-edged: - You likely want an aligned model so it doesn't e.g. harass users or leak secrets or give obviously harmful instructions. But you also might want **control over the alignment**: maybe a government or enterprise has its own notion of what should or shouldn't be allowed (like an enterprise might want it to never give legal advice to avoid liability, etc.). - If you use a closed API, you are subject to their alignment decisions. Sometimes this caused trouble – e.g., some financial firms found ChatGPT refused to discuss certain policy issues they actually needed to analyze, because it triggered content filters ironically. Or in an educational context, maybe you want the model to discuss sensitive historical events bluntly, but the open model might try to sanitize.

Open models allow you to *re-align them to your needs*. This could mean fine-tuning on custom "preference data" or using a system like Anthropic's **Constitutional AI** where you can choose the principles the AI follows [183] [184]. For instance, Anthropic's Claude was trained with a fixed "constitution" of values (like respect, honesty, etc.), but one could imagine adjusting that if you had the model. With open models, one approach is to use **reward modeling and RLHF within your domain** – e.g., train a reward model on what outputs your company's policy considers ideal, then RL tune the LLM.

The risk of "censorship leakage" for sovereign AI is especially salient if you import a model aligned to someone else's regime. The DeepSeek example is cautionary: if a Western company naively used DeepSeek out-of-the-box, it might silently drop politically sensitive info even when asked for analysis on an international issue, giving a skewed view aligned with Chinese state narratives [182] [185]. That could lead to poor decisions. But if they know, they could fine-tune to remove that bias (maybe by including Q&A that emphasize transparency or simply by mixing in datasets from other sources).

Censorship vs Safety: It's worth distinguishing. "Censorship" might mean politically motivated suppression (like the DeepSeek case). "Safety alignment" might mean preventing clearly harmful advice (don't help someone plan violence). There is overlap in methods (both involve instructing the model to not say certain things), but the intentions differ. A sovereign user might want to *strip out any hidden political censorship* but keep the genuine safety constraints. Achieving that is non-trivial: one might need to carefully craft an alignment training that says "It's okay to discuss sensitive political topics factually, but do not produce hate speech or encourage violence." This requires nuance and possibly curated training signals.

One advantage sovereign teams have: **transparency**. Since you can inspect and even extract chains-of-thought as the DeepSeek audit did, you can identify when a model is holding back in reasoning. The study literally recovered suppressed sentences by prompting the model to output its chain-of-thought, which contained the content it omitted in final answer [186] [187]. That allowed them to identify what topics were being suppressed. A closed model doesn't give you its thought (unless you jailbreak or it inadvertently leaks instructions). So open models allow deeper auditing for both bias and safety issues, enabling more informed mitigation.

6.4 Context Window Capabilities

The **context window** of an LLM is how much input (and recent conversation) it can consider. Proprietary models have pushed this to impressive lengths: - OpenAI's GPT-4 has a 32K token version (and an experimental 128K token version tested by some users). - Anthropic's Claude-2 supports up to 100K tokens context.

Open-source models historically were limited (LLaMA 2 default 4K, some extended to 16K via RoPE scaling or fine-tune, Mistral 7B 8K by default). But recently, techniques to extend context for open models have proliferated: - **Positional Embedding RoPE extrapolation:** People found you can modify the rotary position embeddings to allow larger contexts (just extrapolate the rotation frequency). This works to some extent but model quality may degrade beyond certain length it saw in training. Some have done up to 32K or 64K with some quality loss. - **Fine-tuning on long context data:** E.g., Meta's **LongLLaMA** and projects like **GPT4All** did fine-tunes to extend context to 32K or more by training on long sequences so the model learns to utilize further positions properly. - There's also **RMT (Retrieval Memory Transformer)**, **SSM (State Space Models)**, and other research that can handle ultra-long sequences by design. These aren't widely deployed yet, but e.g. MISTRAL AI's research had a model with 1M token context using state-space layers for memory (though with some trade-offs) [188].

Notably, DeepSeek-V3 took context seriously: they did a two-stage *context length extension fine-tune*, first to 32K then to 128K tokens [189] [190]. Using a method called **YaRN (Yet another RoPE Extension)** [191], they performed 1000 steps of fine-tuning to extend from 4K to 32K, then another 1000 to 128K [191]. They evaluated it on LongBench and other tasks, showing it worked: DeepSeek-V3 achieved top-tier performance on long-context tasks and could handle documents ~100k tokens with only slight trailing behind GPT-4 on a

QA benchmark requiring 100k context [192] [193]. They even mention demonstrating the model on tasks requiring question-answering over 100k token contexts (FRAMES dataset), where it outperformed all models except GPT-4 (and was close behind it) [194] [193]. They also did a nifty trick: they ran a **two-stage** process to extend context: 1. Expand position embedding and fine-tune on 32K. 2. Then expand further and fine-tune to 128K. This gradual approach stabilizes training, as jumping to 128K at once might be too hard.

So open models have caught up in context length ability. For sovereign use, having large context is valuable: you can feed entire documents, large codebases, or multi-document bundles for analysis without chunking. Google's **NotebookLM** is an example where large context helps – it allowed up to 50 sources in a notebook (links, PDFs, etc.) [97] [98]; presumably their model (likely a version of PaLM) had the capacity to ingest all that. With open models, one could replicate that using a long-context fine-tuned LLM like LongChat-13B (OpenLLaMA fine-tuned to 16K) or DeepSeek-V3 for really long.

However, very long context usage comes with steep compute cost (self-attention is O(n^2) in sequence length). 128K context on a 37B model is enormous compute (~ (128k)^2 ~ 1.6e10 attention operations per layer, though MoE might restrict some of that by splitting among experts?). DeepSeek must use optimizations or accept slower inference for those cases. Possibly they use sparse patterns or some caching (not described, but maybe using something like MQA (Multi-Query Attention) helps memory overhead by sharing key/value projections [195] [196], DeepSeek did implement *low-rank key/value compression* from v2 onward [197] [198] which reduces KV cache size and presumably helps long context by not blowing up memory usage for KV).

Open vs closed context: Closed models still have advantage at extremely high lengths (Claude's 100k is proven; open, only DeepSeek claims 128k but not widely tested by outsiders). But the gap is narrowing as techniques spread. We might also see open models implement the likes of **Recurrence or Sliding Window** to effectively get unbounded context. E.g., **Attention with Linear Bias** or the recent **RWKV** model that is an RNN with the flavor of transformer, can handle streaming input of arbitrary length (but you might lose some info from far past if not explicitly stored).

One interesting open solution: **MCP + retrieval** can circumvent need for huge context. Instead of feeding 500k tokens raw, an agent can search and pick the 5k relevant ones to give to model. This often is more efficient and even more accurate if the retrieval is good, because the model doesn't have to read through irrelevance. So some say extremely large context might be somewhat overrated – beyond a point, retrieval-augmented strategy might be better. But having up to 100k can simplify some cases (like analyzing a long legal contract in one go without building an external search index).

From a sovereignty perspective, having models that can directly accept a user's large data (like whole manuals, logs, etc.) is very useful. If one had to rely on closed API with 4k limit, it might require sending piece by piece, which is slow and possibly error-prone. With open long context model, one could run it on their machine and feed entire logs or book and get an answer in one shot (provided memory allows).

Memory and hardware: Running a 128k context model indeed requires a lot of RAM/GPU memory. For instance, if using 16 bytes per token per head for keys/values, 128k*16 bytes=2MB per head per layer just for KV, scaled by number of heads and layers. If 40 layers and 32 heads, that's 2MB*40*32 = 2.56GB just for KV cache. Manageable, but borderline on a single GPU for large models. Possibly they offload KV to CPU memory or compress it (some techniques compress KV with product or low-rank as they did).

Anyway, we see open models and proprietary ones leapfrogging: open tackled MoE and FP8 first at scale in DeepSeek, closed had bigger context sooner. But by late 2025, the lines are blurry.

Secrecy vs openness: Another interesting angle is *censorship leakage in closed models via context window*. If a closed model has hidden rules (like ChatGPT has a hidden system prompt "You are ChatGPT, follow OpenAI rules..."), large context and clever prompts can sometimes surface those. E.g., people managed to get GPT-4 to reveal its system message by asking it to roleplay or by string matching hacks. That's a form of the model leaking its safety instructions when exploited. In open models, there is no hidden prompt unless you put one – you have full control. So ironically, closed models might be more vulnerable to certain prompt injections because they carry developer-provided hidden contexts that can be manipulated if the model isn't robust enough to protect them. Open models, if run locally, you often just give user and maybe a short system prompt. Tools like **MCP** also complicate context (multiple sources injecting data), making prompt injection a big web security concern: e.g., if an agent reads from a website and the site has malicious prompt text, it could trick the agent. That's something both open and closed have to handle by sanitizing inputs or limiting model autonomy.

Model comparisons beyond DeepSeek: DeepSeek is one, but also: - Meta's **Llama 2** (July 2023) which is widely used (7B, 13B, 70B variants) and came with a fine-tuned "Chat" version. Many derivatives on that. - **Mistral 7B** (Sept 2023) – small but very high quality for size, and built with multi-scale contexts. - **Falcon 40B** (2023) – another open model, which had good performance. - New 2025 open entrants might be MosaicML's 70B, JiuZhang, etc.

Proprietary LLMs around now: - OpenAI GPT-4: still likely the best general model (as per evaluations and usage). - Google's Gemini (expected late 2023/2024, presumably a multi-modal model with possibly >1T parameters). - Anthropic's Claude 2 and upcoming iterations (they focus on making it follow instructions and have more knowledge). - Others like Microsoft's internal Orca or Inflection's Pi.

Open models are closing the gap in many benchmarks (as noted by DeepSeek's arXiv showing near parity). One open advantage: **no usage restrictions**. GPT-4 or Claude might refuse certain enterprise-specific tasks (e.g., "help me plan a strategy to beat competitor X" might be considered a *conflict-of-interest or illicit behavior? Possibly not, but certain personal data tasks might be refused due to policy). Open models will do whatever they are prompted, unless you restrict them. For better or worse, it's a tool that obeys the operator.

This means risk of misuse is higher if not controlled – an organization must implement its own usage policies and maybe fine-tune the model to refuse truly harmful requests by their employees (some companies might actually want the model to sometimes refuse if an employee tries to generate harassing language, etc., as part of internal ethics). So if you deploy an open model in a company, you might still apply some safety fine-tune or a monitoring layer to catch obviously problematic outputs (like filter out slurs, etc.). Proprietary providers often have such filters server-side (OpenAI monitors and will ban if used for disallowed content at scale). Sovereign use places that responsibility on the user organization.

Censorship leakage example in multi-culture: The BankInfoSecurity article describes how various AIs align to different "realities" – Western model vs Musk's Grok vs DeepSeek vs an Arabic model – each could give different answers on contentious issues, reinforcing their aligned worldview [199] [85]. If each group uses their own AI (as sovereignty implies), we risk a "loss of shared reality" [10] [200]. People trust AI outputs and may enter echo chambers validated by "their truth" AI [201] [202]. This is a socio-cultural challenge:

alignment vs fragmentation. Section 10 touches more on that, but from a technical standpoint, it means a sovereign AI might be intentionally tuned to one viewpoint. That's fine for internal consistency, but one should be aware of how that interacts with global discourse. It may put burden on users to practice critical thinking across AI outputs (the article concludes that – we must cultivate critical thinking and engage multiple perspectives, not treat AI as oracle) [184] [203].

To conclude Section 6: - **DeepSeek and similar open LLMs** show that open models can achieve top-tier performance with advanced training techniques (FP8, MoE, RLHF). - **FP8** drastically cuts training cost, enabling even local fine-tunes of moderate models (on H100 or future GPUs) with far less memory – relevant if an organization collects say a few hundred GB of domain data and wants to fine-tune a 70B model; with FP8 maybe they do it with fewer GPUs. - **MoE** offers a way to have huge knowledge capacity – possibly letting one model handle multi-lingual or multi-domain better by having experts – something proprietary models also do internally (like mixture-of-experts might be part of Google's Gemini or others, though not confirmed). - **Safety alignment** for open models is a customizable aspect – advantage is control, disadvantage is more responsibility. Censorship biases can exist and need auditing if you adopt models from different origins. - **Context length** no longer strongly differentiates closed vs open – open can reach 100K with careful fine-tune as proven by DeepSeek-V3 [190] [204]. This means tasks like ingesting large texts can be done locally. - **Quality**: While GPT-4 still likely has an edge in complex reasoning or very knowledge-heavy queries, the gap is narrowing. A KKR analysis noted DeepSeek's surprising leap as an open model and how it "surprised much of the community" [141] [205] – it indicates we could soon have open models that only slightly lag the absolute best.

So from a sovereignty perspective, it is increasingly viable to use open LLMs without a huge performance hit, and in return you gain transparency, customizability, and lower marginal cost (no per-query fees). The next section (7) will look at use cases leveraging these AI capabilities in creative and enterprise scenarios.

7. Use Cases in Creator Economy and Enterprise

AI systems like QFaaS Triple-Transformer agents and advanced LLMs are being applied across a wide swath of domains. Here we survey several prominent use cases, focusing on how **sovereign AI** (locally controlled, customized AI) can drive value in the **creator economy** (individual content creators, developers, artists) and in **enterprise settings** (business operations, compliance, DevOps, etc.). We examine examples including AI-driven DevOps automation, intelligent compliance and legal assistance, video content generation and editing, and AI research assistants like Google's NotebookLM for scientific discovery.

7.1 AI-Driven DevOps and Software Engineering Automation

In software development and IT operations, AI tools are transforming how code is written, reviewed, and deployed: - **DevOps Automation:** Agents can generate configuration files (Kubernetes YAML, Terraform scripts), detect issues in CI/CD pipelines, and even manage infrastructure by translating high-level directives into cloud API calls. For instance, an AI agent could monitor system logs and automatically adjust scaling or restart services when anomalies are detected, essentially acting as a junior SRE (Site Reliability Engineer). Harness's engineering blog evaluated leading LLMs on tasks like generating CI pipeline configs from descriptions and found they can **automate pipeline creation** effectively [206] [207]. Using an internal knowledge base of system architecture (perhaps via MCP integration with monitoring tools), an agent could respond to alerts: e.g., if database CPU spikes, the agent (with context of known incidents) suggests or executes adding an index or scaling read replicas. - **Code Generation and Review:** Beyond GitHub Copilot's

code suggestions, autonomous agents (as in Antigravity IDE or Microsoft's experiment with "Athena" in Teams [208]) can take on tasks like writing whole functions given a spec, or reviewing a merge request for bugs and compliance. Microsoft's dev blog describes an internal Athena agent that collaborates in Teams chats to assist developers [209] [208]. A sovereign instance of such an agent could be fine-tuned on the company's code style and guidelines, ensuring generated code and reviews follow internal best practices. Crucially, running it on-prem means proprietary code never leaves the environment (a big concern that has slowed adoption of cloud-based AI coding tools in some companies). - **Incident Response and Knowledge Management:** DevOps also involves documenting processes, writing runbooks, etc. An agent can automatically generate documentation from code or vice versa (update code based on documentation drift). It can also answer developers' questions about legacy systems by searching internal docs (like an always-on StackOverflow that's company-specific). NotebookLM (see §7.4) is one example geared for research notes, but similar ideas apply: feed all design docs and tickets into an AI that developers query for "why was X designed this way?" or "how to deploy Y service," reducing time digging through Confluence or SharePoint.

For sovereign AI in DevOps, an advantage is integration with actual systems. A cloud AI might suggest pipeline steps but not have access to run them. A local agent, however, can be given credentials (with careful sandboxing) to actually execute commands. For example, via Conductor's MCP integration with Orkes (workflow engine) [60] [59], an AI could trigger a Jenkins build or a Kubernetes deployment. One Medium article touted using an LLM to analyze environment configs for consistency or to generate Terraform that reflects a textual description of an architecture [210] [211]. That was likely done by manually copying context, but an integrated sovereign agent could do it seamlessly.

Case Study: *LADs: LLM-driven DevOps* (from arXiv 2023) proposed a framework using LLMs to ensure robust cloud ops [212] [213]. They emphasize the LLM's adaptability and some early successes in cloud management tasks. Another piece (LangChain for DevOps by Pulumi) discusses how GenAI can unify Dev, Sec, Ops tasks by auto-generating configuration and even doing security scans on infrastructure as code [214]. All these indicate a strong movement to incorporate AI in the development lifecycle.

Enterprise readiness: A Nutanix survey found 85% of orgs repatriating workloads on-prem for cost control [79], and specifically around AI, many are building "LLMOps" discipline (like MLOps but for large language models) [215]. This includes pipeline for model updates, monitoring output quality, etc., which sovereign control aids (as opposed to being at the mercy of an API's version changes).

7.2 Intelligent Compliance and Legal Assistance

Compliance (ensuring that business processes and documents meet regulatory and policy requirements) is a labor-intensive area ripe for AI: - **Document Analysis:** Enterprises have voluminous policy documents, contracts, and regulations to follow. AI can quickly parse these and highlight relevant sections or check for compliance issues in other texts (e.g., does this marketing material comply with financial advertising rules?). For example, an AI could scan a set of contracts and identify clauses that deviate from company standard or from new regulatory requirements – a task traditionally for legal teams. SiliconFlow's benchmark compared open legal LLMs like BloombergGPT, LegalLLaMA, etc., showing that specialized open models can excel in contract analysis [216] [217]. - **Question Answering on Regulations:** Instead of an employee reading a 300-page GDPR document, they can ask the AI questions ("Can we store user IP addresses?") and the AI, given the regs and internal policies, provides an answer with citation to the relevant section. Tools like **Regology** are building purpose-built LLM solutions that ingest laws and let users query in natural language [218] [219]. A

sovereign AI approach might ingest all laws relevant to the company plus internal rules, enabling instant Q&A without exposing data to external providers. - **Automated Auditing:** Continuously monitoring communications or transactions for compliance breaches. E.g., an AI agent could monitor employee email or chat (with appropriate privacy considerations) to flag if someone is sharing sensitive data externally, aligning with internal compliance rules. Many banks and firms already use simpler rule-based or keyword systems for this; an LLM could understand context better (for example, telling apart a harmless mention of a project codenamed “Super Secure” vs an actual leak of something confidential). - **Legal Drafting and Review:** This includes generating first drafts of NDAs, contracts, or compliance reports, which humans then refine. Also summarizing legal changes: if a new law is passed, AI can summarize what it means for the company’s policies. The *MIT Law Review article* calls this era the "Dawn of AI in Compliance" and discusses how LLMs follow structured reasoning to verify compliance [220] [221]. That suggests using LLMs to trace through checklists or logic trees that compliance officers use, which is feasible by chain-of-thought prompting or fine-tuning LLMs to output those intermediate steps (some initial research shows LLMs can mimic multi-step legal reasoning if trained appropriately). - **Case Outcome Prediction:** More for legal teams, but AI can predict likely outcomes of litigation or regulatory approval by comparing with past cases. That helps compliance teams advise on risk (though this probably remains supplemented by human judgment).

A salient example is handling of personal data with privacy laws. A compliance AI could be given an export of company databases (fields and usage) and relevant laws (like GDPR, CCPA) and then asked “Which of our data practices are not compliant with GDPR?” It would identify potential issues (maybe storing data without consent or not deleting upon request), citing the law [222] [218]. It could even suggest remedies (like “implement a consent flow for X”).

Why sovereign AI is beneficial here: compliance data is highly sensitive. Many organizations wouldn’t trust uploading all their contracts or incident reports to a third-party AI service. By using a local model, they maintain confidentiality. For instance, a **merge.dev** article mentions MCP enabling connecting AI directly to company data sources [223] [224] – imagine an internal compliance MCP server that feeds the AI with policy docs and logs on request, all within firewall.

Also, **multi-agent** usage appears: one agent can retrieve relevant reg text (via MCP to a reg database), another agent (the reasoning one) uses it to analyze an internal document. Agent communication (through a safe channel) means the reg text is only used for that task and not retained by model (for confidentiality of the reg database if needed).

Limitations and biases: LLMs may incorrectly interpret laws or be too lenient/strict if not tuned well. It's crucial they provide rationale and evidence (citations) – something we can enforce via prompting (like “always cite regulation text for your conclusions” – akin to how NotebookLM works giving source references [225]). Also, legal decisions often hinge on subtle definitions and context – AI might oversimplify. So often these tools are pitched as assistants to compliance professionals, not replacements. That being said, they can drastically cut the time to find information. There’s mention that open legal models (like an open version of Harvey AI used in law firms) do fairly well in summarizing and checking but still require lawyer oversight due to occasional errors (especially if the input facts are tricky).

7.3 Video Content Creation and Automation

The **creator economy** – individual content creators on YouTube, Twitch, TikTok, etc. – is seeing AI help generate and edit content, which can also benefit enterprises for marketing and training videos. Key AI applications in video: - **Automatic Video Editing:** As per the University of Toronto's **LAVE** system, LLM-powered agents can assist in editing tasks: summarizing raw footage, suggesting storyboards, retrieving relevant clips, trimming segments, and even applying transitions or effects based on text commands [226] [227]. LAVE (Language-Augmented Video Editor) allowed a user to say "make a travel vlog of my Paris trip" and the agent planned the sequence (using footage categories, story ordering, etc.) [228] [229], then executed edits (like trimming to only scenes with Eiffel Tower for a section) [230] [231]. The user could approve each step. This greatly lowers skill needed to produce a polished video. A sovereign version of this could be integrated into local video editing software (like a Final Cut Pro plugin that runs an agent on the workstation, without uploading footage to cloud). - **AI-Generated Video and Automation:** On the more generative side, models like **Runway Gen-2** or Meta's **Voicebox** (for audio) allow generating new video or doing complex editing by text. E.g., you can generate B-roll scenes or replace backgrounds with AI. A workflow might be: LLM agent writes a script -> text-to-video model generates rough scenes -> another agent or model refines or stitches them. For now, generative video is limited in fidelity and length, but improving. Still, for content creators, AI can already generate subtitles automatically (Whisper ASR), suggest titles/hashtags, and even make short highlight clips from a long stream (some AI tools auto-detect exciting moments for e.g. gaming streams to cut into a montage). - **Video Q&A and Search:** If an enterprise has a lot of recorded meetings or training videos, an AI can let employees query them. E.g., "Find the segment in the town hall where the CEO mentions Q3 revenue." An AI can have transcripts via ASR and use LLM to match that query semantically and return the video timestamp. Possibly it can even answer questions directly ("What are our sales goals?" answered by pulling the relevant clip where those were stated). - **Personalized or Interactive Video:** Agents like those in LAVE point toward interactive editing, but also interactive media content. For example, one could have an AI-driven NPC (non-player character) in a game that uses local LLM for dialogue and video generation for expressions. Or personalized training videos where an AI instructor addresses you by name and adapts to your progress – possible with generative video and local models (though requires a lot of compute for smooth real-time video generation, likely a near-future case as GPUs get stronger).

Sovereign edge for video creators: Many creators are independent and may not afford pricey cloud services, plus might want to keep their footage local until ready (raw footage can be hundreds of GBs). A local AI editing assistant that works offline is ideal. Tools like Descript (for transcript-based editing) or Adobe's Sensei features (like auto-reframe, color match) are somewhat cloud-tied; open or local AI could replicate those. LAVE's user study found it helped novices and even experienced editors, but users still wanted control and were wary of giving full autonomy [232] [233]. Mixed-initiative (as they propose) is key.

Enterprise use of video AI: beyond marketing content, companies have training libraries and recorded meetings. Summarizing meetings into short videos, creating training compilation from multiple videos, or even generating synthetic how-to videos from text instructions (using avatars like Synthesia does, but that's cloud-based). A sovereign company might prefer to generate internal training videos with an on-prem avatar model for confidentiality. That requires multi-modal AI (text-to-speech and some kind of visual avatar). Open-source projects exist for lip-sync (Wav2Lip) and text-to-speech (Coqui), so a pipeline can be built, though probably not as seamless as commercial yet.

Video automation agent example: A media company could set up an agent to ingest their daily news broadcasts, then automatically cut a 1-minute social media promo out of it. Steps: transcribe broadcast (ASR) -> LLM picks key quotes -> uses edit API to cut those segments -> overlays captions (maybe stylized) -> outputs final clip. All these can be orchestrated with something like Conductor or just a custom script. It's plausible entirely locally if one has the software libraries.

Data center angle: Video processing is heavy on storage and GPU usage. If done at scale on-prem, ensures privacy (no cloud uploading thousands of hours of possibly sensitive footage), but one needs robust hardware. Perhaps use idle GPU cycles at night to render auto-edits.

Trends: According to a Medium blog [234], 2025 trends include multimodal LLMs aiding video generation from text. The lines between image, video, text models are blurring (e.g., GPT-4 can analyze images now, future ones likely handle video frames). This means a sovereign multimodal model could one day take raw video frames as input in context and output an edited timeline or summary directly.

7.4 AI Research and Discovery Platforms – e.g. NotebookLM

Google's **NotebookLM** (introduced mid-2023 as Project Tailwind) is an AI-powered notebook intended to help researchers and students synthesize information from their notes and sources [225] [98]. It's a prime example of augmenting intellectual creative work: - **Contextual Q&A:** You load your documents (e.g., lecture notes, research papers) into NotebookLM, and it allows you to ask questions about them. The LLM can reference across all provided sources (50 source limit at launch [97]) and generate answers or new insights, with citations to the specific pages used [225] [98]. This is essentially retrieval-augmented generation done in a UI friendly way. For a student, this means they can ask "Compare the theories of X and Y from these papers" and get a synthesized answer referencing each paper's stance. - **Idea Generation:** NotebookLM is pitched as a "thinking partner" – you can prompt it to brainstorm research ideas or outline a literature review using your sources. It reduces the grunt work of gathering related points from many documents. - **Note Generation and Summarization:** It can summarize a long paper into key bullet points. Or take several papers and produce a combined summary. And because it's in a notebook interface, users can edit or refine the AI's output, then perhaps ask the AI to expand on an edited point – an iterative human-AI collaboration.

For sovereign AI, one could build a similar internal system: feed it all company technical reports, market research, etc., then employees can rapidly extract knowledge or get summaries. Similar to how NotebookLM now supports adding Google Drive files, images, Sheets, etc. [235], an internal variant could plug into internal document repositories and data sources via MCP connectors.

One might ask: why not just use cloud NotebookLM? It's Google-run, presumably using PaLM-2 or soon Gemini LLM. But companies or researchers handling sensitive or proprietary documents may not want to upload them. A local "ResearchLM" could use an open model like Llama-2 70B and an efficient vector DB like FAISS to replicate core functions offline.

Also, a specialized field might benefit from a fine-tuned model: e.g., a pharma company might fine-tune an LLM on internal R&D reports to better answer very domain-specific questions than a generic model would. They can integrate that in a similar Q&A interface.

Athena vs NotebookLM: Interestingly, the ATHENA paper (Section 3.3) is about an agentic autonomous lab making new discoveries in numerical algorithms [68] [67]. NotebookLM is more about assisting human discovery by organizing and synthesizing known info. Both point to AI accelerating knowledge work: - ATHENA shows agents generating new knowledge (like formulas, solutions). - NotebookLM shows summarizing and connecting existing knowledge.

Combined, one can envision a future research assistant that not only tells you what's in the literature (NotebookLM style), but can also simulate experiments or suggest new hypotheses (ATHENA style).

For example, in scientific discovery, ChatGPT plugins allow running code or retrieving data; an advanced agent might automatically run small experiments (like pulling data from a public source or doing a quick simulation) to answer a question more concretely. At that point, it's practically a collaborator.

Enterprise knowledge management: Many companies dream of an "ask the company brain" tool. NotebookLM is the consumer analog for personal knowledge. Enterprises have tried wikis, intranets, etc. to capture knowledge, but employees still waste time searching or duplicating work. A sovereign AI that indexes everything and answers queries in natural language (with references for trust) could be transformative in productivity. This has to be local mainly for privacy and because it involves internal data not suitable for public LLMs. Some vendors (like Microsoft with Copilot for Enterprise or IBM's Watsonx) offer solutions where the LLM runs on Azure or IBM Cloud but can connect to your data with promises of privacy. However, truly sovereign means running on the company's own hardware (some banks insist on that, e.g., they are exploring private GPTs on their servers rather than cloud).

Biases and accuracy: One must consider that LLM answers can sometimes be confident but wrong (hallucinations). NotebookLM tries to mitigate that by forcing citations – the idea is if it can't find support in sources, it should not fabricate. This works to an extent, but an LLM might misattribute or quote out of context. That's why user oversight is needed. However, a well-tuned system with retrieval should have a lower hallucination rate – it largely uses actual text from documents to build answers. It might still be biased by the sources given (if sources all have one viewpoint, answer will reflect that – but that might be fine if those are the authoritative sources).

Cultural Implications: The ease of getting answers could change how people learn. Instead of reading 5 papers fully, a student might lean on the AI summary. This raises concerns about losing nuance or critical reading skills. In enterprise, employees might rely on the AI's summary of policy rather than reading it, possibly missing context. Over time, perhaps these AI assistants themselves will become subject to compliance or quality checks (e.g., a regulatory body might require that internal AI knowledge systems log how they derived an answer to ensure no misinformation is spread in critical contexts).

Creator economy can benefit similarly: YouTubers reading complicated tech docs can use such an assistant to simplify and distill them for content. Writers can have an AI research assistant gather facts or even check consistency across their notes.

Future directions: Multi-modal NotebookLM – currently it can handle text and images (embedding images and likely doing OCR or description). That helps when sources are textbooks with charts – it might incorporate chart info into answers. Possibly integrate audio transcripts too. A sovereign version could link to internal videos (like recorded meetings) as sources.

Overall, the **use cases illustrate that sovereign AI** isn't just about one chatbot – it's about embedding AI throughout workflows: writing code, ensuring compliance, creating media, and generating knowledge. The Triple-Transformer architecture with integrated optimizations, agent frameworks, multi-agent protocols, quantum backends, and powerful local models is the backbone enabling all this.

Next, in Section 8, we step back to examine macro-level infrastructure factors that might affect deploying these systems in practice: data center capacity, power consumption, deployment models (on-prem vs hybrid), and the significant investments tech firms are making in energy and compute to fuel AI – which all influence the feasibility and strategy of maintaining "sovereign" AI compute capabilities.

8. Macro-Infrastructure Trends: Data Center Capacity, Power, Deployment Models, Energy Investments

The rapid growth of AI has macro-scale implications for computing infrastructure. Running advanced AI systems – whether cloud or sovereign – demands massive compute and energy. In this section, we examine trends and challenges in the infrastructure underpinning AI, including **data center scarcity**, escalating **power demands**, the shift between **on-premise vs hybrid deployments**, and how tech companies are investing in energy and new data centers to support AI expansion.

8.1 Data Center Capacity and Scarcity

AI workloads (especially training large models and serving many queries) require data center resources – racks of GPUs, high-speed interconnects, cooling, etc. Over the past couple of years, there has been a notable **shortage of data center capacity** in certain regions due to the AI boom. Key points: - **Physical Space and Construction:** Many major cities (e.g., Silicon Valley, London, Dublin) have limited land/power to add new data centers. The lead time to build or expand is long (18-24 months). Meanwhile, demand is spiking now. This has led to scramble for capacity – e.g., in Virginia (the largest data center market in US), companies are leasing entire new builds often before they're finished. A Deloitte analysis listed "7 shortages challenging AI data center expansion" – likely including land, power, cooling equipment, networking gear, skilled labor, and crucially **accelerator hardware (GPUs)** [236] [237]. - **GPU Supply Crunch:** 2023 saw severe GPU shortages (NVIDIA H100s on backorder for many months). Cloud providers like Azure and Google Cloud had to limit availability of high-end GPU instances, and some AI startups reported long wait times or inflated costs to get capacity. Microsoft CEO Satya Nadella revealed in late 2025 that they actually have "thousands of AI GPUs sitting in inventory" but can't deploy them due to lack of available **power and cooling** in data centers [238] [239]. Power, not chips, became the bottleneck by then [240] [241]. Microsoft had 2GW of new capacity in 2025 but still fell short of AI demand [242] [243]. - **Power and Cooling Limitations:** Traditional data centers often provision ~5–10 kW per rack. AI gear like GPU pods can draw 30–50 kW per rack or more. Many existing facilities cannot handle that without upgrades. This leads to "warm shells" problem – you have space (shell) but not the power infrastructure to utilize it [244] [240]. That's why Microsoft had GPUs idle; they physically had them but no "warm shells" to plug into [240]. - **Lead times for Utility Power:** In some areas (e.g., London's West region), power grid constraints led to moratoriums on new data center hookups. One report said parts of UK wouldn't have new big power capacity until 2030 due to grid limitations. Data centers cause local stress – e.g., in Ireland and Singapore, governments paused approvals out of concern for national grid. - **Energy and Water Usage:** Data centers now account for an estimated 4% of US electricity usage (in 2024) [245] [246]. Pew Research projects a >100% increase by 2030 [247] [247], which correlates with AI demand. Water usage is also huge (cooling); a mid-size DC can use millions of gallons

daily [248] [249], raising concerns in drought-prone areas. E.g., a 1 GW data center campus might require tens of millions of liters of water a day for cooling [250] [251].

In short, **physical infrastructure scaling is struggling to keep up** with AI's exponential compute appetite. For sovereign AI, this is relevant in two ways: - Large organizations wanting to self-host must plan for similar constraints. If a bank wants its own AI supercomputer, can it provide the space and power? Some are building their own (JPMorgan reportedly investing in in-house GPU clusters). - If sovereignty means not relying on public cloud, there might be an upper limit to how much compute one can realistically deploy on-prem without significant lead time. This might push people to consider hybrid or more efficient usage (like using smaller models or optimizing workload distribution). - It also suggests that *efficiency improvements (like FP8 we discussed, or algorithmic optimizations)* are critical to mitigate these constraints. If every org tried to replicate something like GPT-4's training from scratch, it could be infeasible due to these capacity issues. Sharing/trading-off compute might become a factor – maybe sovereign but collaboratively built models (like multiple companies pooling resources to train one model they all use to avoid duplicative loads – akin to how open research models are done by consortia).

8.2 Power Demand Growth and Energy Initiatives

Power consumption for AI is massive and growing. Some stats and responses: - **Energy Use Doubling:** IEA reported data centers used ~220 TWh in 2024 (4% global electricity) and projected a 130% increase to ~500 TWh by 2030 under current trends [247] [247]. That's enormous – roughly equivalent to another mid-sized country's consumption. This is driven largely by AI and cloud (traditional enterprise DCs have plateaued or become more efficient). - **Efficiency Gains vs Load:** Historically, data center energy use was relatively flat 2010-2020 due to efficiency gains (virtualization, better cooling, etc.) even as compute grew. But AI is breaking that – GPU farms don't follow same efficiency curves yet, plus the computations needed are skyrocketing (OpenAI famously noted a 2-year doubling time in compute used for largest models, outpacing Moore's Law). - **Enterprise vs Hyperscaler:** Hyperscalers (the big cloud firms) are investing heavily in renewables and advanced cooling (like immersion cooling, etc.). They can optimize at scale, whereas smaller data centers may be less efficient or have older infrastructure. This could become a centralization vs decentralization debate – is it more energy-efficient overall to have one giant super-efficient AI park vs many scattered smaller ones? Possibly the giant one is better (economies of scale in cooling, etc.). But then again, distribution reduces transmission losses and tail latencies. It's a trade-off. - **Tech Firms and Energy Projects:** Several tech giants are directly investing in power generation: - **Nuclear and SMRs:** Microsoft signed a 20-year deal to buy 835 MW from a revived nuclear plant (Three Mile Island) by 2028 to power AI data centers [252] [253]. They're also exploring small modular reactors (SMRs) for DCs [254] [255]. Deloitte predicted nuclear could supply 10% of DC electricity by 2035 [256] [256] as decarbonization and base load needs push it. - **Renewables:** Google achieved ~66% carbon-free energy for its DCs in 2022 and aims for 24x7 100% carbon-free by 2030. They invest in solar, wind farms near their centers, and use PPAs (power purchase agreements). Amazon, Meta, etc. likewise are among the largest corporate buyers of renewable energy [257]. - **On-site Generation:** Some DCs now have on-site gas turbines (or even repurposed jet engines) for quick power addition [258] [259]. That one example: ProEnergy using jet engines to supply DCs amid gas turbine shortages [258] [259]. This speaks to creative ways to overcome grid lags: basically mini power plants at data centers. - **Grid Impact and Regulation:** Governments are noticing AI's strain. E.g., US DoE working with companies to plan power expansions [260] [261]. In some places, data centers are taxed or their growth limited to ensure local grids serve residents first. There's talk of requiring waste heat reuse or water saving measures (like using wastewater for cooling to conserve potable water). - **Edge vs Centralized**: There's also a concept of *distributed compute* – maybe not all AI inference should happen in

giant DCs. Smart edge (like running smaller models on user devices or on-prem edge servers) can reduce central load. For sovereignty too, local inference (even at a city or building level micro-data center) can cut latency and power needed for transmission/cooling overhead in big DCs. However, edge devices might be less efficient individually, so it depends on scenario.

From a sovereign perspective: - Running large language models 24/7 can be power-hungry: a single A100 40GB card at full utilization draws ~300W. A cluster of 8 is ~2.4 kW, which over a year is ~21 MWh (and potentially double that if cooling overhead ~1:1). That's tens of thousands of dollars in electricity annually. Companies have to weigh that vs cloud costs (cloud includes electricity in their price, plus margin). - If many orgs start hosting their own AI, overall power usage could decentralize from hyperscalers to enterprise data centers, which might be less optimized in PUE (Power Usage Effectiveness). Hyperscalers brag PUE ~1.1 or less (meaning 10% overhead for cooling). Many smaller corporate DCs have PUE 1.5-2.0 (50-100% overhead). So ironically, a shift to sovereign AI could increase total energy footprint unless those orgs invest in efficiency. It's an argument some use to favor cloud for sustainability. But sovereignty might push enterprises to modernize their facilities (some might co-locate in efficient colocation data centers to run their racks, rather than in an old server room). - The scarcity of GPU supply and data center capacity suggests that *collaboration or prioritization* might become needed. We might see government stepping in to allocate or incentivize critical AI (like related to healthcare, climate) vs less critical (like generating endless AI content). Already, some HPC centers create allocation committees for limited resources – maybe something akin could happen in industry if demand far outstrips supply.

8.3 On-Premise vs Hybrid Deployment Models

The pendulum in enterprise IT often swings between centralization (mainframes->cloud) and decentralization (client-server->edge). Right now, cloud adoption is high, but **hybrid cloud** (mix of cloud and on-prem) is recognized as the dominant strategy for large enterprises, especially for AI: - A 2024 Barclays CIO survey (cited by Infinidat) found **83% of enterprises plan to repatriate** workloads to private cloud from public [262] [263]. Similarly, Nutanix reported 85% repatriation intentions for some workloads [79]. This doesn't mean abandoning cloud entirely, but moving things where it makes sense cost-wise or control-wise. - For AI specifically, reasons to keep on-prem: data sensitivity, regulatory compliance (e.g., European banks wanting to keep data under EU jurisdiction, etc.), and ironically, cost stability. Cloud GPU instances are expensive (can be $2-3/hour for one A100). Owned hardware, if utilized fully, can amortize to lower costs (though initial capex is high). - However, on-prem requires talent to manage (MLOps, etc.) which some companies lack. The shortage of AI/DevOps skills can push smaller companies to just use cloud managed services.

Hybrid Approach often emerges: e.g., train large model on cloud where scaling to thousands of GPUs is possible, then deploy inference on-prem for latency/data control. Or vice versa: do sensitive fine-tuning on-prem (with your data local) on a base model that was cloud-trained.

We also have **cloud adjacency** strategies: some clouds offer dedicated appliances (like Azure's Edge Stack or AWS Outposts) that physically sit at customer site but are managed by cloud provider and integrated to cloud management. This gives some sovereignty (data stays local) but is somewhat cloud-dependent still.

For sovereign AI in government or defense, fully air-gapped infrastructure is often required – so they invest in building mini "clouds" on-prem (like the JAIC's DoD AI development platform). These may incorporate open-source or even closed models if licensed but run internally.

Data gravity is a factor: if most relevant data is in your local databases, bringing model compute to the data might be cheaper than pushing data to cloud. One reason for repatriation is egress fees and transfer issues.

Regulatory environment may actually push more sovereign or at least region-sovereign deployments. E.g., EU's AI Act (to be enforced 2025-2026) might require extra controls on general-purpose AI and transparency. Companies might prefer to run an open source model they can inspect and tune to compliance rather than call a black-box API that might not meet all EU obligations out-of-the-box [264] [265]. Also China's new regulations require all AI serving Chinese citizens to align with core socialist values etc., which practically means Chinese companies might prefer domestic models they can ensure follow guidelines, rather than use an API from US which might not censor the same things (though ironically, something like ChatGPT censors political stuff that Chinese govt wants censored anyway, but it also might produce answers out of alignment with CCP narratives beyond just censorship of topics).

So likely we'll see **splintering of AI infrastructure**: US, EU, China, etc., each with more self-contained compute for their models (due to policy or supply chain concerns – e.g., US trying to limit exports of top AI chips to certain countries, causing those countries to develop their own). It's analogous to how cloud has regions – but now possibly entirely separate stacks.

Energy Investment by Tech Firms implies they foresee these constraints and want to secure future capacity. They wouldn't be reviving nuclear plants or acquiring land if they thought current facilities suffice. A World Nuclear Association statement welcoming Microsoft signals synergy between big tech and nuclear industry [257] [266], highlighting that data center demand is pushing nuclear adoption (which ironically might decarbonize some grid parts, a silver lining).

Sovereign AI hardware initiatives: Some countries investing in domestic GPU alternatives (like Europe's EPI project for HPC chips, China's Biren GPU etc.), though so far Nvidia still leads by far. But if geopolitical tension rises (e.g., US-China tech decoupling), each side will invest to have sovereign hardware supply. In extreme scenario, sovereign AI might also mean running on sovereign silicon.

Conclusion of macro trends: The pursuit of AI capabilities is straining existing infrastructure in power and space. This is prompting major investments (nuclear, new DC builds) and may tilt the centralize/decentralize balance: - Hyperscalers will keep building mega-centers (with novel solutions like offshore data centers or those next to power plants). - Enterprises will equip their data centers with more AI gear (some repurposing of older capacity to AI: e.g., shutting some legacy enterprise servers to free power for AI racks). - Efficiency, both algorithmic and hardware (like new cooling or photonic interconnects), is crucial to avoid hitting physical limits too soon.

For a sovereign AI strategy, understanding these trends helps set realistic goals. Not every company can or should have a 1000-GPU cluster – maybe they'll use 100 and then tap into a cloud for burst or large one-time training. Or focus on smaller specialized models that do 90% of tasks at fraction of compute.

One interesting concept is **cooperative federations**: multiple parties contribute to train a model (like BloombergGPT was trained on Bloomberg's data but on AWS infra with huggingface collaboration – mixing private and public). Or aggregating idle compute across organizations (in some grid computing style) – though scheduling and data privacy make that complicated.

In summary, macro-infrastructure limitations are a gating factor for the future of sovereign AI – they will determine how big and how fast one can scale local AI, potentially incentivizing new approaches (like more efficient algorithms or specialized hardware). Tech giants are responding with big capital moves (power plants, advanced cooling, multi-year chip orders), indicating that the demand trajectory is unwavering. Small and mid-size players will need to be smart, leveraging hybrid models and focusing on efficiency to remain in the AI race.

9. Foundational Physics for Quantum Virtual Machines: Structured Light, Holography, and Quantum Basics

To fully appreciate Quantum Virtual Machines (QVMs) and related hybrid quantum-classical systems, it's helpful to review some **foundational physics concepts** that underlie quantum computing and advanced optical computation. In particular, understanding **structured light** (such as light beams carrying orbital angular momentum or forming toroidal vortices) and **computer-generated holography** (which uses patterns like Fresnel zone plates) can illuminate how quantum states might be simulated or manipulated using classical light and how optical elements can be optimized via GPU computation. We also briefly touch on fundamental quantum mechanics principles relevant to QVMs.

9.1 Structured Light: Orbital Angular Momentum and Toroidal Vortices

Structured light refers to light fields that are engineered to have complex intensity, phase, or polarization structures, beyond a simple Gaussian beam [267] [268]. Two key examples: - **Orbital Angular Momentum (OAM):** Light can carry orbital angular momentum by having a helical phase front (phase varies azimuthally around beam axis). Such beams have a donut shape intensity (phase singularity at center) and are often called **vortex beams**. They are characterized by an integer "topological charge" ℓ which indicates how many 2π phase twists occur around a circle. OAM provides a theoretically infinite-dimensional basis of modes (ℓ can be any integer) [269] [270]. This is useful for increasing data capacity in optical communication (each OAM can carry a channel) and in quantum optics for high-dimensional quantum states [271] [272]. Photons with OAM ℓ actually carry $\ell\hbar$ angular momentum each (in addition to spin angular momentum from polarization) [270] [273]. - **Toroidal Vortices (Optical Hopfions):** A toroidal light field is one shaped like a torus (a doughnut), potentially with a twisting structure in both toroidal and poloidal directions. Recent research mentions *photonic toroidal vortices* and even "optical hopfions" (fields with knotted/twisted phase topology) [274] [275]. These are exotic structured lights that can have unique propagation or focusing properties and are studied for *topological photonics* and high-dimensional entangled photon states [272] [275]. For example, a toroidal vortex beam might have OAM in one direction and some longitudinal angular momentum component due to the torus structure. Kremer et al. (Science, 2022) examined how OAM influences the dynamics of a toroidal vortex beam, finding that swirling flow (due to OAM) destabilizes the torus motion [276] [277]. This indicates how complex mode structure interacts.

Why does this matter for QVMs or computing? **Quantum information can be encoded in structured light modes**: Instead of using e.g. polarization (2D Hilbert space), one could use OAM modes to represent qudits (d-dimensional quantum units). Some experiments generate entangled photons entangled in OAM degrees of freedom, achieving higher information density per photon [271] [272]. A QVM that simulates photonic quantum systems might need to simulate such structured modes. Representing an OAM mode requires including phase singularities and possibly large spatial grids – computationally heavy, but GPUs can handle wave propagation.

Furthermore, classical simulation of quantum wavefunctions (like using Fourier optics analogies) may leverage structured light concepts. For instance, an electron orbital in an atom has angular momentum quantum numbers; one could simulate a 2D cross-section of an atomic wavefunction by a phase pattern like a vortex (ℓ corresponds to quantum number m). Indeed, some proposals use lasers with OAM to manipulate atoms (optical tweezers that impart orbital momentum to trapped particles).

Light-based computing and holography: OAM beams can be generated by computer-generated holograms (like forked diffraction gratings). GPUs can calculate hologram patterns to create various structured beams. For example, a *phase-only spatial light modulator (SLM)* can display a computed hologram that transforms a Gaussian laser into a vortex beam with desired ℓ. Designing those holograms (like a spiral phase plate or a complex diffractive optical element) is done via algorithms (often iterative Fourier transform algorithms). The GPU acceleration here allows real-time or high-res hologram generation.

9.2 Computer-Generated Holography and Fresnel Zone Plates

Computer-Generated Holography (CGH) is the process of using a computer to calculate the interference pattern (hologram) that produces a desired image or wavefront when illuminated. Two key related concepts: - **Fresnel Zone Plates (FZP):** A simple type of hologram that focuses light, basically a pattern of concentric rings (zones) alternating transparent/opaque or phase shifted by π. It works like a lens – constructive interference at focus for light from alternate half-wavelength zone widths. An FZP is essentially a computer-generated hologram for a focus point. They are used in x-ray optics (where traditional lenses don't work well) and in some projection systems [278] [278]. They can also be considered a special case of a hologram of a point source. Calculating an optimal zone plate (maybe apodized or multi-focal) is done on computer. - **Fourier and Fresnel Transforms:** Holography heavily involves simulating diffraction, which mathematically is a Fresnel or Fraunhofer (far-field) transform of the aperture field. GPUs are excellent at performing FFTs, which is why CGH can be accelerated. For example, to compute the hologram that produces a given image at some distance, one can take the image's Fourier transform (for far-field hologram) and encode that as phase on an SLM. For Fresnel (near-field) holograms, more complex phase patterns (like Fresnel zone plate patterns) are computed taking into account distance. There are direct algorithms like *point-based hologram calculation* which sum contributions of each object point (like a light field simulation) – that's O(N_pixels * N_object_points). GPU parallelization allows computing high-res holograms by distributing the sum across cores [279] [280]. Papers show multi-GPU clusters doing 8K hologram computation in real-time by parallelizing point calculations [279] [281]. - **Speckle and optimization:** CGH often involves optimizing the phase pattern to minimize speckle noise or other artifacts [282] [283]. This can be treated as an iterative problem (e.g., Gerchberg-Saxton algorithm) where each iteration does FFTs – again GPU-amenable. A Nature review (2020) on CGH algorithms highlights how different techniques (Gerchberg-Saxton, simulated annealing, etc.) trade off quality vs compute [282] [284]. With enough GPU power, more brute-force or high-quality algorithms become practical in near real-time (like Mixed Integer programming approaches for multi-depth holograms, which previously were too slow).

Holography for QVMs: If one wanted to simulate a quantum wavefunction propagation (like an electron diffraction through a slit), the physics is identical to light diffraction (Schrödinger equation and paraxial wave equation map to each other). So one could use CGH methods to simulate quantum wave optics. E.g., simulate double-slit interference of electrons by using an FFT of aperture function – exactly how one would compute a hologram for that aperture to see an interference pattern. A GPU can do that quickly, thus acting as a QVM for a quantum experiment in wave optics. For more complex quantum states (like entangled photons or multi-particle states), holography might not directly apply, but e.g., simulating two-photon

interference (Hong-Ou-Mandel effect) could be done by simulating light propagation from two sources overlapping – something optical design software do and can be accelerated.

Quantum mechanical fundamentals relevant to QVM: - **Superposition:** QVMs need to represent superposition of states. In simulation, this means tracking amplitude and phase of each basis state (like a vector of complex numbers). That's why memory grows exponentially with qubit count – 2^n amplitudes. Techniques like **State Vector simulation** (simulate full 2^n vector, often done with GPU linear algebra) vs **Tensor Network** (simulate using network contraction, which can handle more qubits if entanglement is limited) come into play. - **Entanglement:** Simulating entanglement is where classical difficulty comes. If we restrict to low-entanglement states (e.g., a cluster state with limited connections), one can use approximations or break it into smaller subsystems (some QVMs do this adaptively). - **Measurement:** QVM must simulate measurement statistics. That's done by computing probabilities (squared amplitude) and using random numbers to pick an outcome according to distribution. On GPU, one can do that by partial summations of amplitude norms. - **Quantum gates:** Each gate is a unitary matrix on state vector. GPU can apply those by matrix-vector multiplies (for 2-qubit gates on portion of vector, etc.). Libraries like cuQuantum provide these primitives.

Bridging Physics and GPU computing: The mention of **AlphaQubit** earlier draws a line – they used Transformers to analyze output of quantum error circuits [122] [121]. That itself is bridging physics experiment data with AI. Under the hood, they had to simulate many quantum error events (with QVM) to train the model. That means heavy use of randomness and physics simulation – reminiscent of Monte Carlo. GPUs, besides deterministic FFTs, also excel at parallel Monte Carlo (shoot many random trials of a quantum circuit with noise, which can be embarrassingly parallel across threads). That's how Google likely produced millions of training samples for AlphaQubit: simulate error patterns on 49 physical qubits with some error model – each simulation independent, distributed across TPU/GPU cores – then feed results to the neural net.

Optical computing and holography to implement quantum-like processes: Some propose using optical setups as analog computing for certain math. E.g., an optical 4-f system performs a Fourier transform physically, essentially doing with light what a GPU does with FFT (but at speed of light). If structured light can encode a matrix problem, shining through a hologram yields an answer (see optical neural networks using SLMs as weight matrices). But accuracy and flexibility are issues.

Relevance to sovereign AI computing: Why do we care about these physics in our context? 1. To simulate or design quantum hardware (like verifying a quantum photonic chip's behavior), a sovereign AI might run large physics sims on GPUs – knowledge of structured light and holography helps them optimize code or understand results (like using Fourier optics analogies to validate a quantum optical circuit design). 2. If someone aims to implement a QVM, knowing physics helps ensure simulation fidelity (e.g., simulating orbital angular momentum qubits or using holographic methods for larger quantum systems). 3. Some advanced AI sensors (like holographic memory or photonic computing accelerators) might come into play. For instance, **holographic optical memories** store data in volume using interference patterns – managing those requires computing zone patterns to write/read without cross-talk [285] [285]. Sovereign computing might incorporate novel storage or interconnect using holography (holographic data storage has high density potential). 4. **Quantum-inspired algorithms:** Concepts like OAM provide mathematical tools (like using angular momentum basis for transforms). Perhaps algorithms for AI (like using Fourier transforms in cylindrical coordinates or so) are inspired by these physical analogies.

In summary, foundational physics – structured light and holography – not only underlie how we might simulate quantum phenomena but also inspire computational techniques (like using light as analog compute or designing GPU algorithms to mimic wave propagation). Understanding them enriches the capability to build efficient simulations (QVM) and even consider hybrid optical-electronic computing for future AI accelerators.

9.3 Quantum Mechanics Fundamentals in QVM Context

To tie off the physics review, recall a few **quantum fundamentals** relevant to simulation: - **Qubits vs classical bits:** Qubits exist in $|0\rangle$ and $|1\rangle$ superposition $\alpha|0\rangle+\beta|1\rangle$. Simulating one qubit is like a point on Bloch sphere (two complex parameters). Many qubits: state = tensor product of each, rapidly expanding parameter count. - **Unitaries:** Quantum gates are unitary matrices (preserve norm). Simulators must apply these exactly (e.g., a rotation gate is $\cos\theta$, $i\sin\theta$ matrix). Physically, these correspond to evolutions (like an electromagnetic pulse causing a rotation on Bloch sphere). - **Interference:** At core, quantum amplitudes interfere (like waves). That's why analogies to light interference are apt. E.g., probability of two paths is |amp1+amp2|^2. Holography deals with interference patterns to encode info – similar math. - **Entanglement:** If a simulator tracks a state vector, entanglement naturally arises since vector isn't separable into product form. But if using approximate methods, one might restrict entanglement (like Matrix Product States effectively assume limited entanglement). - **Measurement collapse:** After simulating up to a measurement, one must choose an outcome and update state accordingly (collapsing wavefunction to the subspace of outcome). This means branching – if one wants to explore multiple runs, they may need to clone state with each possible measurement outcome and weight them (like in computing expectation values). - **Noise and decoherence:** Real quantum hardware suffers noise. QVMs often simulate noise by randomizing states or applying Kraus operators. This can drastically increase simulation complexity (need random sampling over many trajectories or evolving a density matrix which is vector of size 4^n for n qubits – expensive). Often, *quantum trajectories* method is used: do many pure state simulations each with stochastic noise, average results [286] [287]. GPUs shine at many parallel runs (like one noise seed per thread). - **Quantum Volume metrics:** There's metrics like Quantum Volume (QV) to measure hardware capability. QFaaS used QV, CLOPS (circuit layer operations per second) as resource allocation guides [288] [288]. A QVM might be used to estimate these metrics for hypothetical hardware or to validate improvements.

All in all, bridging quantum fundamentals with classical computing techniques (like structured light for analogies, holography for computing diffractive transforms) equips one to implement QVMs more effectively or to interpret results of hybrid systems. It also provides a conceptual toolkit to innovate: e.g., using Fourier transforms (like in holography) to diagonalize certain quantum operations for faster simulation, akin to how one uses FFT to speed up convolution (going to frequency domain multiplies instead of time domain convolution sums). Similarly, some quantum evolutions (like free propagation) are essentially Fourier transforms – one can simulate them by switching basis via FFT, apply phase shifts, inverse FFT (this is how light propagation often simulated – using Fourier optics). A QVM might thus incorporate such optimizations – doing an FFT for the parts of Hamiltonian that are diagonal in momentum basis, etc.

In the context of sovereign AI computing, having this deep physical insight is a differentiator: it's a layer of knowledge that may not be widely present in off-the-shelf cloud tools. It can lead to more optimized or specialized local solutions (like a national lab building a QVM with GPU-accelerated photonics simulation to design a new quantum sensor – something better done in-house with domain experts than via a generic cloud service).

We've now covered a broad terrain: from triple-transformer architectures and hardware optimization, to agent frameworks, multi-agent protocols, hybrid quantum integration, model ecosystems, use cases in coding/compliance/media, macro trends in infrastructure, and fundamental physics foundations. Finally, in the concluding section (10), we will discuss open challenges and future directions for scaling sovereign AI – touching on reproducibility, trust, regulation, cultural impact, and economic models – essentially synthesizing where we stand and what lies ahead for truly sovereign AI computing.

10. Open Challenges and Future Directions for Scaling Sovereign AI

Despite rapid advances, there remain numerous **open challenges** on the path to scalable, trustworthy, and widely adopted sovereign AI systems. In this concluding section, we highlight some key issues and future directions: ensuring **reproducibility** of AI results, maintaining **trustworthiness and alignment** of models, navigating emerging **regulations**, overcoming **hardware constraints**, addressing **cultural and societal implications** of widely deployed AI, and developing sustainable **economic models** around AI deployment.

10.1 Reproducibility and Scientific Rigor

As AI systems become more complex (especially with multi-agent loops and dynamic context integration), the ability to reproduce results is paramount yet increasingly difficult. An LLM's outputs can vary with minor prompt changes or as underlying models update [77] [289]. For agent systems that involve randomness (e.g., in exploring solutions or multi-agent interactions), run-to-run variability can be high [76] [290]. This poses problems for both research and deployment: - **Logging and Versioning:** It's critical to log model versions, prompts, and steps in agent workflows. A 2025 commentary stresses that "structured reporting is essential" because without clear documentation of model versions, hyperparameters, and prompting strategies, it's nearly impossible to understand or reproduce an observed effect [291] [292]. They propose a **DEAL checklist** (Development, Evaluation, and Assessment of LLMs) to standardize reporting of experimental conditions [293] [293]. Tools like experiment tracking (Weights & Biases, MLflow) and prompt versioning will likely become standard in sovereign AI pipelines to ensure one can trace what was done. - **Determinism vs Stochasticity:** For some applications, setting a fixed random seed and running deterministic algorithms is feasible (e.g., for certain training jobs or knowledge retrieval). But for interactive agents or RL, some nondeterminism is inherent. A solution is averaging over multiple runs or measuring variability. The Radiology AI piece notes authors should measure and report variability ("how stable the findings truly are") by running models multiple times on same task [294] [295]. In critical deployments, one might run an agent multiple times and either ensemble the results or flag if they differ significantly. - **Evolving Models:** Proprietary cloud models might update silently, breaking reproducibility over time (as noted, cloud LLMs can change outputs as they get fine-tuned in production without version bump) [77] [289]. Sovereign AI mitigates this by giving full control of when a model is updated. One can keep using an older model if needed to reproduce earlier analysis, or at least archive it. Still, if one does continuous fine-tuning, one must version those fine-tunes too. - **Multi-agent complexity:** As we chain agents (especially if each agent is an LLM, themselves with internal randomness or learning), the overall system becomes akin to a non-linear dynamical system that might be chaotic (small differences amplify). Ensuring reproducibility might require developing stable agent protocols (like having them use a shared scratchpad for reasoning so that outcomes depend less on unpredictable emergent dialog and more on structured exchange). - **Benchmarking and Evaluation:** There's also a challenge in reproducibly evaluating AI systems. Many benchmarks were static question sets, but now tasks might be interactive or context-dependent. NeurIPS 2025 workshop on LLM Lifecycle talked about evaluating emergent abilities and such [296] [297]. Possibly new evaluation harnesses (like evaluation agents or simulation environments) will be built. Sovereign developers

should adopt robust internal testing for their models/agents to catch when an update unexpectedly degrades some ability.

In scientific research contexts, a surprising amount of LLM research currently cannot be replicated easily outside the original authors due to missing details or closed models [291] [292]. This is antithetical to science. Sovereign AI, by enabling open models and full-stack control, can improve this by allowing researchers to share full code and weights. The community push (ML Reproducibility Challenge, etc.) is emphasizing this [298] [299]. We expect in future: *reproducibility checklists, model cards with detailed training and alignment info, and possibly regulatory requirements for record-keeping (the EU AI Act will require documentation of training data, model parameters for foundation models)* [300] [301].

10.2 Trustworthiness, Alignment, and Misuse Prevention

Ensuring AI systems behave in **trustworthy** ways is an ongoing challenge: - **Bias and Fairness:** Models can reflect or even amplify societal biases present in training data. Open models like DeepSeek had hidden censorship biases [177] [180]; others might have biases against or for certain demographics. Ensuring fairness might require curating training data or fine-tuning on balanced data. There's also the approach of building **evaluation pipelines**: e.g., have a bias detection agent that probes the main model with various prompts to measure if outputs favor or disfavor certain groups. Then use that feedback to adjust the model (fine-tune or via prompts). Some frameworks (Anthropic's constitutional AI) try to encode broad values to minimize harmful biases [183] [184], but value alignment itself is subjective (who decides the constitution?). - **Preventing Harmful Use:** Sovereign AI systems could be misused to generate disinformation, malware code, etc. Without built-in guardrails of an API, the onus is on operators to implement restrictions. For instance, an enterprise might instruct their model: “Do not output certain sensitive data or perform tasks outside guidelines.” They could enforce this via a combination of training (RLHF to refuse certain classes of requests) and **runtime monitoring** (content filters that scan outputs for red-flag content). There is research on *sandboxing LLM behaviors* – perhaps running an LLM in a VM where if it tries to execute code, you intercept, etc., akin to how you sandbox any application. But LLMs acting as agents blur the line (they might request to run a dangerous tool, and the orchestrator can simply say "no, not allowed"). - **Transparency:** Trust is improved if users understand why the AI produced something. This could be through **explanations** (like chain-of-thought that the user can optionally see, or a separate explanation mode). For multi-agent systems, having logs of agent dialogues and tools used can help an auditor verify the system didn't do something nefarious. Even for final answers, tools like verifying citations (as NotebookLM does) raise trust – you see the source of each claim [225] [98]. There’s movement in research on `scaffolding` and `debugging LLM reasoning` (like MIT’s *Language Model Self-Consistency* ensures multiple reasoning paths converge). - **Continuous Alignment:** Even if a model is aligned at deployment, over time new edge cases or exploits appear (like jailbreaking prompts). A sovereign AI team should treat alignment as a continuous process – collecting user feedback, discovering new failure modes, and then updating the model or adding patch rules. OpenAI does this behind scenes; sovereign users must do themselves. That might involve setting up a feedback loop where if any user sees a problematic output, it’s flagged and triggers re-evaluation and retraining. This is where small-scale actors might struggle (they don't have millions of interactions to learn from or large safety teams). One solution is open collaboration: companies might share safety data for open models (there are efforts like LAION’s OpenAI Safety development, or government creating open safety benchmark datasets). - **Agent Safety:** With multi-agent, there's concern of runaway behaviors (e.g., one agent prompting another to do something unintended). Ensuring agents have **bounded authority** (like each agent can only do specific tools, and a higher-level supervisor agent monitors logic) is important. It's analogous to microservices with limited permissions – principle of least

privilege applied to AI agents. The A2A protocol design principle "secure by default, with enterprise-grade auth" [106] [107] is promising. Practically, it means using authentication tokens on MCP servers, logging all tool uses, and possibly including a "policy agent" that reviews all proposed external actions. - **Human-in-the-loop:** For high-stakes decisions (medical, legal, etc.), a human should verify. The AI should flag uncertainty or ask for confirmation if it's not confident. In sovereign settings, one can custom-build UIs to enforce that (for example, an AI drafting an email in a corporate environment might by policy require user to click send, not auto-send). It's easier to implement these workflows when you control the system end-to-end. - **Mis/disinformation:** A big societal risk is AI-generated false content (text or deepfakes). Solutions include watermarking AI outputs or detection algorithms. If an organization uses AI to generate external content (marketing, PR), they might embed subtle signals so that if that content is found later, it can be identified as AI-generated. Some open watermarking schemes exist (OpenAI researched a text watermark, but that was broken by paraphrasing easily). Another concept is **provenance tracking**: using cryptographic methods to sign outputs at creation so their origin can be verified. For images and video, standards like C2PA (Coalition for Content Provenance and Authenticity) let producers include metadata of how content was created. A sovereign AI content generator could automatically attach a signed metadata that it made this content on this date (if the user chooses to reveal it). This helps fight disinformation because trustworthy publishers can prove authenticity, making the rest suspect by comparison.

10.3 Regulation and Governance

The regulatory landscape for AI is evolving quickly: - **EU AI Act (2024/2025):** This is the first broad regulation. It will require things like: - **Transparency for foundation models:** Providers must disclose training data info, energy use, etc., and mitigate risks [302] [301]. If a company deploys its own foundation model, it might have to self-assess and meet these requirements (depending on interpretation; there's debate if internal-only models are covered). - **High-risk use compliance:** If using AI in high-risk contexts (e.g., credit scoring, medical diagnostics), there are strict requirements on quality, traceability, human oversight. - **Generative AI obligations:** There’s a mention of requiring generative AI to have safeguards against generating illegal content and maybe requiring watermarks/disclosures for AI-generated content [303] [304]. Sovereign AI systems must incorporate those if relevant (for instance, an AI writing news articles might need to clearly label it as AI-generated to comply). - **Data governance:** Ensuring training data is free from illegal content, biases, etc. – so enterprises might have to carefully curate any data they use to train or fine-tune models, not just scrape the web blindly. - **Sectoral Regulations:** Finance (like SEC or FINRA in US) may issue guidelines on using AI in trading or advising (ensuring decisions can be explained, etc.). Healthcare regulators (FDA) are already requiring evidence of safety and effectiveness for AI diagnostic tools. If a hospital uses a sovereign AI model for, say, analyzing X-rays, they might have to validate it similarly to a medical device, including extensive documentation and monitoring of performance drift. - **Intellectual Property:** Another regulatory aspect is copyright/patent. If a sovereign AI is trained on copyrighted data, do outputs infringe? There's ongoing litigation (e.g., authors suing OpenAI). On-prem models likely use public or licensed data, but companies need to mitigate risk by e.g. filtering training data for known copyrighted text or images. Or use new tools that mark where an output might contain memorized copyrighted phrases (some research tries to detect verbatim regurgitation). - **Privacy laws:** Using personal data to train requires either consent or complying with something like GDPR's legitimate interest and data minimization. EU AI Act specifically emphasizes that if model training data has personal data, GDPR still applies – likely requiring deletion requests compliance, etc. For sovereign AI, that means establishing processes: if someone asks "remove my personal info from your model," do you have a way to either retrain or fine-tune to forget it? That's an open research problem (machine unlearning). Possibly we'll see methods to surgically remove traces of specific data from models after training, to comply with such

requests [305] [305] . - **Liability frameworks:** There's debate on who is liable if AI causes harm – the provider, deployer, or user? EU is also working on an AI Liability Act. Sovereign AI means the company deploying it likely bears liability (they can't push blame to a vendor). So that will motivate thorough testing and safety measures internally. - **Standards and Best Practices:** We might see ISO or IEEE standards on AI risk management. NIST already released an AI Risk Management Framework. Adhering to these can both improve outcomes and serve as evidence of due care legally. Sovereign AI projects should integrate risk management from design: e.g., having an AI ethics panel internally, doing bias audits, documenting use cases and limitations (model cards, system cards). - **Global Fragmentation:** Different regions will have different rules (China's laws on deep synthesis content require watermarking and registration of algorithms, etc., effective Jan 2023; EU's approach; US likely more hands-off but with sectoral guidelines). A multinational company running a sovereign AI might either silo models per region or ensure the model meets the strictest common requirements.

Staying ahead of regulation is both a challenge and an impetus for innovation. It might slow some deployments in short term (due to compliance overhead) but in long term can increase trust (people and governments will be more comfortable with AI if robust guardrails are mandated).

10.4 Hardware and Scalability Constraints

We already discussed macro trends in hardware (Section 8). To reiterate challenges specifically: - **Memory and Bandwidth:** The largest models push memory limits of single devices, requiring model parallelism which is complex. Even if compute is plentiful, memory might bottleneck (e.g., needing 1 TB of GPU memory for a trillions-parameter model). Techniques like **mixed precision (FP8)** [89] [90] , **quantization** and **sparsity/MoE** (only using a fraction of weights per inference) will be necessary to go further without exorbitant memory. Also, new memory tech (HBM3e, maybe optical memory) can help. - **Interconnect and Communication:** In multi-GPU systems, network bandwidth can limit scaling efficiency beyond certain number of nodes (that's why many large trainings happen on large pods with fast NVLink/Switch – still, e.g., GPT-3 was across 1024 GPUs and likely had <50% utilization due to comm overhead at times). Solutions: improved interconnect (NVIDIA InfiniBand HDR, NVLink Switch, etc.), and algorithmic (pipeline parallel to overlap comm with compute, partitioning models to reduce needed sync). - **Energy and Cooling:** We see cooling innovations: immersion cooling, liquid direct to chip, etc. Perhaps sovereign data centers will adopt those for dense AI racks. There's also experimentation with alternative computing (like analog in-memory compute to reduce energy per operation). If successful, those might appear in commercial AI accelerators and companies might use them for inference to cut power costs by an order of magnitude. However, analog approaches often face precision issues. Still, any efficiency gain (like 8-bit compute, or digital accelerators like Cerebras wafer-scale engine focusing on memory locality) helps if integrated well. - **Quantum hardware?** The dream: if quantum computers mature, some tasks could be offloaded and done faster. However, currently quantum computers are far from general advantage on ML tasks. Still, certain optimization or sampling tasks could get speed-ups. This is something to watch: if by 2030 there's a quantum accelerator that can do e.g. massive parallel sampling or solve specific NP-hard subproblems, a sovereign AI system might incorporate it (like using a QPU to handle the combinatorial explosion part of a discrete optimization, while classical handles rest). But short-term, hardware constraints are tackled by better classical tech. - **Software and Algorithms:** There's still room for algorithmic breakthroughs to reduce compute needed. E.g., *sparsity*: models currently consider every parameter for every input. Research on selective activation (beyond MoE, maybe conditional computation at a finer grain) could cut computation significantly. Also *distillation*: using smaller models that approximate larger ones for inference saves cost. Many companies might choose to train a big model then distill to a 10x smaller model that is just as good

for domain tasks – massively reducing runtime cost. - **Scaling laws economics:** There's debate if we can continue to benefit from just scaling up parameters and data as much as before. At some point, diminishing returns plus infrastructure costs might slow the brute-force scale trend (like GPT-4 not hugely bigger than GPT-3, focusing more on data quality and multimodality instead). Sovereign AI strategy could be to not chase the absolute biggest model, but focus on *right-sized models* – those that achieve needed performance with reasonable compute. E.g., 40B open models now rival 175B older ones. A well-curated 40B might suffice for many tasks, saving cost and making on-prem feasible, rather than insisting on a 500B model that only a few hyper-scalers can run. Already, we see this thinking: the open LLaMA-2 70B often outperforms GPT-3.5, proving efficiency matters. DeepSeek jumped to 671B parameters but only 37B active, showing clever ways to scale logically without linear cost scaling.

10.5 Cultural and Societal Implications

As AI becomes pervasive (and sovereign AI means every organization or community can have its own model), **cultural impacts** are significant: - **Fragmentation of Reality:** The "Balkanization of AI" piece argued that if every group uses an AI aligned to their views, we lose shared reality and exacerbate polarization [10] [200]. People may trust their AI to reinforce biases (since AI can be made to agree with user strongly [201] [202]). To mitigate, some propose **pluralism** in AI – designing AI to present multiple perspectives or actively challenge a user's stance with alternative views (within respectful bounds). Google's A2A principle of enabling agents from different vendors to collaborate hints at connecting these silos somewhat [86] [306]. - **Disinformation and Social Trust:** With more powerful local generation, anyone can create highly persuasive fake content. Society will need to adapt critical thinking ("trust but verify, then maybe don't trust at all" as the article wryly notes [307] [184]). Sovereign AI developers might build detection tools and champion authentication of legitimate content (maybe adding cryptographic signatures to content their AIs produce so it can be distinguished – technical solutions to maintain trust in an AI-saturated info ecosystem). - **Labor and Economic Disruption:** Locally deployed AIs could automate white-collar jobs (drafting documents, basic design, coding). This raises questions: how to upskill workers to work with AI, how to redistribute productivity gains. For businesses, short-term it might boost productivity; long-term society needs to address shifts in job roles. If every enterprise runs AI interns or assistants, roles like junior analysts, paralegals, etc., might change significantly. - **Education and Creativity:** Students with easy access to AI summaries (NotebookLM style) might rely on it too much, possibly hindering deep learning. Or it could enhance learning by allowing exploration of ideas beyond their current understanding (like a tutor that can answer any question from textbook). Educators are still grappling with preventing misuse (cheating vs legitimate aid). Possibly there will be guidelines where using AI is acceptable (e.g., as study aid) vs not (e.g., to generate an entire term paper). The presence of sovereign AI in every student's device in future could force a rethinking of curricula toward focusing on critical analysis and less on rote tasks (since AI can do rote work). - **Global Inequality:** Large companies or wealthy nations can afford sovereign AI computing infrastructure; smaller players or developing nations might struggle, relying on either foreign cloud or not having access at all. This could widen digital divides. One future direction to counter this: more open models that run on consumer hardware (like how Stable Diffusion brought image generation to anyone with a decent GPU). If efficient models that run on say a smartphone become reality, then AI assistance is democratized. Projects like Mistral AI explicitly aim to make powerful models accessible without huge compute. - **Cultural Preservation vs Homogenization:** Sovereign AI can be tailored to local languages and customs (a positive – we can have AI conversant in low-resource languages which big providers may ignore for lack of profit). This could help preserve languages or provide services in them. On flip side, if most foundational models come from one culture and others fine-tune slightly, there might be cultural bias embedded. Encouraging each region to develop or at least heavily adapt models can ensure AI reflects

diverse worldviews rather than a Silicon Valley-centric perspective. It's a challenge to have truly multilingual, multicultural models; one promising approach is open collaborations across countries (like BLOOM model was a volunteer effort to make a model that supports many languages). - **Ethical frameworks and Value Alignment:** Different societies will program different moral guidelines (China wants alignment with socialism, Western companies often align with human rights and non-discrimination, etc.). This is fine if AIs are largely used internally within those societies. But what if they interact (like an open internet where many AIs talk)? Some foresee "culture clash" at AI level. Principles like those from A2A (ensuring agents can collaborate despite not sharing memory/tools) are akin to making them interoperable even with different values. But if two agents disagree on fundamental values, outcomes could conflict. Perhaps meta-protocols will handle such conflicts or escalate to human arbitration.

10.6 Economic and Sustainability Models

Finally, the question of sustaining sovereign AI development and deployment: - **Cost vs Benefit:** Running huge models is expensive (electricity, hardware depreciation, maintenance). Organizations will weigh if it's worth doing in-house vs renting. Possibly, as open models get better and cheaper to run (via optimization, or specialized hardware that is cost-efficient at scale), we might see on-prem cost drop. Also, the value of customization and data privacy might justify higher cost for many. - **Open Source Funding:** Many open LLMs are created by community or nonprofits (Hugging Face, EleutherAI) sometimes supported by gov grants or company donations. That model might need to continue to ensure a free alternative exists to closed offerings. Governments may invest in open models as a matter of digital sovereignty (EU and some national labs are funding open model projects to reduce reliance on foreign tech). - **AI-as-a-Utility:** Some imagine future where computing (especially AI) is like a public utility. If data center scarcity and power issues persist, maybe governments ration or coordinate AI compute usage (e.g., allocate compute quotas to companies or encourage sharing unused compute for societal needs). This is speculative, but not unprecedented: e.g., radio spectrum is regulated and allocated for common good. If AI becomes critical infrastructure, some oversight or shared frameworks might come. - **Sustainable AI:** There is pressure to make AI development more environmentally sustainable (the carbon footprint of training GPT-3 was estimated equivalent to several car lifetimes). If training is done multiple times by many orgs independently, that multiplies cost. Some call for more model reuse (like open models that everyone fine-tunes rather than each training from scratch). Also, using renewables to power data centers, and optimizing code to minimize wasted compute. Sovereign AI operators should track their energy usage and try to mitigate (some may even get carbon credits or be required by ESG goals to reduce carbon per inference, etc.). Techniques like model compression reduce energy per inference – that’s good for both cost and environment. - **Monetization and Market:** If every company has their own model, what’s the role of AI service companies? Possibly shift from selling inference to selling expertise (like helping set up and maintain sovereign systems, similar to how Red Hat sells support for open source). We might see a robust ecosystem of vendors providing on-prem AI solutions (hardware kits, fine-tuned models for domains, MLOps software to manage them). Cloud providers might adapt by offering hybrid solutions (like rent GPUs but let customer fully control model and data). - **Talent and Training:** Another constraint is having enough ML engineers and DevOps to manage these systems. Not every enterprise has that. This might lead to more user-friendly tools that abstract complexity (like how early DBs required experts but now many use them easily; similarly, one could envision admin consoles for AI models that handle scaling, monitoring, updates with less manual work). However, in short term the demand for talent is high. Companies might form partnerships or outsource management to specialized firms (e.g., there are startups now that will manage a private LLM for you on your infrastructure – essentially providing the missing team if you pay them).

In conclusion, scaling sovereign AI will require overcoming technical, organizational, and societal challenges. But the trajectory seems set: increasing capabilities decentralized into the hands of many, rather than just a few AI giants, providing those challenges can be met responsibly.

Final Thoughts: Sovereign AI computing, epitomized by QFaaS Triple-Transformer systems, represents a paradigm shift towards local empowerment and control of AI. By intelligently leveraging hardware (CPU–GPU–NVMe trifecta) [3] [288], optimizing at low levels [28] [308], and embracing advanced frameworks (Antigravity, Conductor/MCP, ATHENA) [45] [1], organizations can achieve remarkable AI autonomy – performing feats from coding to scientific discovery within their own walls. Multi-agent collaboration via standards like MCP allows combining strengths of many AI components securely [58] [62]. Integrating hybrid quantum simulation hints at future acceleration for certain tasks [131] [134]. Open-source LLMs like DeepSeek demonstrate that with innovation (FP8, MoE) we can close the gap with proprietary models [88] [153], enabling high-performance AI without external dependency.

Yet, with this power comes responsibility: to reproducibly validate results [291] [292], to align AI behavior with human values and laws [183] [184], to mitigate biases [177] [180], and to prevent misuse (whether by bad actors or simply by erroneous AI actions). The future of sovereign AI will involve interdisciplinary collaboration – between engineers, ethicists, policy-makers, end-users – to craft systems that are not only intelligent but also transparent, fair, and beneficial. As each challenge is met – from hardware limits to regulatory compliance – the vision of democratized, sovereign AI comes closer: where individuals and communities harness AI on their own terms, unlocking innovation while preserving the values and autonomy that define them.

References:

1. Abadi, M. et al., *Deep Learning with Differential Privacy*, 2016.
2. Anthropic, *Introducing the Model Context Protocol*, 2024 [51] [58].
3. Ballinger, K. et al., *Conductor: Context-driven development for Gemini CLI*, 2025 [1] [47].
4. Besnier, V. et al., *FP8-LM: Training FP8 Large Language Models*, 2023 [89] [90].
5. Bubeck, S. et al., *Sparks of AGI: Early experiments with GPT-4*, 2023.
6. DeepSeek-AI, *DeepSeek-V3 Technical Report*, 2024 [88] [153].
7. Google Developers Blog, *Build with Antigravity*, 2025 [45] [41].
8. Google Developers Blog, *Announcing Agent2Agent (A2A) Protocol*, 2025 [112] [106].
9. He, C. et al., *Towards higher-dimensional structured light*, Light: Sci & Appl, 2022 [269] [270].
10. Kremer, P. et al., *Photonic Toroidal Vortices*, Science, 2022 [274].
11. Liu, T. et al., *Variational Quantum Rainbow DQN*, arXiv:2512.05946, 2025 [131] [134].
12. Morbin, T., *The Balkanization of AI and the Search for Truth*, 2025 [10] [307].
13. NIST, *AI Risk Management Framework 1.0*, 2023.
14. Pew Research, *Energy use at U.S. data centers and the AI boom*, 2025 [247].
15. Qiu, P. et al., *Censorship in DeepSeek*, Info. Sciences, 2026 [99] [177].
16. Satvik, T. & Alkhulaifat, D., *Reproducibility in LLM and Agentic AI Research*, Radiology: AI, 2025 [291] [294].
17. Wang, B. et al., *LAVE: LLM-powered Video Editing Assistant*, ACM IUI 2024 [228] [230].
18. Zhang, J., *DeepSeek Technical Analysis Series*, Medium, 2025 [89] [165].
19. (Additional references numbered according to source list above)

1 2 7 47 50 56 57 64 65 Conductor: Introducing context-driven development for Gemini CLI - Google Developers Blog
https://developers.googleblog.com/conductor-introducing-context-driven-development-for-gemini-cli/

3 4 123 124 125 126 127 128 135 136 288 QFaaS: A Serverless Function-as-a-Service framework for Quantum computing | Request PDF
https://www.researchgate.net/publication/380247187_QFaaS_A_Serverless_Function-as-a-Service_framework_for_Quantum_computing

5 6 A Software Accelerated Key-value Service via the NVMe Interface
https://dl.acm.org/doi/10.1145/3757347.3759143

8 130 131 132 133 134 137 [2512.05946] Variational Quantum Rainbow Deep Q-Network for Optimizing Resource Allocation Problem
https://www.arxiv.org/abs/2512.05946

9 28 29 30 31 32 33 34 35 38 308 NVMe Optimization for Linux Servers | Best NVMe Tuning 2025
https://perlod.com/tutorials/nvme-optimization-for-linux-servers/

10 82 83 84 85 182 183 184 185 199 200 201 202 203 307 The Balkanization of AI and the Search for Truth
https://www.bankinfosecurity.com/blogs/balkanization-ai-search-for-truth-p-3934

11 [PDF] Chemoenzymatic multistep retrosynthesis with transformer loops
https://boris-portal.unibe.ch/bitstreams/37e58ca0-da3e-434d-851b-e596fc309e97/download

12 15 Squeeze more out of your GPU for LLM inference—a tutorial on ...
https://preemo.medium.com/squeeze-more-out-of-your-gpu-for-llm-inference-a-tutorial-on-accelerate-deepspeed-610fce3025fd

13 [PDF] NVIDIA H100 Tensor Core GPU Architecture
https://www.hpctech.co.jp/assets/images/info/catalog/pdf/gtc22-whitepaper-hopper_v1.02.pdf

14 [PDF] Overcoming the Hurdles of GPU Memory Capacity to Train Massive ...
https://vldb.org/pvldb/vol15/p2747-li.pdf

16 17 18 19 Core Pinning
https://amperecomputing.com/glossary/core-pinning

20 37 CPU Management - CPU Pinning and Isolation in Kubernetes - Intel
https://www.intel.com/content/www/us/en/developer/articles/technical/cpu-management-cpu-pinning-isolation-kubernetes.html

21 Goldilocks Isolation: High Performance VMs with Edera - arXiv
https://arxiv.org/html/2501.04580v1

22 Monitoring your HPC/GPU Cluster Performance and Thermals
https://boeroboy.medium.com/monitoring-your-hpc-gpu-cluster-performance-and-thermal-failures-ccef3561e3aa

23 Hyperthreading in HPC - Julia Discourse
https://discourse.julialang.org/t/hyperthreading-in-hpc/101316

24 NVIDIA Multi-Instance GPU (MIG)
https://www.nvidia.com/en-us/technologies/multi-instance-gpu/

25 36 Multi-tenant GPU workloads are finally possible! Just set up MIG on ...
https://www.reddit.com/r/kubernetes/comments/1l9l8gz/multitenant_gpu_workloads_are_finally_possible/

26 GPU Sharing That Works: MIG, MPS & Schedulers - Medium
https://medium.com/@hadiyolworld007/gpu-sharing-that-works-mig-mps-schedulers-b3105933d1aa

27 GPU architecture and warp scheduling - NVIDIA Developer Forums
https://forums.developer.nvidia.com/t/gpu-architecture-and-warp-scheduling/58010

39 CPU Pinning and CPU Sets (2020) - Hacker News
https://news.ycombinator.com/item?id=29379792

40 41 42 43 44 45 46 Build with Google Antigravity, our new agentic development platform - Google Developers Blog
https://developers.googleblog.com/build-with-google-antigravity-our-new-agentic-development-platform/

48 Google Antigravity - AI-Powered IDE | Build the New Way
https://antigravityai.org/

49 Antigravity Is Google's New Agentic Development Platform
https://thenewstack.io/antigravity-is-googles-new-agentic-development-platform/

51 52 58 62 63 93 114 115 260 261 305 Introducing the Model Context Protocol \ Anthropic
https://www.anthropic.com/news/model-context-protocol

53 MCP - Conductor Docs - Quickstart
https://docs.conductor.is/usage/mcp

54 Conductors to Orchestrators: The Future of Agentic Coding
https://addyo.substack.com/p/conductors-to-orchestrators-the-future

55 Conductor-Creator Architecture - Emergent Mind
https://www.emergentmind.com/topics/conductor-creator-architecture

59 60 Announcing the Conductor Model Context Protocol (MCP) Server
https://orkes.io/blog/conductor-mcp-server-announcement/

61 Conductor MCP Server: AEO & SEO for LLMs & Agents
https://www.conductor.com/platform/features/mcp-server/

66 67 68 69 70 73 74 75 [2512.03476] ATHENA: Agentic Team for Hierarchical Evolutionary Numerical Algorithms
https://arxiv.org/abs/2512.03476

71 [論文評述] ATHENA: Agentic Team for Hierarchical Evolutionary ...
https://www.themoonlight.io/tw/review/athena-agentic-team-for-hierarchical-evolutionary-numerical-algorithms

72 208 Build like Microsoft: Developer agents in action
https://devblogs.microsoft.com/microsoft365dev/build-like-microsoft-developer-agents-in-action/

76 77 289 290 291 292 293 294 295 Reproducibility in LLM and Agentic AI Research: Why the DEAL Checklist Matters
https://radiologyai.substack.com/p/reproducibility-in-llm-and-agentic

78 2025 State of AI Cost Management Research Finds 85% of ... - Mavvrik
https://www.mavvrik.ai/2025-state-of-ai-cost-management-research-finds-85-of-companies-miss-ai-forecasts-by-10/

79 Why Enterprises Are Moving Generative AI On-Premises - Pryon
https://www.pryon.com/landing/enterprises-generative-ai-on-premises

80 [PDF] On-Premise vs Cloud: Generative AI Total Cost of Ownership
https://lenovopress.lenovo.com/lp2225.pdf

81 88 113 143 151 152 153 154 155 156 157 158 172 173 174 175 189 190 191 192 193 194 204 [2412.19437] DeepSeek-V3 Technical Report
https://ar5iv.labs.arxiv.org/html/2412.19437

86 87 91 92 94 103 104 105 106 107 108 109 110 111 112 306 Announcing the Agent2Agent Protocol (A2A) - Google Developers Blog
https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/

89 90 147 148 149 150 161 162 163 164 165 166 167 168 DeepSeek Technical Analysis — (5) FP8 Training | by Jinpeng Zhang | Medium
https://dataturbo.medium.com/deepseek-technical-analysis-5-fp8-training-ff34768727b8

95 Why Your Company Should Know About Model Context Protocol
https://www.nasuni.com/blog/why-your-company-should-know-about-model-context-protocol/

96 What Is MCP? Guide to Model Context Protocol for Salesforce Admins
https://admin.salesforce.com/blog/2025/what-is-mcp-a-simple-guide-to-model-context-protocol-for-salesforce-admins

97 NotebookLM Review 2025: AI Tool for Researchers
https://effortlessacademic.com/notebook-lm-googles-newest-academic-ai-tool/

98 225 Introducing Google's NotebookLM: The AI-powered research assistant
https://its.appstate.edu/news/introducing-google%E2%80%99s-notebooklm-ai-powered-research-assistant

99 100 176 177 178 179 180 181 186 187 Information suppression in large language models: Auditing, quantifying, and characterizing censorship in DeepSeek - ScienceDirect
https://www.sciencedirect.com/science/article/abs/pii/S0020025525008357

101 Model Context Protocol (MCP): Security Risks & Controls
https://www.redhat.com/en/blog/model-context-protocol-mcp-understanding-security-risks-and-controls

102 Model Context Protocol (MCP) - Astrix Security
https://astrix.security/glossary/model-context-protocol-mcp/

116 Quantum Virtual Machine | Cirq
https://quantumai.google/cirq/simulate/quantum_virtual_machine

117 QVM - Rigetti Computing
https://www.rigetti.com/applications/qvm

118 119 [2406.18410] Scaling Quantum Computations via Gate Virtualization
https://arxiv.org/abs/2406.18410

120 121 122 129 AlphaQubit: Google’s research on quantum error correction
https://blog.google/technology/google-deepmind/alphaqubit-quantum-error-correction/

138 139 Quantum Adaptive Self-Attention for Quantum Transformer Models
https://arxiv.org/html/2504.05336v2

140 Our new Quantum Virtual Machine will accelerate research and help ...
https://blog.google/technology/research/our-new-quantum-virtual-machine-will-accelerate-research-and-help-people-learn-quantum-computing/

141 DeepSeek and the Evolution of Large Language Models - KKR
https://www.kkr.com/insights/deepseek-large-language-models

142 Understanding Modern LLMs via DeepSeek
https://planetbanatt.net/articles/deepseek.html

144 145 146 169 170 171 195 196 197 198 DeepSeek Technical Analysis — (1) Mixture-of-Experts | by Jinpeng Zhang | Medium
https://dataturbo.medium.com/key-techniques-behind-deepseek-models-10x-efficiency-1-moe-9bd2534987c8

159 Using FP8 and FP4 with Transformer Engine - NVIDIA Documentation
https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/fp8_primer.html

160 Floating-Point 8: An Introduction to Efficient, Lower-Precision AI ...
https://developer.nvidia.com/blog/floating-point-8-an-introduction-to-efficient-lower-precision-ai-training/

188 Characterizing State Space Model (SSM) and SSM-Transformer ...
https://arxiv.org/html/2507.12442v2

205 Deepseek-v3 vs other LLMs: what's different
https://nebius.com/blog/posts/deepseek-v3-vs-other-llms

206 207 DevOps Meets AI: Evaluating the Performance of Leading LLMs
https://engineering.harness.io/devops-meets-ai-evaluating-the-performance-of-leading-llms-042b77594b49

209 Multi-Agent Systems for Enterprise Workflows - YouTube
https://www.youtube.com/watch?v=ECxZBb5ys_Y

210 211 How LLMs can supercharge your DevOps automation - Medium
https://medium.com/@hadiazouni/how-llms-can-supercharge-your-devops-automation-8917a81b16a9

212 213 [2502.20825] LADs: Leveraging LLMs for AI-Driven DevOps - arXiv
https://arxiv.org/abs/2502.20825

214 Introducing devops-slm-v1: A Small Language Model for ... - LinkedIn
https://www.linkedin.com/posts/prashant-lakhera-696119b_devops-ai-llm-activity-7371607507829493760-78tv

215 LLMOps is the new DevOps, here's what every developer must know
https://langwatch.ai/blog/llmops-is-the-new-devops-here-s-what-every-developer-must-know

216 217 The Best Open Source LLM For Legal Document Analysis In 2025
https://www.siliconflow.com/articles/en/best-open-source-LLM-for-Legal-Document-Analysis

218 219 Large Language Models for Regulatory Compliance - Regology
https://www.regology.com/blog/unlocking-the-power-of-ai-large-language-models-for-regulatory-compliance

220 221 The Dawn of a New Era of Compliance
https://law.mit.edu/pub/thedawnofaneweraofcompliance

222 AI and Data Protection: Strategies for LLM Compliance and Risk ...
https://www.proofpoint.com/us/blog/dspm/ai-and-data-protection-strategies-for-llm-compliance-and-risk-mitigation

223 224 What you need to know about the Model Context Protocol (MCP)
https://www.merge.dev/blog/model-context-protocol

226 227 228 229 230 231 232 233 LAVE: LLM-Powered Agent Assistance and Language Augmentation for Video Editing
https://www.dgp.toronto.edu/~bryanw/lave/

234 AI-Powered Video Editing Trends in 2025 | by VIDIO | Medium
https://medium.com/@vidio-ai/ai-powered-video-editing-trends-in-2025-54461f5d17e2

235 NotebookLM adds Deep Research and support for more source types
https://blog.google/technology/google-labs/notebooklm-deep-research-file-types/

236 237 248 7 shortages challenging AI data center expansion - Insights2Action
https://action.deloitte.com/insight/4719/7-shortages-challenging-ai-data-center-expansion

238 239 240 241 242 243 244 258 259 286 287 Microsoft has AI GPUs “sitting in inventory” because it lacks the ...
https://www.datacenterdynamics.com/en/news/microsoft-has-ai-gpus-sitting-in-inventory-because-it-lacks-the-power-necessary-to-install-them/

245 246 247 US data centers' energy use amid the artificial intelligence boom
https://www.pewresearch.org/short-reads/2025/10/24/what-we-know-about-energy-use-at-us-data-centers-amid-the-ai-boom/

249 Data Drain: The Land and Water Impacts of the AI Boom
https://www.lincolninst.edu/publications/land-lines-magazine/articles/land-water-impacts-data-centers/

250 'Roadmap' shows the environmental impact of AI data center boom
https://news.cornell.edu/stories/2025/11/roadmap-shows-environmental-impact-ai-data-center-boom

251 No, AI Data Centers Are Not Driving Up Electricity Costs
https://reason.com/2025/12/23/the-data-center-price-myth/

252 253 Three Mile Island nuclear power plant to return as Microsoft signs 20 ...
https://www.datacenterdynamics.com/en/news/three-mile-island-nuclear-power-plant-to-return-as-microsoft-signs-20-year-835mw-ai-data-center-ppa/

254 255 Nuclear Powered Data Centers: Microsoft Bets on SMRs to Fuel the ...
https://www.captechu.edu/blog/nuclear-powered-data-centers-microsoft-bets-smrs-fuel-cloud

256 Amazon, Google, Meta and Microsoft go nuclear - Trellis Group
https://trellis.net/article/amazon-google-meta-and-microsoft-go-nuclear/

257 266 World Nuclear Association Welcomes Microsoft Corporation as ...
https://world-nuclear.org/news-and-media/press-statements/world-nuclear-association-welcomes-microsoft-corporation-as-newest-member

262 263 Tech Trends 2025: Big, Bold and Abounding for C-Suites and IT ...
https://www.infinidat.com/en/blog/tech-trends-2025-big-bold-and-abounding-c-suites-and-it-leaders

264 265 303 304 EU AI Act News: Rules on General-Purpose AI Start ... - Mayer Brown
https://www.mayerbrown.com/en/insights/publications/2025/08/eu-ai-act-news-rules-on-general-purpose-ai-start-applying-guidelines-and-template-for-summary-of-training-data-finalized

267 268 269 270 273 Towards higher-dimensional structured light | Light: Science & Applications
https://www.nature.com/articles/s41377-022-00897-3?error=cookies_not_supported&code=4de3fc85-ce90-41c1-bab4-d063205ed206

271 Towards higher-dimensional structured light - Nature
https://www.nature.com/articles/s41377-022-00897-3

272 274 275 Formation and Controlling of Optical Hopfions in High Harmonic ...
https://link.aps.org/doi/10.1103/PhysRevLett.133.133801

276 277 Dynamics of photonic toroidal vortices mediated by orbital angular ...
https://www.science.org/doi/10.1126/sciadv.adz0843

278 Imaging performance evaluation of large photon sieves with high ...
https://preprints.opticaopen.org/articles/preprint/Two_high-efficiency_methods_for_the_imaging_performance_evaluation_of_large_photon_sieves/27424107

279 280 281 Fast high-resolution computer-generated hologram computation ...
https://opg.optica.org/ao/abstract.cfm?uri=ao-51-30-7303

282 283 284 Review of computer-generated hologram algorithms for color ...
https://www.nature.com/articles/s41377-022-00916-3

285 GPU-accelerated T-matrix algorithm for light-scattering simulations
https://www.sciencedirect.com/science/article/abs/pii/S0021999112001490

296 297 NeurIPS 2025 LLM Evaluation Workshop - OpenReview
https://openreview.net/group?id=NeurIPS.cc/2025/Workshop/LLM_Evaluation

298 299 [N] Machine Learning Reproducibility Challenge (MLRC) 2025 ...
https://www.reddit.com/r/MachineLearning/comments/1mhn9lc/n_machine_learning_reproducibility_challenge_mlrc/

300 General-Purpose AI Models in the AI Act – Questions & Answers
https://digital-strategy.ec.europa.eu/en/faqs/general-purpose-ai-models-ai-act-questions-answers

301 302 Foundation Models under the EU AI Act - Stanford CRFM
https://crfm.stanford.edu/2024/08/01/eu-ai-act.html