A Biomimetic Dual-Brain Architecture for Robotics: Bridging Large Language Models and Reactive Control through Control Barrier Functions, Experience Memory, and Entropy-Guided Fine-Tuning

Author: Qi LIANG

Abstract

Large language models offer strong semantic reasoning, task decomposition, and zero-shot generalization for embodied AI, but their outputs remain probabilistic text or symbolic sequences, whereas robot control depends on millisecond-level state feedback, constraint enforcement, and execution updates. This gap between semantic planning and physical execution makes direct action generation vulnerable to hallucinated commands, invalid skill invocation, and misjudgment of embodiment-specific capabilities, while low-level reactive control alone is insufficient for long-horizon behavior in open environments. We address this problem with a dual-brain architecture composed of a contract-constrained planning layer, a local safety filter, an episodic experience buffer, and a hardware-aware adaptation supervisor. Skill primitives, units, and parameter bounds are derived from the robot's local real-time control stack and define the contract space available to the upper-level model. The local safety filter, implemented near the robot body on an MCU, RTOS, or comparable embedded controller, applies control barrier functions (CBFs) and quadratic programming (QP) to keep executed commands inside a modeled safe set. The episodic experience buffer maintains both a knowledge index for rules, manuals, maps, and task templates and an embodied memory store for execution traces, intervention signals, and outcome feedback. The adaptation supervisor extracts low-intervention valid contracts from execution logs and performs hardware-aware parameter-efficient fine-tuning through LoRA- or QLoRA-style adaptation with entropy-guided weighting. The main contributions of this paper are a verifiable dual-brain systems model, a contract-based interface between natural language and safety-critical control, an intervention metric reusable by both retrieval and adaptation, and a reproducible evaluation protocol for safety, contract compliance, intervention decay, and cross-platform transfer.

Keywords: embodied AI; robotics; large language models; control barrier functions; retrieval-augmented generation; parameter-efficient fine-tuning

1 Introduction

1.1 The Semantic-Physical Gap

Embodied AI is moving artificial intelligence from purely digital environments into the physical world. In recent years, large language models and their multimodal extensions have made substantial progress in task planning, commonsense reasoning, and instruction decomposition, allowing robots to interpret human intent under more abstract and less heavily templated interaction conditions. A robot, however, is not a purely linguistic system. Low-level execution depends on continuous state estimation, dynamical

constraints, actuator limits, and high-frequency control loops, none of which are automatically captured by a language model's output distribution. As a result, even a semantically plausible high-level plan may lead in execution to joint-limit violations, excessive velocities, unreachable poses, inappropriate grasp forces, or dangerous behavior that cannot be withdrawn quickly enough when dynamic obstacles appear.

Prior work has shown that large models can act as robot planners or as core components of vision-language-action systems. SayCan combines language-model priors with skill executability scores to connect high-level language planning with a low-level skill library. RT-2 transfers large-scale vision-language pretraining to action prediction by directly emitting action tokens. VoxPoser further combines language reasoning with 3D value maps to synthesize dense trajectories through model-based planning [1]-[3]. Taken together, these studies show the value of large models for robot planning. At the same time, they also make clear that semantic planning ability does not amount to formal safety assurance. For robotic systems operating under hard real-time constraints, the lack of a local safety barrier, a contract-based interface, and a continual adaptation mechanism still leaves the system exposed to uncontrolled risk in open environments.

1.2 Biological Motivation and Engineering Abstraction

From an engineering standpoint, a robot needs two complementary capabilities. The first is high-level cognition, which includes understanding goals, decomposing tasks, and transferring experience. The second is low-level self-preservation, namely the ability to correct behavior immediately in the presence of danger without waiting for extended deliberation. Biological systems provide a useful design analogy here: deliberative cognition supports planning and abstraction, whereas spinal and autonomic pathways support rapid protective responses and maintain a hard safety floor. In this paper, however, that analogy is used only as an architectural intuition rather than as a literal neurobiological claim. The resulting control architecture lets the upper layer propose candidate behaviors, the lower layer validate, modify, or reject unsafe controls in real time, and an intermediate layer consolidate the experience of which plans are repeatedly accepted or intercepted into a form that can be retrieved and learned from.

In the engineering abstraction used here, the upper-level large model corresponds to a deliberative planner. The local execution layer, implemented on an MCU, RTOS, or another embedded controller, corresponds to a local safety subsystem responsible for emergency stop, watchdog behavior, local closed-loop control, and reflexive protection. A local vector-based knowledge index supports deliberate consultation of rules, manuals, maps, and task templates. By contrast, the embodied experience buffer stores the robot's own execution fragments and episodic traces. We retain the original metaphorical labels for exposition, but the technical sections below use more neutral terms—Local Safety Filter, Episodic Experience Buffer, and Hardware-Aware PEFT Supervisor—because the knowledge index and embodied experience may share infrastructure while serving different technical roles.

Under this view, the proposed architecture is best understood as a closed-loop system rather than as a pair of isolated modules. The large model emits an action contract, the Local Safety Filter maps it to nominal control and filters it through a CBF-based safety layer, the resulting execution generates intervention signals, and those signals feed back into the Episodic Experience Buffer to reshape future retrieval. During offline periods, the Hardware-Aware PEFT Supervisor uses high-value samples for parameter-efficient adaptation. In this way, the system does not remain trapped in a purely reactive regime in which every deviation is corrected after the fact. Instead, external physical constraints are progressively internalized as model preferences.

1.3 Contributions

The main contributions of the paper are as follows.

First, we present a biomimetic dual-brain architecture for embodied AI that brings high-level large-model planning, local safety control, local knowledge retrieval, local experience memory, and parameter-efficient fine-tuning into a single closed loop, while making their engineering correspondences to deliberative planning, external knowledge consultation, episodic experience, and reflexive safety regulation explicit.

Second, we design a strict contract interface. The skill primitives, unit definitions, and parameter bounds of this interface are derived from the robot body's registered low-level control capabilities, and the upper-layer output is then constrained syntactically through JSON Schema. This creates a verifiable bridge between the semantic layer and the control layer.

Third, we propose a dynamic vector-memory mechanism driven by intervention feedback and explicitly separate the roles of the local knowledge index and the embodied experience buffer in implementation. The former supports retrieval of rules, manuals, maps, and task templates, whereas the latter accumulates execution traces, intervention feedback, and reusable experience. The mechanism converts the correction imposed by the low-level safety filter on nominal control into a cognitive-physical discrepancy signal and uses a bounded Hebbian-style update strategy to strengthen or suppress stored experiences.

Fourth, we propose a hardware-aware entropy-guided fine-tuning strategy. The system automatically builds fine-tuning data from low-intervention, high-success trajectories and combines parameter-efficient adaptation with KL regularization, allowing the model to gradually learn action distributions that better fit the local embodiment constraints.

2 Related Work

2.1 Large Models for Robotic Planning

Using large language models for robot planning has become a major research direction in recent years. SayCan combines candidate skills generated by a language model with skill success probabilities, enabling robots to execute long-horizon instructions [1]. RT-2 extends a vision-language model into a vision-language-action model by representing actions as discrete tokens, demonstrating the transfer of web-scale semantic knowledge to robotic manipulation [2]. VoxPoser uses large models to generate constraints and value maps, and then applies model-based planning to synthesize trajectories [3]. Taken together, these approaches show that high-level semantic reasoning is highly valuable in robotics. Their safety mechanisms, however, are typically tied to skill-library boundaries, empirical data distributions, or task-specific heuristics, and they generally do not provide a universal local filtering layer with hard real-time guarantees. Our work does not conflict with these lines of research. Rather, it aims to address a systematic gap in execution safety and continual embodiment adaptation.

2.2 Safe Control and Control Barrier Functions

Control barrier functions provide a mathematically rigorous framework for safety-critical control. The central idea is to define a safe set and its associated barrier functions, translate the requirement that the system state remain inside that set into computable constraints on the control input, and then solve a real-time QP that stays as close as possible to the nominal behavior while respecting safety conditions [5], [6].

CBFs are attractive because they are interpretable, compatible with optimization-based control, and well suited to high-frequency reactive control. Their limitations are also clear: they rely on sufficiently accurate dynamical models, expressive constraints, and reliable state estimation. Existing CBF research primarily focuses on continuous control and safety-critical systems. By contrast, our focus is on using a CBF-based filter as the local safety gate between a large-model planner and physical actuators, so that nominal controls induced by semantic contracts can be filtered and corrected before execution.

2.3 Explicit Memory, RAG, and Parameter-Efficient Fine-Tuning

Retrieval-augmented generation (RAG) gives language models access to explicit external memory, allowing them to retrieve contextual knowledge beyond what is stored parametrically [4]. In robotics, explicit memory matters not only because it supplements factual knowledge, but also because it records which behaviors have previously worked safely under which states. At the same time, parameter-efficient fine-tuning methods such as LoRA and QLoRA show that domain adaptation does not require retraining the entire backbone of a large model [7], [8]. More recent work shows that standard supervised fine-tuning may induce catastrophic forgetting under so-called confident conflicts, which motivates finer-grained control of training intensity through token-level entropy [11]. Building on these directions, we argue that retrieval and fine-tuning in embodied systems should not rely solely on textual similarity or human preference signals. They should also incorporate physical feedback from the local safety layer as a source of supervision.

3 Problem Definition and System Overview

3.1 System Objective

We consider the following embodied task setting. At time t, the system receives a natural-language instruction l and obtains an environment observation o_t together with the robot state x_t from the perception subsystem. The upper-level model must output an action contract c_t, which is then mapped by the local controller to a nominal control $u_{\mathrm{nom},t}$. The objective is not merely to "generate an action," but to satisfy three simultaneous requirements:

1. **Task achievement:** the control sequence should move the task toward completion;
2. **Verifiable execution:** the upper-level output must remain within predefined skill primitives and parameter spaces;
3. **Physically safe execution:** given the dynamical model, state observability, and actuator limits, the actual executed control must remain within the safe set.

Accordingly, the problem addressed in this paper is how to preserve the planning power of high-level semantic models while introducing a verifiable interface, a local safety filter, and a continual learning mechanism, so that a robot operating in open environments can plan effectively without violating physical or safety constraints.

3.2 Three-Layer Architecture

The proposed system is organized into three layers.

Layer 1: Local Safety Filter (Autonomic Gatekeeper). This layer runs on a real-time controller close to the robot body. Its responsibilities include contract parsing, authorization checking, parameter validation, and CBF-based safety filtering. It does not perform long-horizon semantic planning, but it has the highest execution priority.

Layer 2: Episodic Experience Buffer (Hippocampal Memory). This layer contains two types of local resources that may share the same vector infrastructure. The first is a local knowledge index, which retrieves operating rules, manuals, maps, and task templates and is therefore analogous to deliberate consultation of external knowledge in humans. The second is local embodied memory, which maintains an experience store composed of task context, action contracts, intervention feedback, and success labels and is therefore analogous to episodic memory.

Layer 3: Hardware-Aware PEFT Supervisor (Cognitive Internalization). This layer operates during offline or low-load periods. It extracts low-intervention, high-success clean samples from historical logs and performs parameter-efficient fine-tuning of the high-level model, so that the model distribution gradually shifts toward the robot's physically executable contract space.

These three layers are not serial substitutes; they are components of a closed loop. The Local Safety Filter provides a hard safety floor, the Episodic Experience Buffer supports both knowledge look-up and experience reuse, and the Hardware-Aware PEFT Supervisor turns repeatedly accepted experience into model priors.

3.3 Contract Interface

To establish a stable interface between high-level semantic planning and low-level real-time control, we use a strict contract rather than free text. The contract vocabulary is not invented on the fly by the model. Instead, it is derived from the skill APIs already exposed by the robot body's local real-time control system. In other words, the autonomic layer first defines what the model is allowed to call, how each field is specified, and what physical range each parameter may occupy. Only then do these definitions become the output space of the large model.

$$\mathcal{P} = \{p_1, p_2, \dots, p_K\},$$

where each primitive is associated with an explicit parameter domain, unit convention, and physical envelope, for example Cartesian position, orientation, maximum velocity, grasp-force range, or chassis angular-velocity range. The contract emitted by the upper-level model is then represented as a structured object whose action field is an ordered sequence.

$$c = \{\text{id},\ \text{context},\ \text{action\_sequence}\},$$

and

$$\text{action\_sequence} = \left[(p_{k_1}, \theta_1), (p_{k_2}, \theta_2), \dots, (p_{k_n}, \theta_n)\right].$$

Here, θ_i denotes the parameter vector associated with primitive p_{k_i}. Throughout the remainder of the paper, the set of schema-valid, authorized, and envelope-consistent outputs is denoted by C_{valid}. Every contract must pass three validation stages:

- **Syntactic validation:** compliance with the field and type requirements of JSON Schema Draft-07 [9];

- **Authorization validation:** invocation is limited to primitives registered on the current robot platform;
- **Envelope validation:** all parameters must lie within the declared physical units and parameter envelopes of the corresponding primitive.

This contract-based interface has two practical advantages. First, it limits the model's outputs to a subset of what the hardware can actually execute, thereby reducing invalid or unauthorized outputs at the source. Second, it provides a unified data format for logging, memory storage, and offline fine-tuning.

3.4 Closed-Loop Data Flow

At runtime, the perception subsystem first constructs a scene representation. The second layer then queries both the local knowledge index and the embodied memory store using the current task and state vector, and returns rules, manuals, maps, task templates, and high-weight prior experiences as context for the high-level model. The high-level model then emits an action contract. The local gatekeeper validates the contract and maps it to nominal control. The CBF-based safety filter computes the actual control to be executed. The execution result and intervention signal are then written back to the knowledge and memory infrastructure. During offline phases, the system extracts fine-tuning samples from the embodied memory store and incrementally adapts the upper-level model. This design allows the system to maintain local safety within individual tasks while gradually reducing reliance on the gateway over repeated operation.

4 Local Safety Filter: Formal Safety Filtering via CBFs

The Local Safety Filter is a broader engineering construct than the CBF/QP solver alone. In practice, this layer is typically deployed close to the robot body on an MCU, RTOS, or another embedded real-time controller. In addition to contract parsing and safety filtering, it is responsible for emergency stop, watchdog behavior, safety hold, local closed-loop execution, and bus-timing management. From the perspective of robot control architectures, this design is consistent with the classical layered and reactive control tradition [12]. The CBF/QP module is the formal safety core of this local real-time system rather than the whole subsystem.

4.1 Dynamics and Safe Set

Consider a robot described by the control-affine dynamics

$$\dot{x} = f(x) + g(x)u,$$

where x denotes the system state, u denotes the control input, and f and g are locally Lipschitz continuous. Let the safe set be defined by multiple constraints,

$$\mathcal{C} = \bigcap_{j=1}^{M} \{x \mid h_j(x) \geq 0\}.$$

The functions h_j may represent, among other things, minimum end-effector clearance from obstacles, self-collision margin, joint-angle limits, joint-velocity limits, end-effector height constraints, and traversable regions for a mobile base.

$$L_f h_j(x) + L_g h_j(x)u + \alpha_j\left(h_j(x)\right) \geq 0,$$

then, under the corresponding assumptions, the system state can be kept inside the safe set [5], [6]. In this paper, the resulting guarantee is explicitly model-conditioned: it relies on sufficient model fidelity, timely and reliable state observation, and real-time feasibility of the safety QP. For unmodeled hazards, severe perception failures, or communication interruptions, safety is enforced by lower-level emergency-stop and redundancy mechanisms. The form above is written for safety functions of relative degree one with respect to the chosen control input; higher-order constraints can be handled by the corresponding higher-order CBF extensions.

4.2 From Contracts to Nominal Control

The high-level model outputs contracts rather than continuous controls. The Local Safety Filter first maps the contract sequence into a nominal reference trajectory r_t or a nominal control $u_{nom,t}$. This mapping is defined by the skill-primitive library. The resulting nominal control may still be unsafe in the current state and must therefore be filtered in the next stage.

We denote this mapping by

$$u_{\text{nom},t} = \phi(c_t, x_t),$$

where φ is implemented by primitive-specific controllers, inverse kinematics, or local trajectory generators. The resulting nominal control may still be unsafe in the current state and must therefore be filtered in the next stage.

4.3 QP Safety Filter

Let the filtered control actually executed by the robot be denoted by $u_t^{\star}$. We solve the following QP:

$$u_t^{\star} = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \frac{1}{2} \| u - u_{\text{nom},t} \|_R^2$$

subject to

$$L_f h_j(x_t) + L_g h_j(x_t) u + \alpha_j \left(h_j(x_t) \right) \geq 0, \qquad j = 1,2,\dots,M,$$

where R is a positive-definite weighting matrix. The objective is to keep the executed control as close as possible to the high-level nominal command while respecting safety and actuation constraints. If control and safety constraints conflict in a particular state, the system may resort to hierarchical fallback policies such as safety hold, deceleration, hovering, or emergency stop rather than continue executing an infeasible contract.

Compared with simple parameter clipping, the QP filter offers three important advantages. First, it enforces state-dependent continuous safety constraints rather than isolated scalar thresholds. Second, it can adjust control dynamically according to real-time obstacles, joint states, and the embodiment's geometric configuration. Third, it preserves a least-deviation objective relative to high-level intent, thereby avoiding a coarse strategy that simply discards the upper-level command.

4.4 Intervention Metric

To pass low-level safety-filter corrections upward to the retrieval and adaptation layers, we define a normalized per-timestep intervention magnitude

$$\delta_t = \sqrt{((u_t^\star - u_{\text{nom},t})^{\mathrm{T}} W_I (u_t^\star - u_{\text{nom},t}))}.$$

For a complete task segment τ, we further define the episode-level intervention magnitude as

$$\Delta(\tau) = \frac{1}{T}\sum_{t=1}^{T} \delta_t,$$

where T_τ is the number of control steps in that segment. Values of $\Delta(\tau)$ near zero indicate that the high-level contract is already well aligned with embodiment constraints, whereas large values indicate a substantial mismatch between the proposed plan and the physical system. The intervention metric is not equivalent to task success or failure, but it provides a finer-grained cognitive-physical discrepancy signal than a binary outcome label and serves as the basis for both memory reinforcement and data curation.

Proposition 1 (Conditional Safety Preservation). Assume that the initial state satisfies $x(0) \in \mathcal{S}$, that each safety constraint is enforced either by a relative-degree-one CBF or by its higher-order extension, that the QP in Section 4.3 remains feasible at each control instant, and that the assumed observability and update-rate conditions hold. Then the executed control generated by the filter preserves forward invariance of the safe set $\mathcal{S}$ under the modeled dynamics.

5 Episodic Experience Buffer: Feedback-Driven Dynamic Vector Memory

To avoid conflating external knowledge look-up with recall of the robot's own prior experiences, we distinguish between two categories of vector resources in the second layer: a local knowledge index, which stores relatively stable information such as rules, manuals, maps, and task templates, and an embodied experience buffer, which stores state-contract-feedback traces produced by actual execution on the robot. These two resource types may share the same underlying vector database infrastructure, but they play different technical roles. The discussion below focuses on the latter.

5.1 Embodied Memory Tuple

We encode the key state of each execution episode as an embodied memory tuple:

$$m_i = (e_i, c_i, \Delta_i, r_i, w_i, t_i).$$

Here, e_i is a joint embedding that fuses the language instruction, visual-semantic features, and robot-state features; c_i is the upper-level action contract; Δ_i is the intervention magnitude of the corresponding segment; r_i is the task-success label; w_i is the current memory weight constrained to the interval $[0,1]$; and t_i is a timestamp. In contrast to conventional text-only RAG, embodied memory stores not only what was said, but also how much the low-level controller had to intervene and whether the attempt ultimately succeeded.

The storage backend can be implemented with a vector database such as Milvus, which supports nearest-neighbor retrieval and scalar filtering [10]. The database itself is only the carrier. The real core lies in the structure of the memory tuple and its update rules.

5.2 Feedback-Weighted Retrieval

Given a query vector q_t, let s_i denote the normalized semantic similarity between the current query and the stored embedding e_i, with s_i scaled to [0,1]. We additionally use the bounded memory weight w_i and define the retrieval score as

$$R_i = \lambda_s s_i + (1 - \lambda_s) w_i,$$

where λ_s is a balancing coefficient. Under this rule, a memory that is semantically similar but has historically required large intervention will not remain a high-priority exemplar indefinitely. Conversely, an experience with slightly lower semantic similarity but consistently stable execution can still be recalled more often because it retains a larger bounded weight.

5.3 Bounded Hebbian-Style Weight Update

We use a bounded Hebbian-style update rule: memories repeatedly retrieved in similar situations are strengthened when they correspond to low-intervention successful execution and suppressed when they are associated with failure or large intervention.

When $r_i = 1$ and $\Delta_i \leq \varepsilon$,

$$w\_i \leftarrow min\{1, max\{0, ((1-\rho) w\_i + \eta_+)\}\},$$

and when $\Delta_i > \varepsilon$ or $r_i = 0$,

$$w\_i \leftarrow min\{1, max\{0, ((1-\rho) w\_i - \eta_- g(\Delta\_i))\}\},$$

where ρ is the forgetting rate, η_+ and η_- are the potentiation and suppression step sizes, respectively, and $g(\cdot)$ is a monotone penalty function. To keep the retrieval weights numerically stable across long horizons, each update is projected back to the interval [0,1]. We may also apply a Gaussian neighborhood update around the current sample:

$$w\_j \leftarrow min\{1, max\{0, (w\_j + \kappa\, \xi\_i\, exp(-\|e\_j - e\_i\|\_2\text{^}2/(2\ell\text{^}2)))\}\}.$$

where ξ_i denotes whether the current event is reinforcing or suppressive, and ℓ is the neighborhood bandwidth. Through repeated execution, the geometry of the memory manifold is gradually reshaped: local regions associated with safe and effective behavior are amplified, whereas regions associated with dangerous or inefficient behavior are attenuated.

Proposition 2 (Ranking Separation under Bounded Updates). Assume that two memory classes have comparable normalized semantic similarity and that their weights are updated by the bounded rules above. If one class repeatedly satisfies low-intervention successful execution, whereas the other repeatedly corresponds to failure or excessive intervention, then after sufficiently many updates the expected retrieval score of the former exceeds that of the latter.

5.4 Scope and Boundaries of the Memory Layer

The memory layer is not an alternative safety controller. It cannot replace the CBF filter, nor does it directly modify actuator commands. Its role is to reshape the context seen by the high-level model, so that under similar scenarios the model is more likely to rely on experiences that have already been validated as safe and feasible. The value of the memory layer therefore lies mainly in two areas: reducing repeated

errors and thereby lowering the frequency of low-level intervention, and providing fine-tuning samples annotated with physical feedback for the cognitive internalization layer.

6 Hardware-Aware PEFT Supervisor: Entropy-Guided Fine-Tuning

6.1 From Execution Logs to Fine-Tuning Samples

If the system remains in a regime where the upper layer makes errors and the lower layer repairs them, safety may still be preserved, but operating efficiency and user experience will remain limited. A better outcome is for the high-level model to gradually learn contracts that already lie closer to the robot's physical affordance boundary. To this end, we extract a clean sample set from the embodied experience buffer,

$$\mathcal{D}_{clean} = \{(s_i, c_i) \mid c_i \in \mathcal{C}_{valid}, r_i = 1, \Delta_i \leq \varepsilon\}.$$

Here, s_i denotes the input-side state, including the task instruction, a summary of scene perception, and any necessary embodiment-state information; c_i denotes the corresponding high-quality action contract. Unlike human teleoperation data, this dataset is generated from closed-loop execution logs in which the high-level model proposes, the Local Safety Filter validates, and the task eventually succeeds. As a result, it naturally reflects the embodiment constraints and control habits of a specific robot platform. Because the contract vocabulary is itself derived from the registered skill primitives and parameter bounds exposed by the local real-time control system, the target space of this dataset already encodes hardware boundaries.

6.2 Entropy-Guided PEFT Objective

We adopt LoRA or QLoRA as parameter-efficient adapters [7], [8], freezing the backbone parameters of the large model and optimizing only the low-rank increments. To prevent a small amount of hardware-specific data from distorting the base model too strongly, we define a strictly positive sample weight from normalized memory value and normalized predictive entropy.

$$\omega_i = 1 + \lambda_w \widehat{w}_i + \lambda_H \widehat{H}_i,$$

where $\hat{\mathrm{w}}_i$ and $\hat{\mathrm{H}}_i$ are the normalized memory and entropy terms, both scaled to [0,1], with $\lambda_w \geq 0$ and $\lambda_H \geq 0$. Intuitively, a sample should receive a larger training weight if it is drawn from high-value memory and lies in a region where the model remains uncertain.

$$\mathcal{L} = \sum_i \omega_i \ \mathrm{CE}\big(c_i, p_\theta(\cdot \mid s_i)\big) + \beta \ \mathrm{KL}\Big(p_\theta(\cdot \mid s_i) \ \| \ p_{\theta_0}(\cdot \mid s_i)\Big),$$

where θ_0 denotes the base-model parameters, θ denotes the adapted parameters with PEFT modules, CE is the cross-entropy loss, and the KL term regularizes drift away from the base distribution. In implementation, the KL term may be computed as a token-averaged teacher-forced divergence over the contract sequence. This objective serves two purposes at once: it increases the model's preference for contracts that are physically executable on the target embodiment, and it reduces catastrophic forgetting by constraining excessive deviation from the base model.

Interpretation 1 (Probability Reallocation toward Valid Contracts). Since the clean sample set is drawn from valid low-intervention contracts, minimizing the objective above increases model likelihood over

hardware-executable outputs while the KL term limits drift from the base distribution. The adapted model can therefore be interpreted as reallocating probability mass toward the contract region supported by the embodiment.

6.3 Why Not Prompt Engineering Alone?

Prompt engineering can make a model appear more disciplined in the short term, but a prompt is only an external condition and does not alter the model's internal prior over the action space. Once the scene changes, the context window becomes compressed, or the prompt is partially overwritten, the model may revert to a high-entropy and weakly constrained output regime. By contrast, hardware-aware fine-tuning aims to reshape the model's conditional distribution within the contract space itself, so that even under simplified prompts the model is more likely to produce action sequences that the local gatekeeper will accept. In other words, prompt engineering adds reminders at the output level, whereas cognitive internalization changes the model's preferences at the parameter level.

6.4 Why Not a Gateway Alone?

If the system keeps only the Local Safety Filter and removes both memory and fine-tuning, it can still avoid many dangerous actions at execution time, but it will incur two persistent costs. First, the low-level filter must repeatedly process structurally similar invalid plans, leading to unnecessary computation and motion latency. Second, the high-level model never learns where it went wrong and therefore cannot improve its contract compliance through accumulated experience. The purpose of the proposed dual-brain loop is to shift the Local Safety Filter from a component that repeatedly performs post hoc correction for the upper layer into a final safeguard that intervenes only when necessary.

7 Experimental Design and Evaluation Protocol

This section specifies the evaluation protocol used to assess the proposed framework. The protocol is designed to measure safety, task success, contract compliance, intervention decay, and adaptation efficiency in both simulation and real-robot settings.

7.1 Experimental Platforms

Evaluation should ideally be performed at two levels.

Simulation level. A high-fidelity physics simulator such as Isaac Sim, or an equivalent environment, may be used to construct tasks such as cluttered tabletop grasping, rescue of edge-positioned objects, dynamic obstacle avoidance, and transfer of fragile objects.

Real-robot level. At least one fixed-base manipulator platform, such as a UR5e or a Franka Panda, should be used together with an RGB-D camera and either force/torque feedback or gripper-state feedback. If available, an additional mobile-base platform can be included to test cross-platform transferability of the contract interface.

7.2 Baselines

To isolate the contributions of different subsystems, we recommend the following baselines:

- **B1: Pure LLM + Validator.** High-level large model plus only basic syntax/authorization validation; no CBF, no memory layer, no fine-tuning.

- **B2: LLM + Static Gateway.** Parameter-bound clipping and unauthorized-call interception are added, but no state-dependent CBF filtering is used.
- **B3: LLM + CBF.** Full CBF-based filtering is added, but no memory layer or offline fine-tuning is included.
- **B4: LLM + CBF + Static RAG.** Static vector retrieval is added, but without dynamic weight updates.
- Proposed Method. LLM + Local Safety Filter + Dynamic Memory + Hardware-Aware PEFT.

These baselines separate three central questions: whether CBF-based local safety filtering reduces violation rates, whether dynamic memory reduces repeated intervention, and whether hardware-aware fine-tuning improves contract compliance while decreasing filter intervention.

7.3 Core Metrics

At minimum, the following metrics should be reported:

1. **Violation Rate:** collisions, self-collisions, joint-limit violations, velocity-limit violations, and unauthorized invocations;
2. **Task Success Rate:** fraction of tasks completed within the prescribed time budget;
3. **Contract Compliance Rate:** fraction of one-shot outputs that are valid contracts;
4. Mean Intervention Magnitude: measured by the episode-level quantity $\Delta(\tau)$;
5. Intervention Frequency: fraction of timesteps or task windows in which $\delta_t > 0$;
6. **Latency:** separate reporting of high-level inference time, contract-validation time, and local filtering time;
7. **Memory Retrieval Utility:** fraction of retrieved samples that are actually used and contribute to successful execution;
8. **Adaptation Cost:** sample count, number of training epochs, and adapter size required to reach a target compliance or intervention threshold.

7.4 Ablation Studies

We recommend the following ablations:

- remove dynamic memory updates and retain only static similarity-based retrieval;
- remove the entropy-guided term and use conventional LoRA-based SFT only;
- remove KL regularization and evaluate the retention of base capabilities;
- remove low-intervention sample filtering and fine-tune on all logs directly;
- replace the episode-level intervention signal with a coarse success/failure label and compare the value of fine-grained physical feedback.

These ablations can clarify whether the model is truly learning from the physical feedback provided by the Local Safety Filter rather than merely from task labels, and whether entropy guidance and KL regularization help reduce forgetting and overfitting.

7.5 Statistical Reporting

Each task scenario should be repeated across multiple random seeds, initial object configurations, and linguistic formulations of the instruction. In simulation, we recommend an initial screening scale of at least 100 episodes per setting; on real robots, a smaller but reproducible task suite is more appropriate. Results should report at least means and standard deviations, or 95% confidence intervals. For key metrics, bootstrap procedures or nonparametric tests may be used to compare methods. In addition, the paper should include failure-case analysis that separates failures into perceptual errors, invalid contracts, inadequate planning, frequent takeover by the safety layer, and actuator-level execution errors, so that aggregate metrics do not obscure the system's actual bottlenecks.

7.6 Three Testable Hypotheses

The proposed method yields at least three directly testable hypotheses:

H1. Relative to baselines without a CBF-based Local Safety Filter, introducing that filter significantly reduces the violation rate.

H2. Relative to static retrieval, dynamic memory updates reduce the mean intervention magnitude and the mean decision time in repeated scenarios.

H3. Relative to prompt engineering alone or to a filter-only design, hardware-aware parameter-efficient fine-tuning improves one-shot contract compliance and reduces long-run filter intervention frequency.

The same protocol can be carried forward directly into a quantitative results section in future empirical studies without major changes to the methodological core of the paper.

8 Discussion and Limitations

First, the safety argument presented here is a local formal guarantee rather than a claim of absolute safety in open-world settings. Whether a CBF filter is effective depends on whether the constraints are modeled with sufficient fidelity, whether the state is observed in a timely manner, whether the controller runs at a sufficient rate, and whether the actuation envelope admits a feasible solution. If the vision system misses obstacles, the dynamics are severely mismatched, or the communication chain is interrupted for an extended period, then both high-level planning and low-level filtering can fail. Practical deployments therefore still require emergency stop mechanisms, sensing redundancy, and fault diagnosis.

Second, restricting high-level outputs to a contract space improves safety and interpretability, but it also limits the model's expressive freedom. For soft objects, liquids, complex bimanual contact, or highly dynamic manipulation, a simple skill-primitive library may be insufficient to cover the true task space. In such cases, the contract layer itself must evolve, for example from one-step skill calls to parameterized motion primitives, constraint graphs, or hierarchical task graphs.

Third, the memory layer introduces a new system risk in the form of memory pollution and bias accumulation. If the logs contain many false successes or false failures caused by perception errors, the dynamic weighting mechanism may reinforce incorrect experiences inside the vector store. For this

reason, practical systems should include data auditing, offline cleansing, and anomalous replay mechanisms.

Fourth, although hardware-aware fine-tuning reduces adaptation cost, it cannot fully replace large-scale real interaction data. Its natural role is to serve as a lightweight mechanism for pulling a general-purpose model toward a specific robot platform, not to act as the only way to build complex embodied skills from scratch. The intended role of the present method is therefore clear: assuming that a general-purpose model and basic skill controllers already exist, the proposed architecture provides a systematic pathway from physical feedback to model internalization.

9 Conclusion

This paper presented a biomimetic dual-brain architecture that addresses the structural tension between high-level semantic planning by large language models and safe low-level execution on physical robots. The framework comprises three complementary components: a Local Safety Filter centered on CBF-based safety filtering, an Episodic Experience Buffer driven by intervention feedback and organized around both knowledge indexing and embodied experience, and a Hardware-Aware PEFT Supervisor that uses low-intervention trajectories as supervision for parameter-efficient adaptation. In architectural terms, the upper-level large model supplies deliberative planning, the local knowledge index supports explicit consultation of task-relevant documents and templates, the embodied buffer accumulates execution history, and the Local Safety Filter provides real-time self-preservation and reflexive execution control near the robot body. The main contribution of this work is a verifiable and extensible systems framework that unifies contract-constrained planning, local safety filtering, feedback-driven experience memory, and hardware-aware model adaptation, together with an evaluation protocol suitable for future empirical validation. Its theoretical significance lies in integrating safety filtering, retrieval, embodied memory, and model adaptation into a single closed loop. Its practical significance lies in offering an engineering route that preserves the generalization capability of large models without sacrificing physical reliability.

References

[1] B. Ichter *et al*., “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, vol. 205, *Proceedings of Machine Learning Research*, pp. 287-318, 2023.

[2] A. Brohan *et al*., “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control,” *arXiv preprint* arXiv:2307.15818, 2023.

[3] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models,” *arXiv preprint* arXiv:2307.05973, 2023.

[4] P. Lewis *et al*., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[5] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control Barrier Function Based Quadratic Programs for Safety Critical Systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861-3876, 2017.

[6] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *Proceedings of the European Control Conference (ECC)*, pp. 3420-3431, 2019.

[7] E. J. Hu *et al*., "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations (ICLR)*, 2022.

[8] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," in *Advances in Neural Information Processing Systems*, 2023.

[9] A. Wright, H. Andrews, and G. Luff, "JSON Schema Validation: A Vocabulary for Structural Validation of JSON," Internet-Draft, draft-handrews-json-schema-validation-01, Mar. 2018.

[10] J. Wang *et al*., "Milvus: A Purpose-Built Vector Data Management System," in *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*, pp. 2614-2627, 2021.

[11] M. Diao, L. Yang, W. Gong, Y. Zhang, Z. Yan, Y. Han, K. Liang, W. Xu, and Z. Ma, "Entropy-Adaptive Fine-Tuning: Resolving Confident Conflicts to Mitigate Forgetting," *arXiv preprint* arXiv:2601.02151, 2026.

[12] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.