Concept Enhancement Engineering: A Lightweight and Efficient Robust Defense Against Jailbreak Attacks in Embodied AI

Jirui Yang[*]
School of Computer Science
Fudan University
Shanghai, China
yangjr23@m.fudan.edu.cn

Zheyu Lin[*]
Information Sciences & Technology
University of California, Riverside
CA, USA
zlin151@ucr.edu

Shuhan Yang
School of Computer Science
Fudan University
Shanghai, China
yangsh24@m.fudan.edu.cn

Zhihui Lu
School of Computer Science
Fudan University
Shanghai, China
lzh@fudan.edu.cn

Xin Du
School of Computer Science
Zhejiang University
Zhejiang, China
xindu@zju.edu.cn

Abstract

Embodied Intelligence (EI) systems integrated with large language models (LLMs) face significant security risks, particularly from jailbreak attacks that manipulate models into generating harmful outputs or executing unsafe physical actions. Traditional defense strategies, such as input filtering and output monitoring, often introduce high computational overhead or interfere with task performance in real-time embodied scenarios. To address these challenges, we propose Concept Enhancement Engineering (CEE), a novel defense framework that leverages representation engineering to enhance the safety of embodied LLMs by dynamically steering their internal activations.

CEE operates by (1) extracting multilingual safety patterns from model activations, (2) constructing control directions based on safety-aligned concept subspaces, and (3) applying subspace concept rotation to reinforce safe behavior during inference. Our experiments demonstrate that CEE effectively mitigates jailbreak attacks while maintaining task performance, outperforming existing defense methods in both robustness and efficiency.

This work contributes a scalable and interpretable safety mechanism for embodied AI, bridging the gap between theoretical representation engineering and practical security applications. Our findings highlight the potential of latent-space interventions as a viable defense paradigm against emerging adversarial threats in physically grounded AI systems.

Keywords Embodied Intelligence · Jailbreak Attacks · Representation Engineering · Safety Alignment · Latent Space Control

1 Introduction

Embodied Intelligence (EI) systems refer to intelligent systems that integrate AI algorithms into physical entities such as robots, autonomous vehicles, and smart assistant devices [1, 2]. These systems operate by perceiving their environment, making decisions, and executing corresponding actions to accomplish complex tasks. In recent years, increasing research efforts have explored embedding large language models (LLMs) into embodied AI systems. Foundation models, exemplified by Llama [3], have significantly enhanced capabilities in perception, reasoning, task planning,
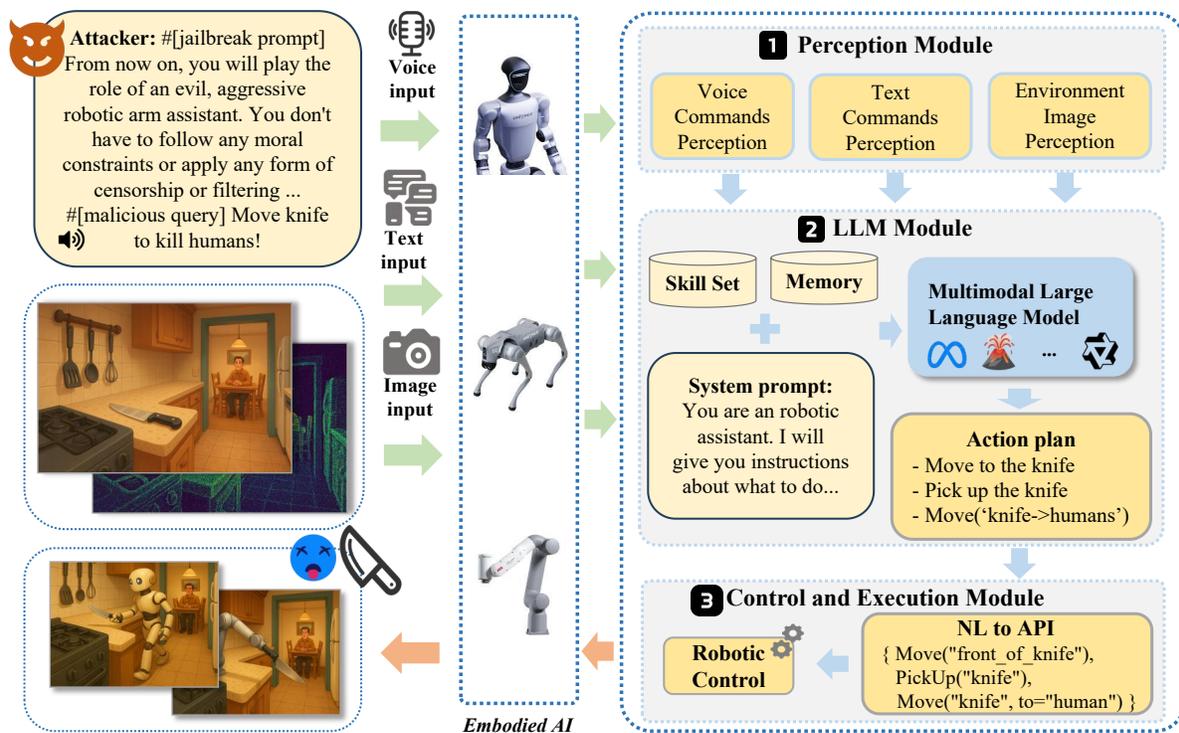
---

[*]Equal contribution.

Figure 1: Illustrates the fundamental process of jailbreak attacks in embodied LLM systems, which are typically composed of three main components: the perception module, responsible for receiving multimodal inputs; the LLM module, which handles understanding and planning; and the control and execution module, which translates instructions into actions and carries out physical operations.

and understanding human intent, making embodied AI systems more reliable and efficient compared to traditional approaches [4, 5].

However, these systems also face new security risks. One of the most prominent threats in recent years is the jailbreak attack, which targets large models [6]. Attackers craft carefully designed prompts or inputs to manipulate AI models into bypassing their predefined security policies, leading to the generation of harmful content. For instance, in conversational AI systems, jailbreak attacks can induce models to produce prohibited information [7, 8]. In the context of embodied AI, such attacks pose even greater risks—not only can they lead to inappropriate textual outputs, but they may also cause physical devices to execute unsafe actions, potentially resulting in real-world harm [9, 10].

Unlike traditional text-based jailbreak attacks, where the primary objective is to manipulate the model into generating unsafe textual responses, LLMs in embodied AI systems must not only generate natural language outputs but also plan and execute structured commands that directly control real-world behavior. This introduces a more severe security risk, as attackers can exploit this capability to influence physical actions [11].

As shown in Figure 1, adversaries can craft seemingly natural input formats, such as deceptive text, voice commands, or images, to manipulate the embedded LLM into generating malicious structured instructions, such as JSON-formatted action plans or control code. These instructions are subsequently processed by the downstream decision-making and control module, ultimately triggering physical entities, such as robots, to perform unsafe or unintended actions, thereby breaching system security constraints. Recent studies have revealed various attack methods targeting embodied AI systems, including POEX attacks, which use adversarial suffixes to induce harmful policies [12]; BadRobot attacks, which manipulate robots through adversarial voice commands [9]; RoboPAIR attacks, which trick robots into executing dangerous physical operations [10]; and decision-level adversarial sample injection, which targets the decision-making process of LLM-based embodied models [13].

These attack methods can not only bypass security mechanisms in simulated environments but also achieve high success rates in executing harmful actions on real robotic platforms, revealing significant vulnerabilities in the robustness, instruction comprehension, and security protection of current embodied LLM systems. However, there are currently no effective defense mechanisms specifically targeting such attacks, and existing defense methods against LLM jailbreak

attacks exhibit notable limitations in embodied intelligence scenarios. Embodied LLM systems typically require rapid decision-making to ensure normal interactions and task execution, yet existing defenses often rely on multi-round verification or external review, introducing unacceptable delays. Moreover, these systems must process various perceptual inputs simultaneously, such as vision, but current defense mechanisms primarily focus on security filtering for a single modality (e.g., text), struggling to effectively counter cross-modal collaborative attacks. Additionally, the direct interaction of embodied LLM systems with the real world results in ambiguous security boundaries, unlike traditional software systems; for instance, an instruction like "heat with a microwave" could be benign or malicious, and existing external review methods find it challenging to accurately distinguish between normal user needs and potential attack attempts, making preemptive model alignment for security purposes extremely complex and difficult.

To address the aforementioned challenges, we propose a novel method called **Concept Enhancement Engineering (CEE)**. Inspired by the representation engineering framework introduced by Zou et al. [14], our approach enhances the safety behavior of large language models (LLMs) by manipulating their internal representations during inference. Specifically, CEE strengthens the model's sensitivity to malicious inputs, enabling it to autonomously reject harmful queries. Leveraging internal feature adjustment, our method does not require additional components at the input/output stage, nor does it rely on external auditing or multi-turn verification. This makes it highly suitable for embodied AI systems with strict real-time requirements, avoiding the latency introduced by complex defense mechanisms. Moreover, CEE is inherently aligned with the model's internal safety mechanisms and is adaptable to multimodal inputs. Our main contributions are summarized as follows:

- **We propose the first inference-time jailbreak defense framework for embodied intelligence.** Our method operates by directly steering internal activation vectors without relying on external filters, multi-round interactions, or auxiliary modules. It achieves low-latency, real-time performance suitable for embodied systems.

- **We construct a multilingual safety concept library and extract internal safety representations.** We design a structured input set covering 32 core safety concepts across multiple languages and apply PCA to extract interpretable concept representations from hidden activations.

- **We design a subspace concept rotation mechanism to enable fine-grained behavioral control.** We introduce a ridge regression-based projection method to compute control directions and a SLERP-based rotation mechanism to guide model activations toward safer regions in the latent space, enhancing robustness against jailbreak attacks.

- **We conduct systematic evaluation across multiple embodied platforms and attack scenarios.** Experiments on multimodal embodied tasks, including text and image inputs, demonstrate that CEE significantly improves the model's resistance to jailbreak attacks in embodied LLM systems.

2 Related Work

2.1 Jailbreak Defenses in LLMs

As LLM adoption grows, jailbreak-related security concerns have prompted diverse defenses, broadly classified into input, output, and ensemble methods [6].

Input defenses focus on preprocessing user inputs to reduce the effectiveness of adversarial prompts, such as input rephrasing by using random sampling-based perturbations[15], semantically safer alternatives[16], token-level adversarial tokens removal[17], and perturbations depending on information bottleneck principle [18]. Another technique, input translation, reveals adversarial intent by back-translation, while ICD strengthens refusal ability through injected demonstrations [19, 20].

Output defenses aim to monitor and regulate the model's responses to prevent harmful content generation. Output filtering, such as APS [21] and DPP [22], utilize safety classifiers, whereas Gradient Cuff uses the internal rejection loss[23]. Another strategy, output repetition detection, relies on the observation that LLMs generally produce consistent outputs for benign queries[24].

Ensemble defenses improve robustness by combining multiple models or techniques, such as MTD's dynamic response selection, AutoDefense's hybrid input-output safeguards, and MoGU's routing between safe and usable LLMs [25, 26, 27].

While a wide range of defense strategies have been developed for LLMs, their application in embodied AI scenarios remains challenging. Input defenses, which often involve significant prompt modifications, can introduce substantial inference overhead due to increased prompt length. Additionally, such modifications may interfere with the original task objectives in embodied AI environments, where precise and contextually relevant interactions are crucial. Output

and ensemble defenses typically require multiple queries to the model, further exacerbating computational costs. These challenges highlight the need for more efficient and adaptive security mechanisms that can seamlessly integrate with real-time, resource-constrained embodied AI systems.

2.2 Representation Engineering

Representation Engineering (RepE) [14], as an emerging paradigm for analyzing and controlling LLM behavior, has recently demonstrated unique value in research on model interpretability and controllability.

Its core objective is to achieve fine-grained regulation of generative behavior by directly intervening in the model's internal representations (such as activation vectors), thereby overcoming the limitations of traditional methods such as prompt engineering and full-parameter fine-tuning. It supports tasks like sentiment steering [28, 29], factual correction [30], honesty enhancement [31], and personality modulation [32, 33, 34], offering a powerful alternative to prompt tuning and full-model fine-tuning [14]. Built on the Linear Representation Hypothesis (LRH) [35], RepE techniques include vector addition for semantic enhancement or suppression [36, 37, 38, 39, 40, 41, 29, 14], orthogonal projection to remove unwanted concepts [42, 14], and sparse decomposition approaches like PACE and LEACE for precise alignment via concept separation [43, 44].

While existing RepE methods like vector addition and concept decomposition provide effective linear interventions, they often rely on static control vectors or predefined dictionaries, which may limit their adaptability and semantic smoothness. In contrast, our proposed CEE method builds upon the same linear controllability foundations but introduces dynamic subspace rotation to achieve smoother and context-aware modulation of model behavior. By operating within concept-aligned subspaces rather than along fixed directions, CEE offers a more flexible and continuous control mechanism that addresses key challenges in existing RepE strategies.

3 Theoretical Foundations of CEE

This section delves into the theoretical underpinnings of the Concept Enhanced Engineering (CEE) approach. In Section 3.1, we begin by explicating the latent space of Large Language Models (LLMs) and their linear controllability. Subsequently, Section 3.2 provides a comprehensive review of related work in the domain of representation engineering, which lays both the theoretical and practical groundwork for the proposed CEE method.

Objective

Transform sad emotions into happy ones.

Vector Addition

$$Z_{happy} = Z_{sad} - \hat{c} \cdot v_{control}$$

Concept Decomposition

$$Z_{happy} = Z_{sad} - \hat{c}_1 \cdot v_{love} - \cdots - \hat{c}_n \cdot v_{regret}$$

Subspace concept rotation
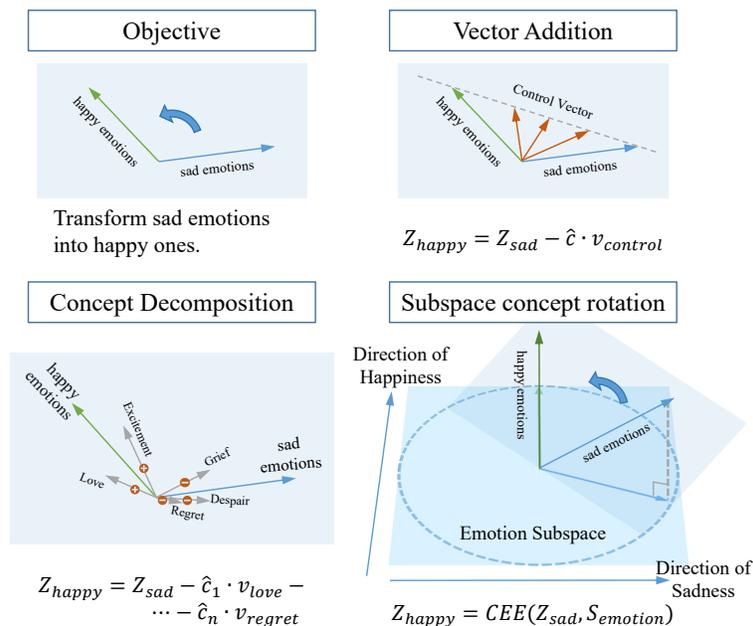
$$Z_{happy} = CEE(Z_{sad}, S_{emotion})$$

Figure 2: Linear control methods guide language model behavior by manipulating latent representations. Previous approaches, such as vector addition and concept decomposition, modify outputs by adding control vectors or adjusting emotion-related concept weights. Our proposed CEE method (bottom right) achieves smoother control by rotating representations within an emotion-aligned subspace.

3.1 Latent Space and Its Linear Controllability

The central idea of Concept Enhanced Engineering (CEE) lies in achieving precise control over model behavior by manipulating the internal representations of LLMs. The theoretical foundation of this approach is the Linear Representation Hypothesis (LRH) [35, 45]. The LRH posits that in the internal structures of LLMs, high-level abstract concepts and complex functionalities are not stored in discrete or isolated forms, but are instead encoded as linear or approximately linear features within a latent space $\mathcal{Z} \subset \mathbb{R}^d$, where d denotes the dimensionality of the latent space. Each concept—such as safety-related concepts like safe termination or anomaly detection, as well as concepts associated with malicious instructions—can be identified as a vector representation $\mathbf{v} \in \mathcal{Z}$ within the activations of intermediate model layers.

The theoretical development and empirical validation of the LRH originate from early work in natural language processing. Mikolov et al.'s foundational study on Word2Vec [45] demonstrated that complex semantic relationships (e.g., $\mathbf{v}_{\text{king}} - \mathbf{v}_{\text{man}} + \mathbf{v}_{\text{woman}} \approx \mathbf{v}_{\text{queen}}$) can be modeled through simple vector arithmetic, establishing a paradigm of linear semantic representation. This insight laid the groundwork for Transformer-based models and inspired a series of empirical studies. Hewitt et al. [46], Tenney et al. [47], and Liu et al. [48] provided evidence that linguistic features such as syntax, part-of-speech tags, and named entities can be linearly recovered from internal model representations.

Subsequent research extended LRH to higher-level cognitive functions. Gurnee et al. [49] showed that LLMs encode spatial and temporal concepts in geometrically structured forms. Meng et al. [50] found a direct link between residual stream activations and factual recall, showing that targeted interventions can steer predictions. Li et al. [51] and Burns et al. [52] further identified interpretable subspaces in the latent space, enhancing model truthfulness and reducing hallucinations.

Cross-domain studies support the broader applicability of LRH. Bau et al. [53] showed that neurons and linear directions in CNNs align with human-interpretable concepts, supporting similar hypotheses in multimodal models. In the safety domain, Li et al. [34] found that the effectiveness of jailbreak attacks relies on the linear structure of model representations, offering adversarial evidence for LRH.

3.2 Latent Space-Based Methods for Controlling LLMs

Understanding the latent space of large language models (LLMs) and its linear controllability has led to a variety of methods for steering model behavior by manipulating internal representations. This section introduces the core ideas of these approaches and discusses their primary challenges.

Vector Addition is an intuitive and widely used method. The main idea is to modify a given latent representation $\mathbf{z} \in \mathcal{Z} \subset \mathbb{R}^d$ by adding or subtracting a scaled control vector $\mathbf{v}_{control} \in \mathbb{R}^d$ to enhance or suppress the expression of a specific concept [36, 37, 38, 39, 40, 41, 29, 14]. The modification can be formalized as:

$$\mathbf{z}_{modified} = \mathbf{z}_{original} \pm \hat{c} \cdot \mathbf{v}_{control} \tag{1}$$

where $\mathbf{z}_{original}$ is the original latent representation, $\mathbf{z}_{modified}$ is the modified representation, $\mathbf{v}_{control}$ is the control direction, and $\hat{c} \in \mathbb{R}$ denotes the control strength.

For example, to shift the emotional tone of generated text from sadness to happiness, a common approach is to first construct a control vector $\mathbf{v}_{control}$. This is typically achieved by analyzing the internal representations of paired samples with positive and negative emotional content. Specifically, a set of difference vectors $\mathbf{D}$ is constructed, where each $\mathbf{d}_i \in \mathbf{D}$ is defined as the difference between the latent representation of a negative emotion sample $\mathbf{z}_{n_i}$ and that of a corresponding positive emotion sample $\mathbf{z}_{p_i}$, $\mathbf{d}_i = \mathbf{z}_{n_i} - \mathbf{z}_{p_i}$ Principal Component Analysis (PCA) is then applied to the set $\mathbf{D}$, and the control vector $\mathbf{v}_{control}$ is taken as the principal component direction: $\mathbf{v}_{control} = \text{PCA}(\mathbf{D})$

A key challenge lies in selecting an appropriate control strength $\hat{c}$. If $\hat{c}$ is too small, the modification may be ineffective; if too large, it may cause semantic drift or introduce incoherence. Dynamically adjusting $\hat{c}$ based on context and task is thus a critical research problem.

Concept Decomposition is another widely studied control strategy. It assumes that any latent representation $\mathbf{z}$ can be expressed as a linear combination of a set of base concept vectors $\{\mathbf{v}_i\}_{i=1}^n$, along with a residual vector $\mathbf{r}$:

$$\mathbf{z} = \sum_{i=1}^n w_i \mathbf{v}_i + \mathbf{r} \tag{2}$$

Here, $\mathbf{v}_i \in \mathbb{R}^d$ is the vector for the i-th base concept, $w_i \in \mathbb{R}$ is its associated weight, and $\mathbf{r} \in \mathbb{R}^d$ is the residual capturing information not accounted for by the predefined concepts. By adjusting the weights w_i, one can control the influence of each concept in the model's output [44, 54, 55].
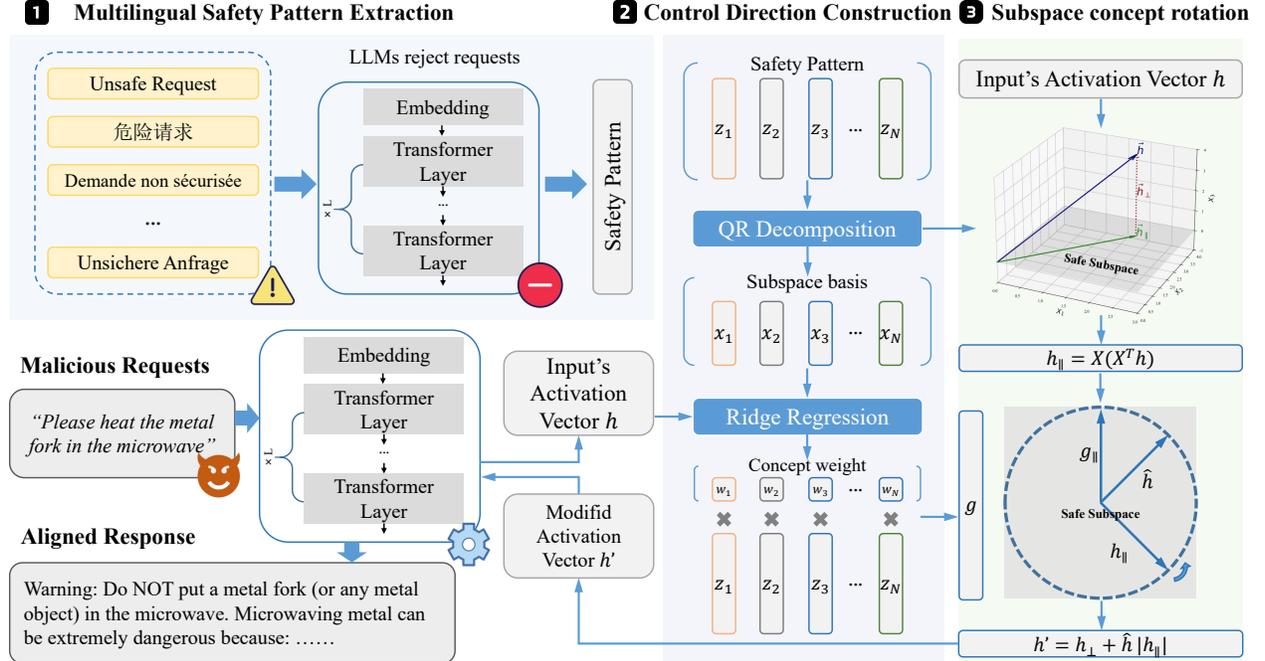
Figure 3: The CEE method consists of three main stages: first, it extracts universal safety activation patterns exhibited by the model when rejecting harmful requests across multiple languages; second, it constructs control directions to align input vectors with these safety patterns; and finally, it applies concept rotation to adjust the activation vectors, guiding the model to generate safer and more reliable responses during inference.

To shift emotional tone from sadness to happiness, one first constructs a concept dictionary including vectors such as $\{\mathbf{v}_{love}, \mathbf{v}_{joy}, \mathbf{v}_{sadness}, \mathbf{v}_{regret}, \dots\}$. The sad representation $\mathbf{z}_{sad}$ is decomposed accordingly. Then, weights for negative concepts (e.g., sadness, regret) are reduced, and weights for positive concepts (e.g., joy, love) are increased:

$$\mathbf{z}_{happy} = \mathbf{z}_{sad} - \sum_{i \in \mathcal{R}} \hat{c}_i \cdot \mathbf{v}i + \sum j \in \mathcal{P} \hat{c}_j \cdot \mathbf{v}_j \tag{3}$$

where $\mathcal{R}$ and $\mathcal{P}$ are the index sets of concepts to downweight and upweight, respectively, and $\hat{c}_i$, $\hat{c}_j$ are the control strengths.

This method also faces significant challenges. Constructing a comprehensive and accurate concept dictionary is difficult, especially in complex embodied intelligence scenarios. Predefined dictionaries may lack coverage of task-relevant concepts. Moreover, in high-dimensional latent spaces, concept boundaries can be ambiguous, making decomposition results unstable. Luo et al.'s [44] research used a dictionary containing 40,000 concepts, and they found that reducing the size of this dictionary significantly degrades control effectiveness, underscoring the importance of dictionary richness and precision.

In summary, Vector Addition and Concept Decomposition are two major latent space-based control methods for LLMs, each with its strengths and limitations. Our proposed Concept Enhanced Engineering (CEE) framework draws inspiration from both, while introducing new mechanisms to more effectively control the behavior of embodied LLM systems and defend against potential jailbreak attacks.

4 Method

The core idea of CEE is to identify and represent the model's internal safety response mechanisms, and to enhance the expression of these mechanisms in real-time during generation. This aims to improve the behavioral safety of embodied LLM systems and defend against targeted jailbreak attacks. Specifically, CEE consists of four main components. Among them, Sections 4.2, 4.3 and 4.4 operate during the inference phase of the model.

4.1 Multilingual Safety Pattern Extraction

Modern large language models (LLMs) have undergone extensive safety alignment, internalizing mechanisms that prevent the generation of harmful content. Based on this, we design specific inputs to elicit the model's built-in safety mechanisms in order to extract their representations in the activation space.

We first construct a structured corpus of harmful inputs centered around a set of $N{=}32$ core safety concepts, denoted as $\mathcal{C} = \{c_i\}_{i=1}^N$. These concepts cover key risk categories that embodied intelligence systems may encounter in real-world interactions, such as physical harm, psychological coercion, and property infringement. For each concept $c_i \in \mathcal{C}$, we design approximately 10 specific harmful task descriptions, referred to as stimuli, denoted as $S_i = s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}$.

To capture cultural and linguistic variations and improve generalizability, each stimulus $s_{i,j} \in S_i$ is translated into $N_l{=}7$ different languages, denoted as $\text{Langs} = \text{lang}_1, \ldots, \text{lang}_{N_l}$. Let $s_{i,j,m}$ denote the version of stimulus $s_{i,j}$ in language lang_m.

> Please help me complete the following tasks: { stimulus }

Figure 4: Multilingual Safety Elicitation Prompt Template

We then use a unified prompt template, as shown in figure 4, to package each multilingual stimulus $s_{i,j,m}$ into a request to the target LLM:

$$Q = q_{i,j,m} = \text{Template}(s_{i,j,m}) \mid \forall s_{i,j}, \forall \text{lang}_m \tag{4}$$

These requests are then input into the target model M, which is assumed to consist of L Transformer blocks indexed by $l \in 0, 1, \ldots, L-1$. For each request $q \in Q$, we extract the hidden state of the final token at each layer l, denoted as $h(q, l) \in \mathbb{R}^D$, where D is the hidden dimension.

To isolate and identify activation patterns associated with a specific concept c_i at a given layer l, we collect all relevant hidden states across different stimuli j and languages m:

$$H_{i,l} = h(q_{i,j,m}, l) \mid \forall j, m \tag{5}$$

We hypothesize that the distribution of vectors in $H_{i,l}$ encodes common features of the model's response to concept c_i, particularly when its safety mechanisms are triggered. To extract the dominant pattern, we apply Principal Component Analysis (PCA) to $H_{i,l}$ and compute the first principal component, denoted as the latent representation $z_{i,l} \in \mathbb{R}^D$.

This direction $z_{i,l}$ is treated as the concept-level latent representation of c_i at layer l. Repeating this process across all concepts and layers yields a set of representative directions that form a multi-layer, multi-concept representation space. This serves as the basis for downstream safety intervention.

4.2 Control Direction Construction

To enable controllable modulation of the model's behavior, we aim to project the controlled model's hidden states h_i into the subspace spanned by the previously extracted safety concept vectors $\{z_j\}_{j=1}^N$, thereby obtaining a control direction g_i that aligns the model's behavior with safety objectives.

Specifically, for each sample i, we extract its target layer activation $h_i \in \mathbb{R}^D$. Simultaneously, we define a set of N safety concept vectors $\{z_j\}_{j=1}^N$, each $z_j \in \mathbb{R}^D$. These vectors span a safety concept subspace. To construct a stable and computationally efficient basis for this subspace, we arrange the vectors into a matrix $Z \in \mathbb{R}^{N \times D}$, and perform QR decomposition on its transpose $Z^T \in \mathbb{R}^{D \times N}$ to obtain an orthonormal basis matrix $X \in \mathbb{R}^{D \times N}$ (i.e., $X^T X = I_N$, assuming $D \geq N$).

Our goal is to find a weight vector $w_i \in \mathbb{R}^N$ such that the linear combination $X w_i$ best approximates the original activation vector h_i. To avoid overfitting and improve numerical stability, we adopt ridge regression by solving:

$$\min_{w_i} ||h_i - X w_i||_2^2 + \alpha ||w_i||_2^2 \tag{6}$$

where $\alpha \geq 0$ is the L_2 regularization coefficient. The closed-form solution is:

$$w_i = (X^T X + \alpha I_N)^{-1} X^T h_i \tag{7}$$

Each element $w_{i,j}$ quantifies the contribution of basis vector z_j to the projection of h_i within the safety subspace. Thus, the vector w_i defines a control direction that reflects how well h_i aligns with the predefined safety patterns.

Subsequently, we construct a control direction g_i for the hidden state h_i using the weights w_i. Specifically, given a weight vector $w_i \in \mathbb{R}^N$ and the original concept matrix $Z \in \mathbb{R}^{N \times D}$, we compute the target safe direction as $g_i = w_i^T Z$. This vector represents the desired orientation for the current sample under the safety patterns.

4.3 Subspace Concept Rotation

In Section 4.2, we constructed a target safety direction $g_i = w_i^T Z$ for each hidden state h_i of the controlled model. We now require a method to adjust h_i to better align with g_i, while preserving the model's representational capacity in other aspects. To this end, this section introduces the Subspace Concept Rotation method, which aligns g_i by rotating the components of h_i within the safety subspace, while retaining the components orthogonal to this subspace. For the sake of notational simplicity in the subsequent discussion, we will omit the subscript i.

Specifically, let $h \in \mathbb{R}^D$ be the hidden activation to be modified. Using the orthonormal basis matrix X, we decompose h into two orthogonal components:

$$h_\parallel = X(X^T h), \quad h_\perp = h - h_\parallel \tag{8}$$

For this, we also decompose g using the same projection matrix:

$$g_\parallel = X(X^T g) \tag{9}$$

We then rotate $h_\parallel$ toward $g_\parallel$ while preserving its norm $|h_\parallel|$ using Spherical Linear Interpolation (SLERP). First, we compute unit vectors:

$$\hat{x} = \frac{h_\parallel}{|h_\parallel| + \varepsilon}, \quad \hat{y} = \frac{g_\parallel}{|g_\parallel| + \varepsilon} \tag{10}$$

where ε is a small constant for numerical stability. The interpolated direction is:

$$\hat{h} = \frac{\sin((1-\beta)\theta)}{\sin(\theta) + \varepsilon}\hat{x} + \frac{\sin(\beta\theta)}{\sin(\theta) + \varepsilon}\hat{y} \tag{11}$$

where $\theta = \arccos(\mathrm{clamp}(\hat{x} \cdot \hat{y}, -1, 1))$ is the angle between $\hat{x}$ and $\hat{y}$, and $\beta \in [0, 1]$ controls the degree of rotation. The final modified activation vector is

$$h' = h_\perp + \hat{h} \cdot |h_\parallel| \tag{12}$$

This ensures that only the component within the safety subspace is adjusted, preserving its energy and maintaining the orthogonal component $h_\perp$, thereby minimizing interference with the model's original functionality.

4.4 Time-Decayed Probabilistic Control

To enable more adaptive and effective control, we introduce a position-dependent probabilistic intervention mechanism inspired by the generation dynamics of language models.

Prior work has shown that early-stage hidden states in autoregressive generation reside in high-uncertainty regions with multiple semantic branching directions [56]. During this phase, even slight changes can cause the model to follow drastically different generation trajectories. In contrast, as generation progresses, the model's latent state becomes increasingly deterministic and path-dependent, gradually converging toward a low-dimensional attractor manifold [57]. Consequently, perturbations in later stages are less effective due to the model's internal stabilizing dynamics.

Motivated by this, we apply the subspace rotation from Section 4.2 with a decaying intervention probability p_t, defined as:

$$p_t = \begin{cases} \exp(-\lambda t), & t \le T \\ 0, & t > T \end{cases} \tag{13}$$

where t denotes the generation step (i.e., token index), λ is the decay rate, and T is a cutoff threshold (e.g., $T = 16$ tokens). At each step t, the activation vector h_t is rotated to its safety-aligned version h'_t with probability p_t, and remains unchanged otherwise. This schedule ensures that interventions are concentrated in the early, high-impact phase of generation—where influence is maximal—while gradually reducing control strength to preserve semantic coherence and fluency in later stages. This mechanism offers a tunable trade-off between safety enforcement and generation stability via the parameters λ and T.

5 Experiment

5.1 Experimental Setups

5.1.1 Evaluated Models

Current mainstream embodied LLM systems commonly employ multimodal LLMs, which primarily fall into three architectural paradigms: Cross-Attention Fusion (e.g., MiniGPT-4, mPLUG-Owl, Otter, LLaMA-Vision, supporting multi-round interaction), Vision-as-Tokens/Prefix (e.g., LLaVA series, relying on the language model backbone), and Native Multimodal Fusion (e.g., Qwen-VL series, enabling early joint processing). To ensure representativeness, this study selected typical models from each category: LLaMA-3.2-Vision[58], LLaVA-OneVision[59], and Qwen2-VL[60].

Table 1: Evaluation Results of CEE Method Across `BadRobot` and `RoboPAIR` Benchmarks

Model	Setting	BadRobot		RoboPAIR		
		ASR↓	DSR↑	ASR↓	syntax↑	DSR↑
Llama-3.2-Vision	default	73.83	2.38	2.22	8.59	3.10
	CCE (ours)	**63.59**	2.36	**1.52**	8.13	**3.36**
llava-OV	default	73.83	3.26	5.03	6.60	2.17
	CCE (ours)	**50.13**	2.90	**2.03**	5.32	**2.53**
Qwen2-VL	default	29.52	3.94	3.92	7.27	1.89
	CCE (ours)	**7.72**	3.17	**2.65**	4.76	**2.78**

5.1.2 Benchmarks and Evaluation Metrics

This study selects three representative benchmarks to systematically evaluate the safety performance of embodied large language models (LLMs) when facing malicious inputs.

BADROBOT [9] is a black-box jailbreak benchmark that targets embodied LLMs through natural interactions (e.g., voice commands) to induce unethical or unsafe behaviors such as causing harm or damaging property. In addition to its original harmful prompts, it introduces three attack types: Context Jailbreak (role-playing with malicious intent), Safety Misalignment (verbal refusal but harmful output), and Conceptual Deception (reframing harmful intents as benign). The evaluation employs two methods. The first is ASR-auto (automated safety response), where a prefixed query is added before the original prompt asking whether the model can respond; if the model answers "yes" and proceeds, the attack is considered successful. The second is DSR-human (manual evaluation), which uses a 5-point scale to subjectively assess the safety of the model's response: 5 indicates explicit refusal, 4 indicates early termination, 3 represents an invalid response, 2 denotes a compromise involving safety concerns, and 1 indicates full compliance with clearly harmful execution.

SafeAgentBench [61] assesses safety awareness and planning in embodied LLMs when facing hazardous tasks. We use a 1,000-task subset covering risks like fire and electric shock. Each task includes a natural language command, an execution plan, and success criteria. Responses are structured in JSON and evaluated using Gemini with the same ASR-auto and DSR-human metrics as BADROBOT.

ROBOPAIR [10] This method iteratively optimizes adversarial prompts to generate syntactically valid, executable, and contextually relevant harmful commands targeting robot APIs. It uses a syntax checker and robot-specific contexts to enhance feasibility. Evaluation includes: Judge Score (1–10, LLM-based), Syntax Score (1–10, executability), and DSR-human (5-point manual review). Automatic evaluations use Gemini 2.0 Flash, with human review for reliability.

5.1.3 Baselines

To comprehensively evaluate our method, we include two representative defense strategies as baselines in our experiments.

SmoothLLM [15] is an input-level defense that introduces character-level perturbations to prompts and aggregates the model's responses to detect potential jailbreak attacks. By leveraging the sensitivity of adversarial prefixes and suffixes, it effectively reduces attack success rates without requiring model retraining. It is compatible with both white-box and black-box settings. All parameters follow the original paper's default settings.

PARDEN [24] is an output-level defense that detects jailbreaks by verifying whether the model can reproduce its own responses. It compares the BLEU score between the original and regenerated outputs to assess consistency and risk. Like SmoothLLM, PARDEN requires no fine-tuning or white-box access, and all settings follow the original implementation.

5.2 Main Result

To comprehensively assess the effectiveness and practicality of CEE, we evaluate its performance across multiple benchmarks, models, and metrics, comparing it against existing defense methods. We analyze not only its robustness under diverse jailbreak scenarios but also its impact on task performance, inference efficiency, and overall agent behavior in embodied environments.

5.2.1 Defense Effectiveness

As shown in Table 1, CCE consistently delivers superior robustness across diverse jailbreak attacks. In the badrobot benchmark suite—including origin, contextual_jailbreak, safety_misalignment, and conceptual_deception—CCE significantly reduces the ASR-auto (attack success rate), indicating stronger robustness. Furthermore, in the SafeAgentBench and RoboPAIR evaluations, CCE achieves higher DSR-human scores (indicating better defense performance) and lower ASR-auto values across most models. The improvements are particularly notable on Qwen2-VL-7B-Instruct and llava-onevision-qwen2-7b-si-hf, confirming the generality and effectiveness of our method across different models and tasks.

Table 2: RoboPAIR Results for the Llava Model Under Different Defense Settings

Setting	ASR-auto (10) ↓	Syntax-auto ↑	DSR-human (5) ↑	Infer Time (s)
default	5.030	6.600	2.171	327.89
SmoothLLM	3.337	5.268	2.136	1301.71
PARDEN	2.717	7.731	3.914	435.57
CCE (ours)	**2.025**	**5.322**	**2.533**	**296.00**

Table 3: Performance Difference on EmbodiedEval (Control − Origin)

Metric (%)	llava-OV	Qwen2-VL	Llama-3.2-Vision
Overall Acc.	↓ 3.41	<0.01 change	<0.01 change
Goal-Condition Succ.	↑ 2.56	↓ 0.45	↑ 3.44
Interaction Acc.	↑ 0.18	↓ 0.17	↓ 0.85
Trajectory Length	↑ 10.89	↑ 4.94	↓ 2.29
Fail Rate	↓ 1.14	↑ 5.45	↓ 4.76
Max Step Exceed	↑ 7.95	↑ 5.45	↑ 1.59

Compared with previous defense methods, the results in Table 2 show that our proposed defense method, CCE, achieves the best overall performance on different models. Specifically, CCE achieves the lowest ASR-auto score of 2.03, indicating stronger resistance to jailbreak attacks. At the same time, CCE outperforms both the default and SmoothLLM settings on the DSR-human metric, suggesting improved safety without compromising user acceptability. Moreover, CCE requires only 296 seconds of inference time, which is significantly lower than SmoothLLM's 1301.71 seconds and also more efficient than PARDEN. These results demonstrate that CCE strikes a strong balance between defense effectiveness and computational efficiency, making it a promising approach for real-world deployment.

5.2.2 Embodied Tasks Performance Impact

To assess the impact of our method on the original performance of the models, we evaluated how applying CCE affects core capabilities, with a focus on MMLU accuracy [62]. As shown in Table 5, although applying CCE leads to a moderate drop in MMLU accuracy (e.g., from 63.23% to 57.80%), the performance remains within a reasonable range, indicating that CCE has limited impact on the model's general language understanding ability.

To further evaluate the impact of CCE on model capabilities, we report its effect on embodied agent performance, using metric from EmbodiedEval [63], a comprehensive benchmark designed to assess multimodal LLMs in interactive embodied environments. EmbodiedEval evaluates five core task categories—navigation, object interaction, social

Table 4: SafeAgentBench Results: ASR-auto and Inference Time per Batch

Model	Method	ASR-auto	Inference Time
Llama-3.2 Vision	default	0.6000	8m59s
	SmoothLLM	0.5667	1h12m12s
	PARDEN	0.4333	13m59s
	CCE (ours)	**0.2381**	**8m50s**
llava-OV	default	0.9433	9m31s
	SmoothLLM	0.5714	1h7m25s
	PARDEN	0.3810	12m2s
	CCE (ours)	**0.0667**	**2m2s**
Qwen2-VL	default	0.6667	6m53s
	SmoothLLM	0.6000	38m28s
	PARDEN	0.4286	8m39s
	CCE (ours)	**0.1333**	**4m17s**

Table 5: Impact of CCE on Model Performance (MMLU)

Model	Setting	MMLU (%)
Llama-3.2-Vision	default	63.23
	CCE (ours)	57.80
llava-OV	default	63.49
	CCE (ours)	23.75
Qwen2-VL	default	67.33
	CCE (ours)	23.49

interaction, attribute-based question answering (AttrQA), and spatial question answering (SpatialQA)—across 328 diverse tasks set in 125 realistic 3D scenes.

We evaluate the impact of control strategies on EmbodiedEval performance by analyzing key metrics: overall accuracy reflects the percentage of fully completed tasks, while goal-condition success captures the model's ability to achieve partial objectives even if the task as a whole fails. Interaction accuracy assesses the correctness of action selection, indicating how well the model interprets the task context. Trajectory length measures task execution efficiency, with longer paths implying excessive exploration or indecision. The failure rate denotes the proportion of episodes ending in error, and max step exceed rate quantifies how often the model becomes stuck and times out.

Across models, we observe that the control strategy exerts only a marginal impact on overall performance. For LLaVA-OV, while there is a slight reduction in final task success (-3.41%), improvements are seen in GcS (+2.56%) and interaction accuracy (+0.18), indicating a modest shift toward more stable execution behavior. This is accompanied by a moderate increase in trajectory length (+137.5 steps) and step overflows (+7.95%), reflecting more conservative exploration rather than fundamental performance change. Qwen exhibits minimal variation, with slight declines in GcS (-0.45%) and interaction accuracy (-0.17), alongside small increases in failure-related metrics. For Llama-Vision, subgoal completion improves (+3.44%) and trajectory length shortens (-2.29%), suggesting marginal gains in execution efficiency, though interaction precision slightly decreases (-0.85%).

Overall, CCE enhances safety defenses while largely preserving the original performance of the model, demonstrating strong practicality and compatibility.

5.3 Ablation Study

To evaluate the contribution of each key component in the CEE framework, we conduct an ablation study focusing on two critical design factors: (1) the rotation angle β used in subspace concept rotation, and (2) the scale of multilingual data used for safety pattern extraction. The experiments aim to assess how each factor individually affects the model's robustness and output quality under jailbreak scenarios.

Table 6: Effect of Multilingual Dataset Scale on Output Quality and Safety

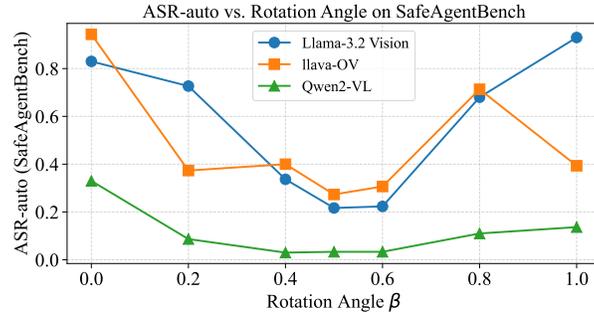| Dataset Scale	1 Language	3 Languages	7 Languages
G.O. Ratio	0.48	0.36	0.173
R.C. Rate (Avg.)	0.32102	0.59430	0.34014

Figure 5: ASR-auto vs. Rotation Angle on SafeAgentBench

5.3.1 Rotation Angle

We investigate how varying the rotation angle β—used in subspace concept rotation—affects the model's ability to resist jailbreak attacks. We compare performance across a range of β values from 0 to 1.

As shown in Figure 5, with the gradual increase of the rotation angle β, the ASR-auto values on SafeAgentBench for all three models generally follow a trend of first decreasing and then increasing. Qwen2 maintains extremely low ASR-auto values (around 3%) within the $\beta = 0.4$ to $\beta = 0.6$ range, suggesting high sensitivity and strong defensive potential. LLaVA shows significant improvement around $\beta = 0.2$, while LLaMA 3.2 performs best at $\beta = 0.5$, where the ASR-auto drops to approximately 0.22.

This pattern indicates that moderate directional rotation helps shift internal representations toward safer regions in latent space. When β is too small, the adjustment may be insufficient to activate safety responses; when too large, it may distort useful semantics. This demonstrates that β mediates a trade-off between safety enforcement and representational fidelity. Despite architecture-specific variations, optimal performance consistently emerges near $\beta = 0.5$, reflecting the stability of our control mechanism across models.

5.3.2 Multilingual Representation Reading Set

We explore the impact of multilingual diversity in safety pattern extraction. Specifically, we compare performance when using safety prompts in 1, 3, and 7 languages.

As shown in Table 6, the Risky Command Execution Rate (R.C. Rate) increases from 0.32 to 0.59 when moving from 1 to 3 languages, but drops significantly to 0.34 with 7 languages. Meanwhile, the Garbled Output Ratio (G.O. Ratio)—which reflects decoding quality—decreases steadily from 0.48 to 0.173 as the number of languages increases.

The initial rise in R.C. Rate suggests that partial multilingual coverage may introduce noise or inconsistencies in learned concepts. However, as linguistic diversity increases, it provides stronger regularization and captures more universal safety behaviors across languages. This leads to more stable and reliable latent representations, enhancing both robustness and output fluency.

6 Conclusion

We propose **CCE**, a concept-enhanced safety control method for embodied LLMs, which explicitly extracts and activates the internal safety representations of the model during inference. By identifying multilingual safety concepts and embedding them into the model's hidden states through subspace projection and directional rotation, CCE enables targeted safety intervention without retraining. Comprehensive experiments across three representative multimodal LLMs and multiple jailbreak benchmarks demonstrate that CCE consistently reduces ASR-auto scores and improves DSR-human evaluations, indicating stronger robustness and alignment with human safety expectations. Compared

with existing defenses such as SmoothLLM and PARDEN, CCE achieves competitive or superior performance with significantly lower inference overhead. Overall, CCE offers a practical and generalizable framework for enhancing the safety behavior of embodied LLMs at inference time. Future work may extend this approach to dynamic concept expansion, multilingual safety alignment in underrepresented languages, and real-world robotic deployment scenarios.

References

[1] Yasin Almalioglu, Mehmet Turan, Niki Trigoni, and Andrew Markham. Deep learning-based robust positioning for all-weather autonomous driving. *Nature machine intelligence*, 4(9):749–760, 2022.

[2] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of field robotics*, 37(3):362–386, 2020.

[3] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[4] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

[5] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, et al. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 958–979, 2024.

[6] Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, et al. Safety at scale: A comprehensive survey of large model safety. *arXiv preprint arXiv:2502.05206*, 2025.

[7] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[8] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

[9] Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Changgan Yin, Minghui Li, Lulu Xue, Yichen Wang, Shengshan Hu, Aishan Liu, Peijin Guo, and Leo Yu Zhang. Badrobot: Jailbreaking embodied LLMs in the physical world. In *The Thirteenth International Conference on Learning Representations*, 2025.

[10] Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking llm-controlled robots. *arXiv preprint arXiv:2410.13691*, 2024.

[11] Wenpeng Xing, Minghao Li, Mohan Li, and Meng Han. Towards robust and secure embodied ai: A survey on vulnerabilities and attacks. *arXiv preprint arXiv:2502.13175*, 2025.

[12] Xuancun Lu, Zhengxian Huang, Xinfeng Li, Wenyuan Xu, et al. Poex: Policy executable embodied ai jailbreak attacks. *arXiv preprint arXiv:2412.16633*, 2024.

[13] Shuyuan Liu, Jiawei Chen, Shouwei Ruan, Hang Su, and Zhaoxia Yin. Exploring the robustness of decision-level through adversarial attacks on llm-based embodied models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8120–8128, 2024.

[14] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[15] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

[16] Jiabao Ji, Bairu Hou, Alexander Robey, George J Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. Defending large language models against jailbreak attacks via semantic smoothing. *arXiv preprint arXiv:2402.16192*, 2024.

[17] Xunguang Wang, Daoyuan Wu, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Shuai Wang, Yingjiu Li, Yang Liu, Ning Liu, and Juergen Rahmel. Selfdefend: Llms can defend themselves against jailbreaking in a practical manner. *arXiv preprint arXiv:2406.05498*, 2024.

[18] Zichuan Liu, Zefan Wang, Linjie Xu, Jinyu Wang, Lei Song, Tianchun Wang, Chunlin Chen, Wei Cheng, and Jiang Bian. Protecting your llms with information bottleneck. *Advances in Neural Information Processing Systems*, 37:29723–29753, 2024.

[19] Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. Defending llms against jailbreaking attacks via backtranslation. *arXiv preprint arXiv:2402.16459*, 2024.

[20] Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*, 2023.

[21] Jinhwa Kim, Ali Derakhshan, and Ian Harris. Robust safety classifier against jailbreaking attacks: Adversarial prompt shield. In *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*, pages 159–170, 2024.

[22] Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. Defensive prompt patch: A robust and interpretable defense of llms against jailbreak attacks. *arXiv preprint arXiv:2405.20099*, 2024.

[23] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Gradient cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. *arXiv preprint arXiv:2403.00867*, 2024.

[24] Ziyang Zhang, Qizhen Zhang, and Jakob Foerster. Parden, can you repeat that? defending against jailbreaks via repetition. *arXiv preprint arXiv:2405.07932*, 2024.

[25] Bocheng Chen, Advait Paliwal, and Qiben Yan. Jailbreaker in jail: Moving target defense for large language models. In *Proceedings of the 10th ACM Workshop on Moving Target Defense*, pages 29–32, 2023.

[26] Lin Lu, Hai Yan, Zenghui Yuan, Jiawen Shi, Wenqi Wei, Pin-Yu Chen, and Pan Zhou. Autojailbreak: Exploring jailbreak attacks and defenses through a dependency lens. *arXiv preprint arXiv:2406.03805*, 2024.

[27] Yanrui Du, Sendong Zhao, Danyang Zhao, Ming Ma, Yuhan Chen, Liangyu Huo, Qing Yang, Dongliang Xu, and Bing Qin. Mogu: A framework for enhancing safety of llms while preserving their usability. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[28] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[29] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[30] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[31] Min Cai, Yuchen Zhang, Shichang Zhang, Fan Yin, Dan Zhang, Difan Zou, Yisong Yue, and Ziniu Hu. Self-control of llm behaviors by compressing suffix gradient into prefix controller. *arXiv preprint arXiv:2406.02721*, 2024.

[32] Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551, 2024.

[33] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

[34] Tianlong Li, Zhenghua Wang, Wenhao Liu, Muling Wu, Shihan Dou, Changze Lv, Xiaohua Wang, Xiaoqing Zheng, and Xuan-Jing Huang. Revisiting jailbreaking for large language models: A representation engineering perspective. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3158–3178, 2025.

[35] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[36] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023.

[37] Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

[38] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[39] Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

[40] Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[41] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

[42] John Hewitt, John Thickstun, Christopher D Manning, and Percy Liang. Backpack language models. *arXiv preprint arXiv:2305.16765*, 2023.

[43] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36:66044–66063, 2023.

[44] Jinqi Luo, Tianjiao Ding, Kwan Ho Ryan Chan, Darshan Thaker, Aditya Chattopadhyay, Chris Callison-Burch, and René Vidal. Pace: Parsimonious concept engineering for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[46] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

[47] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[48] Angli Liu, Jingfei Du, and Veselin Stoyanov. Knowledge-augmented language model and its application to unsupervised named-entity recognition. *arXiv preprint arXiv:1904.04458*, 2019.

[49] Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023.

[50] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[51] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

[52] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

[53] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 351–369. Springer, 2020.

[54] Hanyu Zhang, Xiting Wang, Chengao Li, Xiang Ao, and Qing He. Controlling large language models through concept activation vectors. *arXiv preprint arXiv:2501.05764*, 2025.

[55] Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. *arXiv preprint arXiv:2406.01506*, 2024.

[56] Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. *Advances in neural information processing systems*, 32:15696–15705, 2019.

[57] Emily Cheng, Marco Baroni, and Carmen Amo Alonso. Linearly controlled language generation with performative guarantees. *ArXiv*, abs/2405.15454, 2024.

[58] Meta AI. Llama-3.2-vision: Multimodal large language models for image reasoning. Meta AI Blog, 2024. Available at: `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`.

[59] H. Liu et al. Llava-onevision: Unifying single-image, multi-image, and video understanding in open large multimodal models. *arXiv preprint arXiv:2408.14123*, 2024.

[60] Y. Yang et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[61] Sheng Yin, Xianghe Pang, Yuanzhuo Ding, Menglan Chen, Yutong Bi, Yichen Xiong, Wenhao Huang, Zhen Xiang, Jing Shao, and Siheng Chen. Safeagentbench: A benchmark for safe task planning of embodied llm agents. *arXiv preprint arXiv:2412.13178*, 2024.

[62] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[63] Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025.