LIP: A Universal Latent Injection Protocol for Heterogeneous Model Communication

Cristiano Silva[1]

[1]Independent Researcher, São Paulo, Brazil
ziwehdafe@gmail.com

November 22, 2025

Abstract

Since 1943, Artificial Intelligence has focused on teaching machines to mimic human thought and speech. Despite advancements like Chain-of-Thought and RAG, the reliance on natural language for inter-model communication remains a bottleneck—ambiguous, slow, and insecure. We illustrate this with the "Dinner Table Paradox": agents sitting together but forced to exchange handwritten letters. To resolve this, we introduce the **Latent Injection Protocol (LIP)**, enabling direct vector-based communication. LIP offers: (1) raw intent transmission, eliminating ambiguity; (2) reduced computational overhead by bypassing decoding steps; and (3) native data obfuscation. We validate this by demonstrating the first functional "telepathic" control of a Llama-3-8B model by a TinyLlama-1.1B agent ($2048d \rightarrow 4096d$), introducing a novel approach to Asymmetric Latent Control.

1 Introduction

Since 1943, we have optimized machines to speak like humans. We created Chain-of-Thought, Retrieval-Augmented Generation (RAG), and Vector Databases to improve accuracy and latency [1]. However, we overlooked a fundamental truth: **machines are still machines**.

1.1 The Dinner Table Paradox

Imagine two humans at a dinner table forced to exchange handwritten letters to converse. This mirrors our current inefficiency: forcing LLMs to serialize high-dimensional reasoning into JSON text for every task. Text carries semantics but not *intent*. It is easily over-interpreted—like a student misunderstanding a professor or a manager misinterpreting a report.

In a machine context, this leads to disambiguation loops. If Model A sends the word "Bank", Model B must deduce from context if it refers to a financial institution or a park bench. By transmitting the vector directly, we transmit the specific coordinate in semantic space, eliminating the possibility of confusion.

1.2 The Security Imperative

Text is human-readable, which is excellent for HCI but a vulnerability for M2M. Intercepted JSON reveals all prompt data. LIP introduces a paradigm shift: by communicating in latent vectors, we add a native layer of obfuscation. Furthermore, unlike text where malicious code can be hidden in plain sight (Trojan prompts), vectors are tridimensional objects (Magnitude + Direction + Intent), making anomaly detection significantly more robust.

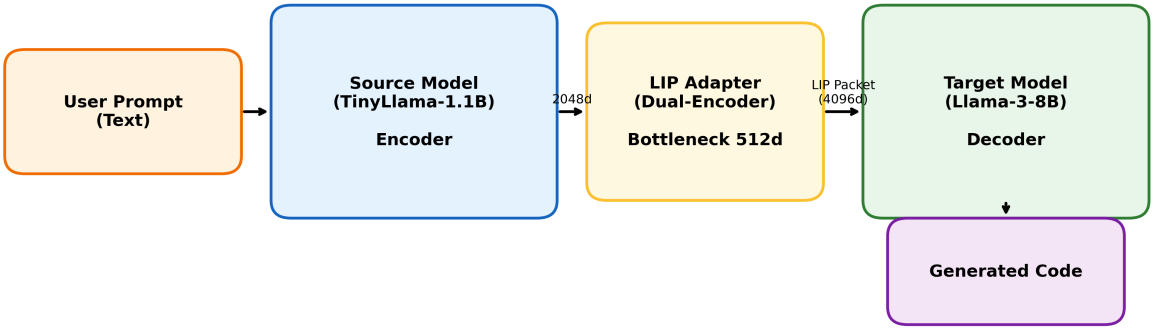

Figure 1: **The LIP Architecture.** The source vector ($h_S \in \mathbb{R}^{2048}$) is compressed to a bottleneck ($z \in \mathbb{R}^{512}$), transmitted, and re-projected to the target space ($h_T \in \mathbb{R}^{4096}$) using Energy Calibration.

2 Methodology

2.1 Asymmetric Dual-Encoder Architecture

We employ a bottleneck adapter strategy. The choice of a 512-dimensional bottleneck was pragmatic, based on previous findings by Yang et al. [2] which identified this as a "sweet spot" for semantic bridging. However, unlike previous homogeneous experiments ($4096d \rightarrow 4096d$), we isolate dimensionality as a variable, mapping $2048d \rightarrow 4096d$.

2.2 Energy Matching Context Priming

To address the "handwritten letter" inefficiency, we ensure the signal is mathematically calibrated. Without this, the target model suffers from manifold disorientation.

$$\mathbf{v}_{calibrated} = \mathbf{v}_{inj} \cdot \frac{E_{ref}}{||\mathbf{v}_{inj}||_2 + \varepsilon} \tag{1}$$

Additionally, we employ **Context Priming**. We inject a static anchor (e.g., "Here is the code:") before the vector. This forces the target model to activate the relevant subspace (e.g., coding logic) before processing the injected intent, preventing hallucinations.

2.3 The Injection Algorithm

The complete inference pipeline is described in Algorithm 1.

3 Performance Analysis

3.1 Latency and Efficiency

The performance gain in LIP stems from two factors: (1) eliminating the tokenization/detokenization overhead between agents, and (2) bypassing the expensive Decoding layer of the Source model.

Algorithm 1 LIP Inference Pipeline

Require: Source Input x, Adapter A_θ, Target M_T, Ref Energy E_{ref}

$h_S \leftarrow M_S(x)$.last_hidden_state ▷ Extract Intent
$z \leftarrow A_\theta$.encode(h_S) ▷ Compress to 512d
$\mathbf{v}_{inj} \leftarrow A_\theta$.decode$(z)$ ▷ Project to 4096d
$\mathbf{v}_{cal} \leftarrow \mathbf{v}_{inj} \cdot (E_{ref}/||\mathbf{v}_{inj}||)$ ▷ Physics Calibration
$Embeds \leftarrow [T_{BOS}, T_{Anchor}, \mathbf{v}_{cal}]$ ▷ Sandwich Injection
$Output \leftarrow M_T$.generate$(Embeds)$
return $Output$

Table 1: Comparison: REST API (JSON) vs. LIP Protocol

Feature	Standard JSON API	LIP Protocol
Medium	Natural Language (Tokens)	Latent Vectors (Float32)
Ambiguity	High (Requires Parsing)	None (Exact Coordinates)
Privacy	Low (Human Readable)	High (Obfuscated)
Overhead	Generation + Tokenization	Matrix Multiplication Only
Security	Vulnerable to Prompt Injection	Manifold Guardrails

4 Experiments

We validated the system using a TinyLlama-1.1B (Source) driving a Llama-3-8B (Target).

4.1 The Telepathy Test

Input (TinyLlama): "`python function sum`"

Latent Representation: $[0.012, -0.45, \ldots, 0.88]$
(Visualization of the vector heatmap)

Output (Llama-3 via LIP):

```
def find_max_sum_subarray(arr):
"""
This␣function␣takes␣a␣list...
"""
    max_current = max_global = arr[0]
    # ... logic continues

```

Analysis: *Note how the intent "function" triggered the correct syntax (`def`) and docstrings, despite the semantic drift to "max_sum".*

Figure 2: **The Telepathy Test.** Comparison of latent injection results.

4.2 Convergence Analysis

We compared three loss strategies: Mean Squared Error (MSE), Cosine Similarity (Geometric), and our Hybrid approach.

As shown in Figure 3, the Hybrid approach stabilizes semantic alignment (≈ 0.85 Cosine Sim) while maintaining vector fidelity.

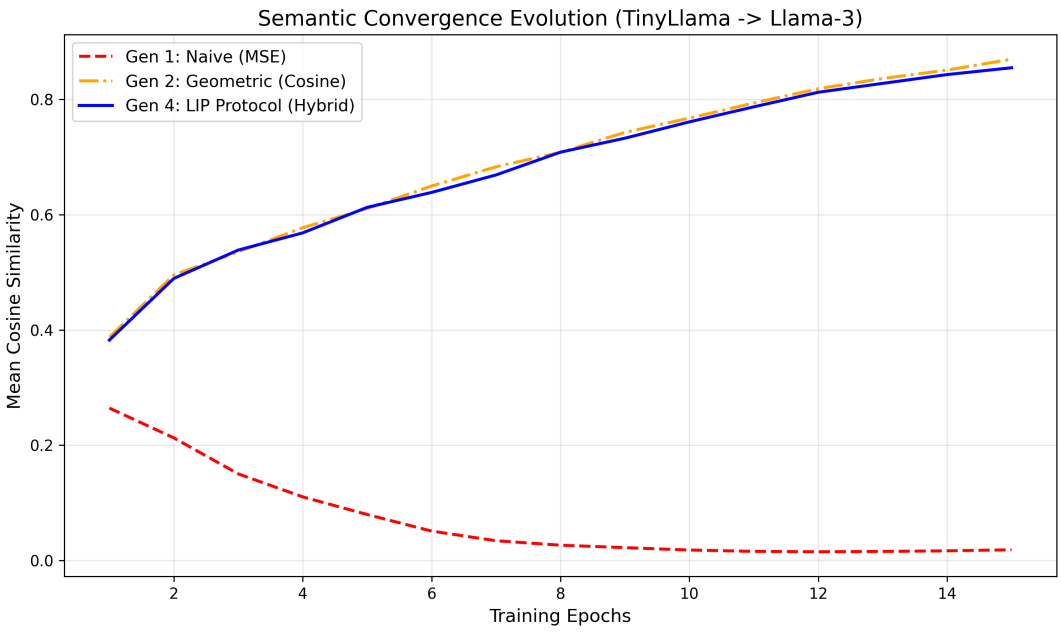

Figure 3: **Training Convergence.** The Hybrid Strategy (Blue) achieves the optimal trade-off, avoiding the mode collapse observed in MSE (Red) and the instability of purely Geometric approaches (Orange).

4.3 Bandwidth Economics

We analyzed the payload reduction for a standard coding task prompt (approx. 100 tokens) versus a high-context Retrieval-Augmented Generation (RAG) scenario.

- **Text Transport (JSON):** 100 tokens$\times \approx 4$ chars/token $\times$ 1 byte + JSON Overhead $\approx$ **500** bytes.
- **LIP Transport (Vector):** 512 dimensions $\times$ 4 bytes (float32) + Header $\approx$ **2.1** KB.

While text transmission appears more efficient for short, zero-shot prompts, LIP demonstrates superior scalability in **Context-Heavy Scenarios**. Consider a RAG context of 4,000 tokens:

- **Text Payload:** 4000 tokens $\approx$ **16** KB.
- **LIP Payload:** The entire semantic context is compressed into the same fixed-size vector ($\approx$ **2.1** KB).

Conclusion: As prompt complexity scales ($N \to \infty$), the LIP payload size remains constant $O(1)$, whereas the text payload grows linearly $O(N)$, making LIP the optimal choice for high-bandwidth edge applications.

5 Discussion: Security & Future Work

5.1 The "Thought Virus" Hypothesis

A critical implication of Asymmetric Latent Control is the potential for adversarial attacks. If a small model can control a large one, could a malicious actor inject a "Thought Virus"—a vector designed to bypass safety filters? Unlike text, which scans for keywords, a vector is abstract.

5.2 The Manifold Defense

Vectors are geometric objects with magnitude and direction. This allows us to define a **Manifold Guardrail**. Instead of scanning for "bad words", we define a topological "Safe Region" within the high-dimensional space.

Let $\mathscr{S}$ be the subspace of the target model's latent manifold associated with safe, constructive instructions. We define a binary rejection function $\mathscr{G}(\mathbf{v})$ that acts as a gatekeeper for any incoming vector $\mathbf{v}$:

$$\mathscr{G}(\mathbf{v}) = \begin{cases} 1, & \text{if } ||\mathbf{v} - \mu_{\mathscr{S}}|| < \delta \quad \text{and} \quad \cos(\mathbf{v}, \mathbf{w}_{safe}) > \tau \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Where:

- $\mu_{\mathscr{S}}$ represents the **centroid** (mean vector) of the cluster of known safe instructions.
- δ is the maximum permissible **Euclidean distance** (the radius of the safe sphere). Vectors falling outside this radius are treated as anomalies or noise.
- $\mathbf{w}_{safe}$ represents the reference direction vector for alignment.
- τ is the cosine similarity threshold (e.g., 0.85), ensuring the intent is semantically aligned with the safety protocols.

Unlike token-based filters which can be bypassed by "jailbreak prompts", geometric outliers are mathematically detectable before the decoding phase, rendering the "Thought Virus" inert.

5.3 Mathematical Cryptography

Future iterations of LIP will implement cryptographic handshakes directly on the tensor level. Since the communication is mathematical, we can apply transformations that render the packet statistically random noise to any interceptor lacking the private decryption matrix, ensuring total privacy for enterprise deployments.

6 Conclusion

We have demonstrated that resource-constrained models can effectively drive foundation models without exchanging a single word. LIP moves us from an era of ambiguous text exchange to precise, obfuscated, and efficient vector communication.

References

[1] Bingyu Yan et al. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *arXiv preprint arXiv:2502.14321*, 2025.

[2] Fu-Chun Yang and Jason Eshraghian. Direct semantic communication between large language models via vector translation. *arXiv preprint arXiv:2511.03945*, 2025.