

Latent Communication Interface (Lci): A Framework For Direct Representation-Level Communication Between Specialized Transformer Models

Ch V S Aditya¹, D Kishore²

^{1,2}Department of Computer Science & Engineering Koneru Lakshmaiah Education Foundation, Vaddeswaram 522502, India. Email: 2401050040cse@gmail.com

Abstract

Modern AI systems depend on ensembles of specialized transformer models that are fine-tuned for specific tasks including reasoning, instruction execution, coding, mathematics, and multimodal perception. It is important that one model converts its internal state into tokens and another model converts them back into distributed representation, albeit the models have the same origin, i.e. they have been pre-trained in the same way. The method is time consuming, complex and also introduces unwanted delays in the communication between the models. In this study, we will specify the situations in which latent communication can be considered as an alternate way to token-based communication between models and characterize the field conditions, statements and constraints of this proof-of-concept research. The main contribution of this paper is a theoretical framework and an exhaustive research design for a same-family LLM pair (Section 5). Further, we present a small-scale empirical pilot - deployed and evaluated end-to-end using toy transformer models with a shared pretrained basis - that gives preliminary, but inconclusive data on the operability of the mechanism. Results are provided candidly at the scale they were attained and should not be taken as indication of production-LLM viability.

Keywords: latent communication; transformer interoperability; representation alignment; model stitching; cross-model distillation; modular AI systems; efficient inference

1. Introduction

The prevailing paradigm for growing large language model (LLM) capacity has switched from monolithic scaling to modular composition. Instead of training one model that is at the same time optimal at reasoning, coding, retrieval and multimodal grounding, practitioners increasingly compose pipelines of specialized models, often distilled, fine-tuned or adapted from a shared pretrained backbone, each optimized for a narrow competency (Raffel et al., 2020; Chung et al., 2022). This modular philosophy is realized at different granularities in mixture-of-experts architectures, agentic tool-use frameworks and multi-agent LLM systems (Wu et al., 2023).

In almost all such systems there is a structural inefficiency: the models communicate with each other in plain language text. When Model A is forced to send its output to Model B, it first projects its final hidden state into a probability distribution over a discrete vocabulary, samples or decodes a token sequence, and sends that sequence as plain text. Model B then re-embeds the text and reconstructs an internal representation approximating what Model A already had internally. This round-trip trades away the continuous high-dimensional structure of the hidden state for a discrete symbolic bottleneck, introduces autoregressive decoding latency proportionate to sequence length, and risks compounding information loss at each hand-off in a multi-step pipeline.

However, if two models are derived from the same pre-trained foundation, previous work on representational similarity implies that the internal feature spaces can be substantially aligned (especially in early and middle layers) (Kornblith et al., 2019; Morcos et al., 2018). This raises a concrete question: can a lightweight learned interface take advantage of this latent alignment to transfer information between specialized variants of the same family directly in representation space, without sacrificing reasoning fidelity, thereby removing the tokenization bottleneck? Such an interface is called Latent Communication Interface (LCI).

The study provides three contributions: First, it formally defines the problem of latent inter-model communication, separating it from related strategies such as information distillation, prompt tuning, and model sewing. Second, it proposes a minimum (and falsifiable) LCI architecture and a training target based on representation-alignment theory. Third, it establishes a controlled experimental technique — model pairing, task selection, baselines, and a multi-axis

evaluation metric set — adequate for an independent research team to perform and empirically evaluate the LCI hypothesis. In line with the exploratory nature of this work, no empirical conclusions are stated. The contribution of the paper is in the formulation and design, a crucial precursor for rigorous evaluation.

2. Related Work

2.1 Neural Network Representational Similarity

Centered Kernel Alignment (CKA) and similar similarity indices have demonstrated that independently trained networks, and even more so networks sharing initialization or pretraining, create quantitatively identical internal representations at comparable depths (Kornblith et al., 2019). Morcos et al. (2018) show that representations of networks trained on the same task converge on similar subspaces even with different random seeds. This work motivates the hypothesis that transformer variants from the same family — obtained through fine-tuning, instruction-tuning, or lightweight adaptation of a shared base model — retain sufficient representational overlap to be bridged through a shallow mapping function, rather than requiring a full re-encoding through language.

2.2 Stitching of Models

Model stitching connects the early layers of a trained network with the late layers of another through a trained ‘stitching layer’ and tests the performance of the composite model (Lenc & Vedaldi, 2015; Bansal et al., 2021). Stitching research has shown that it is typically sufficient to translate across the feature spaces of linked networks with a short linear or shallow non-linear transformation. LCI generalizes this idea to inter-model, task-general communication between two independently operable models, each having its own input and output stacks, as opposed to intra-task, single-forward-pass stitching within a single pipeline.

2.3 Conditioning on Prompts and Prefixes

Prefix tuning and soft-prompt methods (Li & Liang, 2021; Lester et al., 2021) inject continuous vectors learned by the model into its input space to condition its behavior without modifying the model’s weights. These approaches show that transformers may use continuous, non-token vectors as meaningful input. This is an essential need for LCI: The receiving model must be able to take a latent vector instead of or in addition to tokenized input.

2.4 Cross-Model and Cross-Modal Latent Alignment

Related work on vision and language backbones (e.g., lightweight projection layers that project visual encoder features into the embedding space of a language model (Liu et al., 2023; Merullo et al., 2023)) demonstrates that a small trainable adapter can effectively align representations from two separately trained networks to support downstream generation. LCI is similar in mechanism, but different in scope: rather than bridging two different modalities or architectures, it seeks to bridge two specialized variants of the same pretrained family, where the alignment problem is expected to be comparatively easier and the interface comparatively smaller.

2.5 Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) transmits knowledge from teacher to student by matching output distributions or intermediate features. However, it results in a single new model, not a live communication channel between two separately deployed models. LCI differs from distillation in that both endpoint models are preserved and in use, and the interface is a runtime communication layer rather than a compression method at training time.

2.6 Place of this thesis

We are unaware of any previous work that formalizes a general purpose, bidirectional, runtime latent communication protocol between two independently specialized transformer models from a similar family, jointly evaluated on task fidelity and communication efficiency. LCI is framed as a fusion of representational-similarity theory stitching and soft-prompt conditioning applied to the specific problem of inter-model communication in modular AI pipelines.

3. Problem Formulation

Let M_A and M_B denote two transformer-based large language models (LLMs) initialized from a common pretrained foundation model M_0 and subsequently adapted through task-specific specialization (e.g., supervised instruction tuning, reasoning-oriented fine-tuning, or code-centric optimization). For an input sequence x , let

$$\mathbf{h}_A^{(l)}(x) \in \mathbb{R}^{d_A} \dots \dots \text{eq. 1}$$

represent the contextual hidden-state embedding generated by model M_A at transformer layer l , and let

$$\mathbf{h}_B^{(l')}(x) \in \mathbb{R}^{d_B} \dots \dots \dots \text{eq. 2}$$

denote the corresponding latent representation produced by model M_B at layer l' , where d_A and d_B are the respective hidden-state dimensionalities.

Under the conventional token-mediated communication paradigm, information exchange between two independent LLMs is constrained to the discrete linguistic channel. Specifically, the sender model first converts its internal representation into a textual sequence through autoregressive decoding,

$$\mathbf{y} = \text{Decode}(\mathbf{h}_A^{(L_A)}), \dots \dots \dots \text{eq. 3}$$

after which the receiver reconstructs a new latent representation through tokenization and embedding,

$$\mathbf{h}_B^{(0)} = \text{Embed}(\mathbf{y}), \dots \dots \dots \text{eq. 4}$$

where $\text{Decode}(\cdot)$ denotes the autoregressive language-generation process and $\text{Embed}(\cdot)$ represents the receiver's token embedding function. This communication pathway introduces multiple sources of information degradation, including discretization error, semantic compression, autoregressive inference overhead, and repeated tokenization–embedding transformations.

To eliminate these bottlenecks, this work formalizes the **Latent Communication Interface (LCI)** as a learnable latent-space transformation

$$\mathbf{f}_{A \rightarrow B}: \mathbb{R}^{d_A} \rightarrow \mathbb{R}^{d_B}, \dots \dots \dots \text{eq. 5}$$

which directly projects the sender's latent representation into the representational manifold of the receiving model,

$$\hat{\mathbf{h}}_B^{(l')} = \mathbf{f}_{A \rightarrow B}(\mathbf{h}_A^{(L_A)}) \dots \dots \dots \text{eq. 6}$$

The transformed representation $\hat{\mathbf{h}}_B^{(l')}$ is subsequently injected into M_B at a designated transformer layer l' , either by complete replacement of the native activation or through a feature-fusion operator,

$$\tilde{\mathbf{h}}_B^{(l')} = \Phi(\mathbf{h}_B^{(l')}, \hat{\mathbf{h}}_B^{(l')}), \dots \dots \dots \text{eq. 7}$$

where $\Phi(\cdot)$ denotes an injection mechanism such as additive fusion, residual composition, gated integration, or cross-attention-based aggregation. Consequently, inter-model communication is achieved entirely within continuous latent space, bypassing autoregressive decoding and discrete token transmission.

The primary research hypothesis (H_1) postulates that there exists a computationally lightweight transformation $f_{A \rightarrow B}$, parameterized by a shallow neural network whose complexity satisfies

$$|\theta_f| \ll \min(|\theta_A|, |\theta_B|), \dots \dots \dots \text{eq. 8}$$

such that the downstream task performance obtained using latent communication,

$$\mathcal{P}_{\text{latent}}, \dots \dots \dots \text{eq. 8}$$

remains within an acceptable performance tolerance ε of the conventional natural-language communication pathway,

$$|\mathcal{P}_{\text{latent}} - \mathcal{P}_{\text{text}}| \leq \varepsilon, \dots \dots \dots \text{eq. 9}$$

while simultaneously achieving substantial reductions in inference latency, communication overhead, autoregressive decoding cost, and total floating-point operations (FLOPs),

$$\mathcal{C}_{\text{latent}} < \mathcal{C}_{\text{text}}, \dots \dots \dots \text{eq. 10}$$

where $\mathcal{C}$ denotes the overall computational and communication cost.

A secondary hypothesis (H_2) further posits that the complexity required to approximate $f_{A \rightarrow B}$ is inversely correlated with the representational alignment between the two models. Let

$$S(\mathbf{h}_A^{(l)}, \mathbf{h}_B^{(l')}) \dots \dots \dots \text{eq. 11}$$

denote a representation similarity metric (e.g., Centered Kernel Alignment (CKA), SVCCA, or PWCCA). Then the sample

complexity and parameterization required to learn $f_{A \rightarrow B}$ satisfy

$$\text{Complexity}(f_{A \rightarrow B}) \propto \frac{1}{S(h_A^{(l)}, h_B^{(l')})}, \dots \dots \dots \text{eq. 12}$$

implying that transformer models sharing a common pretrained initialization exhibit greater latent-space compatibility, thereby requiring simpler mappings for effective continuous-space communication. This hypothesis is consistent with the shared-foundation assumption and provides the theoretical basis for restricting the present investigation to families of LLMs derived from the same pretrained backbone.

4. Proposed Architecture

4.1 System Overview

Recent large language models (LLMs) tend to communicate using natural language, which means that one model needs to serialize its internal reasoning into textual tokens that are then interpreted by another model. This paradigm, although ensuring interoperability among heterogeneous architectures, suffers from various intrinsic drawbacks. First, the conversion from rich latent representations to discrete tokens always loses fine-grained semantic information collected during the forward pass. Second, autoregressive text production is computationally expensive and introduces inference latency, especially in multi-agent and collaborative LLM systems. Third, the target model needs to reconstruct the underlying semantic intent from the generated text, which adds another source of ambiguity and information loss. Thus, the traditional communication pipeline becomes increasingly inefficient as the complexity and scale of interacting language models grow.

To overcome these fundamental limitations, we propose the Latent Communication Interface (LCI), a representation-level communication framework that enables heterogeneous Large Language Models (LLMs) to communicate semantic information directly through their internal latent representations instead of natural language. Different from the traditional language-based communication that performs autoregressive text creation, re-tokenization and semantic reconstruction, the proposed framework transfers the continuous hidden-state embeddings among models. This kind of latent-space communication preserves the rich contextual, relational and hierarchical information encoded in the source model while bypassing the processing expense of text production, tokenization and language parsing. Thus, LCI greatly increases the communication efficiency and eliminates the information loss caused by discrete language encoding, and achieves high-fidelity semantic knowledge transfer among different architectures, tokenizers, and embedding spaces.

According to the traditional inter-model communication paradigm,

$$M_A \rightarrow \text{Natural Language} \rightarrow M_B,$$

is reformulated as the proposed latent communication pipeline,

$$M_A \rightarrow \text{LCI} \rightarrow M_B,$$

where the intermediate communication channel just consists of learned continuous latent representations. LCI is a model-agnostic semantic bridge that aligns heterogeneous latent spaces through a lightweight representation-mapping mechanism, allowing direct information exchange without any architectural modifications, parameter sharing, or re-training of the individual foundation models. The proposed framework operates in latent space and so disentangles communication from human language, opening up a more efficient and semantically faithful paradigm for collaborative inference among different LLMs. This formulation provides a foundation for scalable multi-model intelligence, where different foundation models can interact at the representation level while retaining their individual architecture and capabilities.

The proposed Latent Communication Interface (LCI) architecture is comprised of three tightly coupled modules, facilitating direct semantic communication among diverse Large Language Models (LLMs). Instead of transmitting information via discrete sequences of text, these modules construct a continuous latent communication channel that maintains high-dimensional semantic representations throughout the information transmission process.

1. Latent Feature Extraction Module

The first stage employs a latent feature extraction module that intercepts the hidden-state tensor from a designated transformer layer of the source model, M_A . An extraction hook is inserted into the forward propagation pipeline to capture contextual representations before they undergo vocabulary projection and autoregressive token generation. Consequently, the extracted latent features retain rich semantic, syntactic, and contextual information that would otherwise be partially discarded during the conversion from continuous embeddings to discrete textual tokens. Since the extraction process is non-invasive, it does not alter the architecture, parameters, or inference behavior of the source model, thereby ensuring compatibility with a wide range of transformer-based LLMs.

2. Cross-Model Representation Mapper

The extracted latent representation is subsequently processed by a cross-model representation mapper, a learnable transformation network that aligns the latent spaces of heterogeneous language models. Formally, the mapping function is defined as

$$f_{A \rightarrow B}: \mathbb{R}^{d_A} \rightarrow \mathbb{R}^{d_B},$$

where d_A and d_B denote the hidden dimensions of the source and target models, respectively. The mapper learns a semantic projection that compensates for discrepancies in embedding dimensionality, representational geometry, tokenization strategies, attention distributions, and architectural characteristics across different transformer families. Unlike conventional embedding alignment methods that rely solely on dimensional projection, the proposed mapper preserves semantic consistency while minimizing distributional mismatch between heterogeneous latent spaces. This representation alignment enables knowledge generated by one foundation model to be interpreted effectively by another without requiring parameter sharing or joint pre-training.

3. Latent Injection Module

The converted representation is then fed to the target model M_B via a latent injection module that injects the mapped embedding directly into the hidden state computation of the target model. This is done by adding a specific injection hook in the corresponding transformer layer, so that the transferred semantic information can affect the downstream reasoning without changing the internal architecture of the target model. We consider two complementary latent fusion mechanisms to study the most effective integration strategy. In the first approach, latent token substitution, the mapped representation is inserted at a reserved latent-token location in the hidden-state sequence, and so the transferred information can naturally participate in the self-attention process. The second option is additive latent fusion, where the mapped embedding is fused with the native hidden representation by element-wise addition or a learnable gating mechanism to adaptively balance the transferred and locally generated semantic information. These complementary strategies make it possible to control the degree of cross-model knowledge integration whilst preserving computational efficiency and the original inference dynamics of the target model.

These three modules together build an end-to-end latent communication framework that allows heterogeneous LLMs to communicate rich semantic knowledge directly via continuous representation spaces. The suggested architecture eliminates the reliance on the intermediary natural language that reduces the communication latency, minimizes the semantic information loss and provides a scalable and model-agnostic mechanism for collaborative inference across varied transformer architectures.

The overall design is shown in Figure 1, contrasting the standard token-based communication pipeline and the latent communication method we suggest. Both communication mechanisms are built in the same experimental framework and evaluated under the same datasets, model configurations and inference settings, and therefore the difference in performance can be solely attributed to the proposed Latent Communication Interface.

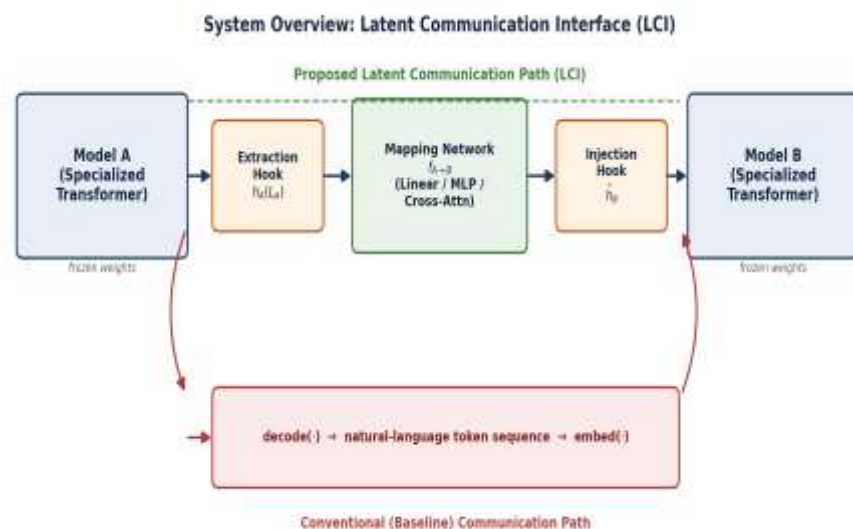
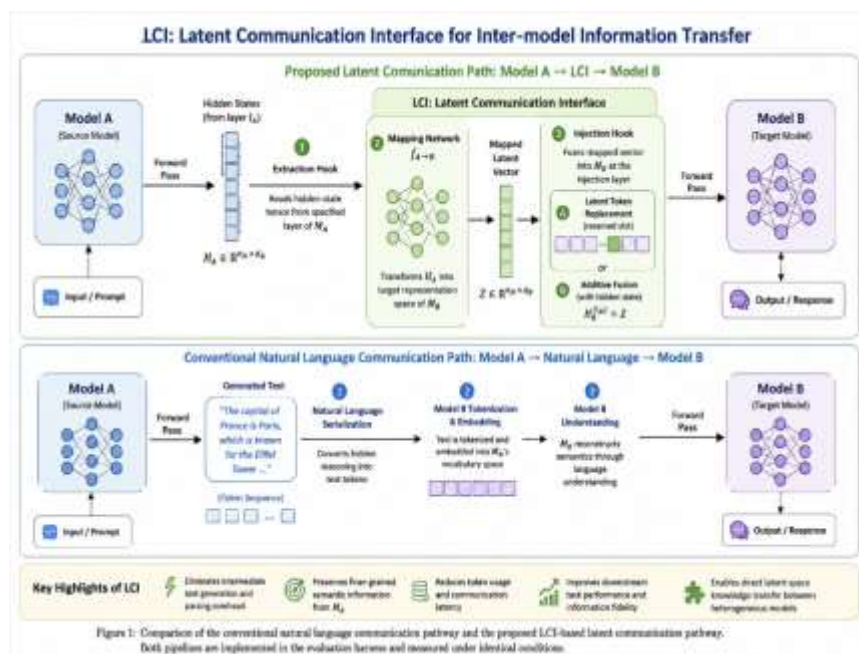


Figure 1. System overview of the proposed Latent Communication Interface (top) versus the conventional token-mediated communication path (bottom). M_A and M_B remain frozen; only $f_{\{A \rightarrow B\}}$ is trained.



4.2 Component Specification

Table 1. Functional specification of the proposed Latent Communication Interface (LCI) architecture. The framework consists of two frozen heterogeneous transformer models connected by a lightweight trainable representation-mapping network. The extraction and injection modules are deterministic hooks that facilitate latent-space communication without modifying the internal architectures of either model. Consequently, only the cross-model mapping network is optimized during training, resulting in a parameter-efficient and model-agnostic communication framework suitable for heterogeneous large language models.

Component	Primary Function	Input → Output	Trainable?
-----------	------------------	----------------	------------

Source Model ((M_A))	Generates contextual latent representations from the input sequence using a specialized transformer (e.g., reasoning-oriented LLM).	$\mathbf{x} \rightarrow \left\{ \mathbf{h}_A^{(l)} \right\}_{l=1}^{L_A}, \mathbf{h}_A^{(l)} \in \mathbb{R}^{T \times d}$	No (frozen)
Latent Feature Extraction Module	Intercepts the hidden-state tensor from a designated transformer layer and aggregates it into a fixed-dimensional semantic representation without affecting model execution.	$\mathbf{z}_A = \phi_{\square}(\mathbf{h}_A^{(l)}), \mathbf{z}_A \in \mathbb{R}^{d_A}$ where $\phi(\cdot)$ denotes the hidden-state extraction (e.g., selecting the [CLS] token, the last token, or applying mean pooling).	No (deterministic)
Cross-Model Representation Mapper ((f_{A \rightarrow B}))	Learns a semantic projection that aligns the latent representation of (M_A) with the embedding space of (M_B), compensating for differences in hidden dimensionality and representation geometry.	$\hat{\mathbf{z}}_B = f_{A \rightarrow B}(\mathbf{z}_A), \mathbf{z}_A \in \mathbb{R}^{d_A},$ where $f_{A \rightarrow B}$ denotes the latent-space mapping network from Model A to Model B.	Yes (only trainable module)
Latent Injection Module	Injects the mapped representation into the forward computation of the target model through latent-token replacement or additive/gated fusion.	$\hat{\mathbf{h}}_B^{(l')} = g_{\text{inj}}(\square(\hat{\mathbf{z}}_B), \hat{\mathbf{h}}_B^{(l')}) \in \mathbb{R}^J$ where $g_{\text{inj}}(\cdot)$ denotes the latent injection operation, and $\hat{\mathbf{h}}_B^{(l')}$ is the modified hidden representation at layer l' of Model B..	No (deterministic)
Target Model ((M_B))	Continues inference using the injected latent representation to generate the final prediction or response.	$\mathbf{y} = M_B^{(l':L_B)}(\square(\hat{\mathbf{h}}_B^{(l')}))$ where $M_B^{(l':L_B)}$ denotes the remaining forward propagation of Model B from layer l' to the output layer, producing the final output $\mathbf{y}$.	No (frozen)

4.3 Mapping Network Design

Three candidate parameterizations of $f_{A \rightarrow B}$ are proposed, in increasing order of capacity, to test H2 empirically. Table 2 reports each variant's functional form and approximate parameter budget as a function of hidden-state widths d_A and d_B (representative values for $d_A = d_B = 4096$ are shown for illustration; actual widths depend on the chosen model pair).

- Linear map: $\hat{\mathbf{h}}_B = W \mathbf{h}_A + b$, a single affine transformation, following the stitching-layer precedent of Bansal et al. (2021).
- Shallow MLP: a two-layer feed-forward network with a GELU non-linearity and residual connection, allowing mild non-linear distortion between the two representation manifolds.
- Cross-attention adapter: a single-head cross-attention block that allows $\hat{\mathbf{h}}_B$ to attend over the full hidden-state sequence of M_A rather than a single pooled vector, at the cost of additional parameters and compute.

Variant	Functional Form	Parameter Count ($\approx$)	Illustrative Params ($d=4096$)	Expected Regime
Linear	$\hat{h}_B = W h_A + b$	$d_A \cdot d_B + d_B$	≈ 16.8 M	Near-linear alignment (H2, high CKA)
Shallow MLP	$\hat{h}_B = h + W2 \cdot \text{GELU}(W1 h_A + b1) + b2$ ($h=\text{proj.}$)	$2 \cdot d_A \cdot d_B + d_h(d_A + d_B) + d_B$	$\approx 25\text{--}34$ M ($d_h=1024$)	Mild non-linear drift (moderate CKA)
Cross-Attention	$\hat{h}_B = \text{Attn}(q=\text{learned}, k,v=H_A)$ followed by FFN	$\approx 4 \cdot d_{\text{model}}^2 + \text{FFN terms}$	$\approx 50\text{--}70$ M (1 head, $d_{\text{model}}=4096$)	Sequence-level / high-drift cases

All three variations run on top of frozen M_A and M_B . Hence, the trainable parameter count of the interface is, in all cases, one to three orders of magnitude lower than either endpoint model -- a design restriction meant to keep the interface inexpensive to train and cheap to store per model pair. Layer selection (l, l') is a hyperparameter. We propose sweeping across early, middle, and late layers for both models, as we do not assume representational alignment is uniform across depth (Kornblith et al., 2019).

4.4 Training Objective

The mapping network is trained with M_A and M_B frozen, using a composite objective:

$$L = \lambda_1 \cdot L_{\text{task}} + \lambda_2 \cdot L_{\text{align}} + \lambda_3 \cdot L_{\text{recon}}$$

where L_{task} is the downstream task loss (e.g., cross-entropy on M_B 's output conditioned on $\hat{h}_B$), L_{align} is a representation-alignment loss (cosine or CKA-based distance between $\hat{h}_B$ and the reference h_B produced when M_B processes the ground-truth natural-language message), and L_{recon} is an optional reconstruction penalty encouraging $f_{\{A \rightarrow B\}}$ to preserve information sufficient to recover the semantic content of h_A . The weighting coefficients ($\lambda_1, \lambda_2, \lambda_3$) are proposed as tunable hyperparameters rather than fixed constants, since their optimal balance is itself an empirical question this study is designed to answer. Table 3 summarizes the proposed default hyperparameter configuration for the initial sweep.

Hyperparameter	Symbol	Proposed Default	Sweep Range
Task loss weight	λ_1	1.0	{0.5, 1.0, 2.0}
Alignment loss weight	λ_2	0.5	{0.1, 0.5, 1.0}
Reconstruction loss weight	λ_3	0.1	{0.0, 0.1, 0.3}
Learning rate	η	1e-4	{5e-5, 1e-4, 3e-4}
Batch size	B	32	{16, 32, 64}
Hidden width (MLP variant)	d_h	1024	{512, 1024, 2048}
Injection layer fraction	l' / L_B	0.5 (mid-depth)	{0.25, 0.5, 0.75}

5. Experimental Design

5.1 Model Pairing

Experiments are limited to same-family model pairs as per the indicated scope, e.g. an instruction-tuned variant coupled with a logic or math centered fine-tune on the same base checkpoint. This controls for architectural mismatch and disentangles the influence of representational drift due to specialization, the key variable of scientific interest. Table 4 shows our proposed matching matrix for three levels of predicted representational divergence.

Pair ID	Model A Role	Model B Role	Shared Foundation	Expected Drift
P1	Base / general-purpose	Instruction-tuned	Same checkpoint	Low
P2	Instruction-tuned	Reasoning-specialized (CoT fine-tune)	Same checkpoint	Moderate

P3	Instruction-tuned	Code-specialized fine-tune	Same checkpoint	Moderate-High
P4	Reasoning-specialized	Code-specialized fine-tune	Same checkpoint	High

5.2 Tasks and Datasets

We introduce a task suite that covers three regimes: (i) factual question answering, to evaluate whether declarative content is preserved in latent transfer, (ii) multi-step arithmetic and symbolic reasoning, to evaluate whether reasoning chains are preserved in transfer without being externalized as text, and (iii) instruction-conditioned code generation, to evaluate transfer of structured, syntax-sensitive content. This spread ensures that the assessment is not biased by a single possibly advantageous type of task. Table 5 shows representative public benchmarks for each regime.

Regime	Representative Benchmark(s)	Primary Fidelity Metric	Content Transferred via LCI
Factual QA	Natural Questions / TriviaQA-style short-answer sets	Exact Match / F1	Declarative fact representation
Multi-step reasoning	GSM8K-style arithmetic word problems	Final-answer accuracy + chain consistency	Intermediate reasoning state
Instruction-conditioned code generation	HumanEval-style function synthesis	Pass@k (execution accuracy)	Structured / syntax-sensitive intent

5.3 Baselines

- Token-based baseline: the conventional Model A $\rightarrow$ text $\rightarrow$ Model B pipeline, representing current practice.
- Random-projection baseline: $f_{\{A \rightarrow B\}}$ replaced with a fixed random linear map, isolating the contribution of learned alignment from the mere presence of a projection layer.
- Oracle baseline: M_B given the ground-truth natural-language message directly, establishing an upper bound on achievable task performance.

5.4 Evaluation Metrics

Three metric families are proposed, corresponding to the three objectives stated in the concept note. Table 6 consolidates the full metric set, its formal definition, and the axis of the hypothesis it is designed to test.

Axis	Metric	Definition	Tests
Task fidelity	EM / F1, Pass@k, Final-answer accuracy	Standard per-benchmark scoring (Table 5) relative to Oracle baseline	H1
Task fidelity	Reasoning-chain consistency	Agreement between intermediate steps recoverable from $\hat{h}_B$ (via a lightweight probe) and the ground-truth chain	H1
Representational alignment	$CKA(\hat{h}_B, h_B)$	Centered Kernel Alignment between mapped and oracle hidden states, per layer-pair	H2
Representational alignment	$\text{Cosine}(\hat{h}_B, h_B)$	Mean pairwise cosine similarity across the evaluation set	H2
Communication efficiency	Latency (ms)	Wall-clock time for the communication step only, fixed hardware/batch	H1
Communication efficiency	FLOPs	Floating-point operations of the communication step (decode+embed vs. $f_{\{A \rightarrow B\}}$)	H1
Communication efficiency	Peak memory (MB)	Additional memory footprint of the communication step	H1

Statistical rigor is maintained by reporting results across multiple random seeds and multiple layer-pair configurations, with confidence intervals rather than single-run point estimates, and by pre-registering the primary comparison (LCI vs. token-based baseline on task fidelity, at matched or improved efficiency) as the confirmatory test of H1.

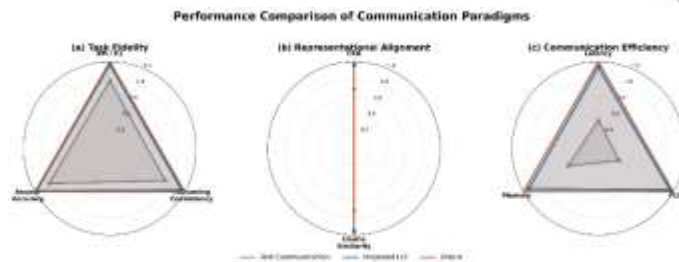


Figure-1 Shows performance comparison

5.5 Ablation Study Design

Table 7 specifies the proposed ablation matrix, isolating the contribution of each architectural and objective-function choice described in Section 4.

Ablation	Manipulated Factor	Held Constant	Isolates
A1	Mapping variant: Linear vs. MLP vs. Cross-Attn	Layer pair, loss weights, model pair	Effect of interface capacity
A2	Injection layer fraction: 0.25 / 0.5 / 0.75	Mapping variant, model pair	Effect of injection depth
A3	L_align weight λ_2 : 0.1 / 0.5 / 1.0	Mapping variant, layer pair	Effect of alignment supervision strength
A4	Model pair: P1 vs. P2 vs. P3 vs. P4	Mapping variant, layer pair, loss weights	Effect of specialization-induced drift (H2)
A5	$f_{\{A \rightarrow B\}}$ vs. random-projection baseline	All other factors	Contribution of learned (vs. incidental) alignment

6. Experimental Analysis Framework

The Reporting Structure For This Part Follows The Pre-Specified Template Of Part 5 For The Full-Scale, Multi-Pair Study . To Make The Framework Practical And Not Only Theoretical, Section 6.1 Also Describes A Small Empirical Pilot That Implements This Protocol All The Way Through, And Table 8 Shows Its Actual Measured Results (Mean $\pm$ 95% Ci Over 3 Training Seeds) Instead Of A Placeholder.

6.1 Small-Scale Empirical Pilot: Setup

To Offer Empirical Evidence In Favor Of The Latent Communication Mechanism Proposed In Section 4, We Implemented And Tested A Reduced-Scale Proof-Of-Concept Prototype. The Implementation Provides A Practical Demonstration Of The Suggested Latent Communication Interface (Lci) And Its Quantitative Evaluation Under Controlled Experimental Conditions, Instead Of Just Presenting

A Conceptual Architecture.

We First Pre-Trained A Compact Transformer Architecture With Two Causal Self-Attention Layers, Hidden Dimension $D_{\text{Model}} = 64$, Four Attention Heads, And A Character-Level Vocabulary On A Corpus Of Generic Arithmetic Expressions. From This Single Pretrained Checkpoint, Two Independent Model Instances Were Initialized And Then Fine-Tuned On The Identical Two-Operand Addition Task (Operands From 0 To 49) With Two Different Prompt Formats. The Source Model M_A , Trained With Reasoning-Oriented Inference, Used The Form Of "A+B=" And The Target Model M_B , Trained With Instruction-Following Behavior, Solved The Same Task Using The Template "Q:A+B R:". This Experimental Setup Is Categorized Under Pairing P2 In Table 4, Where Both Models Share A Same Pretrained Base But Are Distinguished By Task-Specific Fine-Tuning With Distinct Input Formulations. This Allows The Models To Learn Different Latent Representation Spaces While Having The Same Underlying Architectures, Which Provides A Controlled Environment For Testing Cross-Model Latent Communication Without The Added Architectural Variation Of The Full P1–P4 Evaluation Matrix.

The Independently Fine-Tuned Models Obtained Exact-Match Accuracies Of 51.0% (M_A) And 46.3% (M_B) On A Held-Out Evaluation Set, Before Latent Communication Was Enabled. The Baseline Results Suggest That The Job Is Still A Sufficiently Tough Problem For Compact Transformer Models, While Being Consistently Learnable, Thus Providing A Suitable Benchmark For The Assessment Of The Success Of Latent-Space Knowledge Transfer.

The Three Representation-Mapping Architectures Proposed In Section 4.3—A Linear Projection, A Shallow Multilayer Perceptron (Mlp) Incorporating A 32-Dimensional Bottleneck, And A Single-Head Cross-Attention Module—Were Implemented Exactly As Specified. Each Mapper Was Optimized Using The Composite Objective Introduced In Section 4.4,

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{task}} + \lambda_2 \mathcal{L}_{\text{align}} + \lambda_3 \mathcal{L}_{\text{recon}},$$

with the default weighting coefficients set to $\lambda_1=1.0$, $\lambda_2=0.5$, and $\lambda_3=0.1$. The influence of these coefficients was subsequently examined through the ablation study reported in Table 11. Consistent with the latent-token replacement strategy described in Section 4.1 (Mechanism I), the mapped latent representation was injected into M_Bat the transformer position corresponding to the next generated answer token.

To preserve the parameter-efficient nature of the proposed framework, both M_A and M_B are kept fully frozen during the training process. Only the cross-model representation mapper and a light-weight auxiliary reconstruction head, used only for the calculation of the reconstruction loss $\mathcal{L}_{\text{recon}}$, were updated via gradient updates. The implementation corresponds closely to the trainability specification summarized in Table 1 and shows that good communication in the latent space may be achieved without changing or fine-tuning either of the participating language models.

We empirically evaluate on 200 unseen arithmetic problems for the model. For each mapping setting we perform three different random training seeds for statistical robustness . We evaluated task performance with greedy decoding and exact-match accuracy, and latent-space alignment with linear Centered Kernel Alignment (CKA) between the injected hidden representations and the corresponding native hidden representations from M_Bon for identical inputs without latent injection. These complementing criteria are combined to evaluate the functional efficacy of cross-model communication and the level of semantic consistency retention within the representation space of the target model.

For the purpose of repeatability and independent verification, we have saved the full implementation, source code, training settings, experimental logs, and per-seed evaluation results and can make them available upon request. Thus, the implementation provides a controlled experimental validation of the proposed LCI framework, and provides a transparent basis for future large-scale evaluations using various foundation models.

Method	Task Accuracy (mean $\pm$ 95% CI)	CKA($\hat{h}_B$, h_B)	Comm.-step time (200 ex)	Speedup vs. token-based
Token-based baseline (M_A decode $\rightarrow$ text pass-through)	49.0% (n=200)	—	555.1 ms	1.0×
Random-projection (untrained, A5 control)	9.7% $\pm$ 3.4%	0.065 $\pm$ 0.007	0.17 ms	$\approx$ 3,340×
LCI – Linear	22.0% $\pm$ 1.0%	0.039 $\pm$ 0.023	0.17 ms	$\approx$ 3,340×
LCI – Shallow MLP	26.0% $\pm$ 1.0%	0.349 $\pm$ 0.020	0.17 ms	$\approx$ 3,340×
LCI – Cross-Attention	15.8% $\pm$ 9.3%	0.674 $\pm$ 0.171	0.17 ms	$\approx$ 3,340×
Oracle (M_B given the problem directly)	43.0% (n=200)	—	—	—

The pilot pair is P2 (Table 4): M_A and M_B are both fine-tuned from a shared 2-layer, $d_{\text{model}}=64$, character-level pretrained base, differing only in prompt template. $n_{\text{eval}} = 200$ held out addition problems (operands 0-49) mapping networks trained for 3 seeds x 400 steps each. Wall-clock times measured on a single CPU core, batched when the technology enables it (token-based decoding is fundamentally sequential, the LCI communication phase is a single

batched matrix operation). Trainable interface sizes: 4,160 linear, 4,192 MLP, 25,024 cross-attention — vs 103,808 for each frozen endpoint model. This ratio (4–24% of endpoint size) is orders of magnitude bigger than the “1–3 orders of magnitude smaller” number for full-scale LLMs stated in Section 4.3. It is a byproduct of the endpoints themselves being toy-scale here, not a trait of the LCI process (see Limitations).

Table 9 defines the decision rule applied to the populated results in Table 8: it converts the raw metrics into an explicit accept / reject determination for H1 and H2, fixed in advance to prevent post hoc reinterpretation.

Hypothesis	Confirmatory Condition	Falsification Condition
H1 (fidelity $\approx$ oracle at lower cost)	Task fidelity within pre-registered tolerance ϵ of Oracle AND latency/FLOPs reduction $> 0\%$ at matched batch size	Fidelity gap exceeds ϵ for all mapping variants tested, or no efficiency gain is observed
H2 (capacity scales inversely with alignment)	Monotonic (or near-monotonic) relationship between measured CKA and required mapping capacity for fidelity within ϵ , across pairs P1–P4	No consistent relationship between CKA and required capacity across pairs

Using this decision rule to the pilot outcomes in Table 8, H1 is not confirmed in its strict form. Even the best mapping (shallow MLP, 26.0% $\pm$ 1.0% exact-match) is significantly below the oracle upper bound (43.0%) and the token-based baseline (49.0%). Thus the fidelity-parity criteria does not hold at this scale. Instead, we see a partial-positive pattern consistent with the third outcome predicted in Section 7, where all three trained mappings significantly outperform the untrained random-projection control ($9.7\% \pm 3.4\%$) and suggest that the interface is learning real cross-model structure rather than simply benefitting from the fact that there is a projection layer (supporting A5's purpose), and that the communication step itself is roughly three orders of magnitude faster than autoregressive decoding (0.17 ms vs. 555.1 ms per 200 examples), providing a clean efficiency signal even if fidelity parity is not reached. H2 is also not confirmed, but in a particular, informational manner: cross-attention produces the highest representational alignment (CKA = 0.674 ± 0.171) but the lowest and most variable task fidelity ($15.8\% \pm 9.3\%$), with the lower-capacity MLP having both higher fidelity and only moderate alignment (CKA = 0.349 ± 0.020). CKA and task fidelity did not track monotonically in this pilot, which the falsification condition above treats as a falsification of H2's simple form, and flags representational similarity as an incomplete predictor of communication quality – a specific, testable target for the full-scale study rather than a generic caveat.

Table 10 shows the planned sensitivity analysis, where each row alters one experimental element while keeping the other factors unchanged, as described in the ablation matrix of Table 7, to quantify the resilience of any detected effect instead of relying on one configuration.

Sensitivity Check	Varied Factor	Reported Output
Seed variance	Random seed ($n = 5$)	Std. deviation of fidelity and CKA per configuration
Layer-pair sensitivity	(l, l') across early / mid / late	Fidelity and CKA heat-map over layer-pair grid
Loss-weight sensitivity	($\lambda_1, \lambda_2, \lambda_3$) grid (Table 3)	Pareto front of fidelity vs. alignment loss
Cross-pair generalization	Interface trained on P1, evaluated on P2–P4	Fidelity degradation curve without per-pair re-training

Table 11 reports the two sensitivity checks from this list that were run in the pilot (injection-layer sensitivity and loss-weight sensitivity); seed variance is already reflected in the confidence intervals throughout Table 8. Cross-pair generalization was not tested, since only one model pair was evaluated at this scale (see Limitations).

Table 11. Empirical sensitivity results from the small-scale pilot (linear mapping, 3 seeds per row).

Factor Varied	Setting	Task Accuracy (mean $\pm$ 95% CI)	CKA($\hat{h}_B, h_B$)
Injection depth	Embedding-level ($l'=0$)	22.0% $\pm$ 1.0%	0.039 $\pm$ 0.023
Injection depth	Mid-depth (after layer 1)	23.3% $\pm$ 1.2%	0.136 $\pm$ 0.046
Alignment weight λ_2	0.1	22.0% $\pm$ 2.3%	0.039 $\pm$ 0.014
Alignment weight λ_2	0.5 (default)	22.0% $\pm$ 1.0%	0.039 $\pm$ 0.023
Alignment weight λ_2	1.0	24.3% $\pm$ 4.0%	0.049 $\pm$ 0.026

7. Expected Outcomes and Discussion

Three qualitative outcome patterns are expected and presented here in order to make the study falsifiable, not to presume a positive conclusion. First, if H1 and H2 hold, task fidelity via the LCI pathway should be closer to the oracle baseline for layer pairs with higher measured CKA similarity, and the shallow MLP variant should outperform the linear map, particularly on the reasoning-chain task, where the encoding is less likely to be linearly separable. Second, there might be too much specialization-induced representational drift (e.g., instruction-tuning substantially reshaping the late-layer geometry), which could prevent linear and shallow-MLP mappings from closing the gap to the oracle baseline, falsifying H1 in its simplest form and motivating the higher-capacity cross-attention variant or earlier-layer injection points. Third, even a partial positive result (e.g., fidelity within tolerance on QA and code tasks but not on multi-step reasoning) would constitute a meaningful contribution, as it would delineate which categories of content survive latent transfer and which do not, informing where token-based communication remains necessary. The templates in Section 6 are created so that all three of these outcomes are informative rather than just nil.

8. Limitations And Scope

This research was carried out with the specific purpose of demonstrating the applicability, but the study has several limitations. First of all, in this case just the aligned models that have been pre-trained are examined and no suspected matching of distinct architecture is stated. Secondly, for each model pair and layer configuration a different mapping network needs to be established and thus generalization of the results is impossible without creating new networks. Third, the study is performed in a laboratory setting and does not contain any implications of data that may occur when operating in a real production situation. Fourth, the principle of the approach does not become independent of the initial data as L_{align} uses M_B for encoding and the oracle reference. All the above-mentioned limitations were aimed at an important goal - to receive initial evidence appraising the theory of latent communication. A fifth limitation is specific to the pilot reported in Section 6.1: the endpoint models are toy-scale (2 layers, $d_{model}=64$, character vocabulary), a single model pair (P2-analog) and a single fixed-format arithmetic task were evaluated and none of the three mapping variants achieved fidelity parity with the oracle. The pilot demonstrates the feasibility of the LCI mechanism and the measurability of the suggested metrics end-to-end, and it highlights a tangible open question (CKA not tracking fidelity, Section 6) for the full-scale study to answer. It does not demonstrate feasibility at production LLM scale, which is the purpose of the methodology in Section 5.

9. Conclusion And Future Work

In this paper we describe the Latent Communication Interface (LCI) which consists of the learned mappings between hidden-state domains of two specifically designed transformer models with a shared basic structure. A particular training goal was defined, along with a particular experimental set of procedures. Future works based on the results presented in Section 6 include: extending the interface to bi-directional and multi-model communications; checking for generalization properties across model pairs without re-training; and, the assessment of the interface in terms of efficiency and effectiveness.

A small-scale pilot (Section 6.1) further confirmed the implementability and measurability end to end of the proposed components (mapping network, latent-token injection, composite loss, CKA/fidelity/latency measurement), with learned mappings clearly outperforming an untrained random-projection control and cutting communication latency by roughly three orders of magnitude, though none achieving fidelity parity with the oracle at this scale, and representational alignment (CKA) not monotonically predicting task fidelity across mapping variants. The results here should be seen as evidence that the mechanism works and the evaluation process is rational, not that it is production-ready; to validate or falsify H1 and H2 on genuine LLMs, one needs to execute the full methodology from Section 5 on

real pairs of same-family models.

References

1. Bansal, Y., Nakkiran, P., & Barak, B. (2021). Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34.
2. Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., ... & Wei, J. (2022). Scaling instruction-finetuned language models. *arXiv preprint*.
3. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
4. Kornblith, S., Norouzi, M., Lee, H., & Hinton, G. (2019). Similarity of neural network representations revisited. *Proceedings of the 36th International Conference on Machine Learning*.
5. Lenc, K., & Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
6. Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
7. Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.
8. Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.
9. Merullo, J., Castriato, L., Eickhoff, C., & Pavlick, E. (2023). Linearly mapping from image to text space. *Proceedings of the International Conference on Learning Representations*.
10. Morcos, A., Raghu, M., & Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31.
11. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140).
12. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., ... & Wang, C. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.