Secure Linear Alignment of Large Language Models

Anonymous Authors[1]

Abstract

Language models increasingly appear to learn similar representations, despite differences in training objectives, architectures, and data modalities. This emerging compatibility between independently trained models introduces new opportunities for cross-model alignment to downstream objectives. Moreover, it unlocks new potential application domains, such as settings where security, privacy, or competitive constraints prohibit direct data or model sharing. In this work, we propose a privacy-preserving framework that exploits representational convergence to enable cross-silo inference between independent language models. The framework learns an affine transformation over a shared public dataset and applies homomorphic encryption to protect client queries during inference. By encrypting only the linear alignment and classification operations, the method achieves sub-second inference latency while maintaining strong security guarantees. We support this framework with an empirical investigation into representational convergence, in which we learn linear transformations between the final hidden states of independent models. We evaluate these cross-model mappings on embedding classification and out-of-distribution detection, observing minimal performance degradation across model pairs. Additionally, we show for the first time that linear alignment sometimes enables text generation across independently trained models. Our code is included in the submission and will be released upon publication.

1. Introduction

Large language models (LLMs) have become the standard paradigm for language understanding and generation, with both encoder-style and autoregressive architectures achieving strong generalization across diverse tasks (Srivastava et al., 2022). Their rapid progress is driven by scaling laws that link model size, compute, and data volume to emergent capabilities (Kaplan et al., 2020; Hoffmann et al., 2022). As LLMs continue to scale, recent evidence from the Platonic Representation Hypothesis suggests that different models are becoming more aligned in their learned representations (Huh et al., 2024), raising the possibility that independently trained systems share a common latent structure.

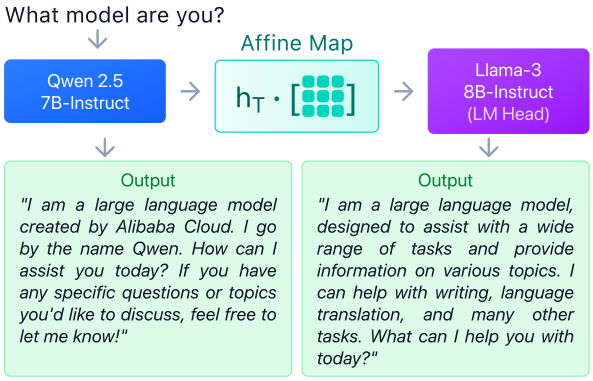

Figure 1. **Text Generation via Cross-Model Linear Alignment:** We learn an affine map from Qwen's hidden states into Llama's feature space, enabling Qwen representations to be decoded by Llama's token head. The resulting hybrid model combines Qwen's encoder/transformer blocks with Llama's output head, producing coherent responses without adopting either model's identity.

This emerging representational similarity enables new forms of model interoperability. For example, prior work on model stitching shows that independently trained models can be aligned through learned linear transformations, allowing downstream feature transfer across architectures (Bansal et al., 2021). Such compatibility supports multi-model inference pipelines, modular system design, and collaboration across heterogeneous systems (Jiang & Li, 2024; Chen et al., 2025). These capabilities are particularly valuable in settings where privacy constraints, proprietary considerations, or regulatory requirements (e.g., GDPR (Voigt & Von dem Bussche, 2017); HIPAA (Gostin & Hodge, 2000)) prohibit direct data or model sharing (Diebold, 2023; Reimsbach-Kounatze et al., 2025).

In this work we introduce HELIX (Homomorphically Encrypted Linear Inference across models), a privacy-preserving framework that exploits representational similarity to enable cross-silo inference between independent LLMs. The key insight is that when models learn similar representations, their feature spaces can be aligned through a simple linear map (Figure 1), which can be executed efficiently under homomorphic encryption. HELIX operates in two phases. During training, the client encrypts their embeddings from public data and sends them to the service provider, who computes an alignment map under encryption

and returns it. During inference, clients apply the alignment locally to their embeddings, then encrypt and send the transformed representations to the provider, who applies a linear classifier homomorphically and returns the encrypted prediction. Because the alignment and classification operations are linear, the protocol achieves 128-bit security with sub-second per-sample latency.

HELIX relies on the assumption that linear transformations can preserve task performance across models. To validate this assumption, we conduct an empirical investigation into linear alignment across diverse encoder-based and autoregressive LLMs. First, we verify that independently trained models exhibit nontrivial shared linear structure (Figure 2). We then measure alignment performance on supervised classification and out-of-distribution (OOD) detection, observing minimal degradation across models when using a fixed target linear head.

For text generation, we evaluate linear alignment across 34 model pairs using instruction-tuned generative models. Our investigation reveals two patterns: First, tokenizer compatibility strongly predicts success, with exact token match rate (r = 0.898) and Jaccard index (r = 0.822) correlating with text generation quality. Second, model scale imposes a minimum threshold, as all pairs with source models below 4B parameters produce lower quality results regardless of tokenizer alignment. We assess quality through LLM-as-a-Judge evaluation, embedding similarity to source models, and human judgment.

The remainder of this paper is organized as follows. Section 2 reviews related work on representational similarity and privacy-preserving inference. Section 3 establishes the effectiveness of linear alignment across diverse tasks. Sections 4–5 present the HELIX framework and experiments. In summary, our contributions are as follows:

- We propose HELIX, a privacy-preserving framework for cross-silo inference with independent embedding models. Leveraging the representational similarity of LLMs, HELIX encrypts only linear operations (alignment and classification) rather than full transformer models, achieving sub-second inference latency while protecting client queries.
- We provide a systematic characterization of cross-model text generation via linear alignment. Analyzing 34 LLM model pairs, we identify tokenizer compatibility and model size as primary determinants of success.
- We show that supervised linear alignment preserves classification and OOD detection performance across embedding model pairs with minimal degradation.

2. Related Work

Understanding whether neural networks converge to similar representations despite stochastic training and non-convex optimization is a central question in machine learning (Li et al., 2015; Raghu et al., 2017; Klabunde et al., 2025). A growing body of work shows that independently trained models often discover surprisingly aligned representations, particularly in overparameterized, high-capacity regimes.

Representational Similarity. Kornblith et al. (2019) introduced CKA, showing that identically structured CNNs trained from different seeds learn similar intermediate features. Morcos et al. (2018) found that wider networks, and those that generalize better, exhibit stronger representational alignment. At a higher level, the Platonic Representation Hypothesis (Huh et al., 2024) proposes that large models increasingly converge toward a shared statistical understanding of the world, with similar geometric structure emerging across modalities and architectures.

Model Stitching. Building on representational convergence, model stitching asks whether a lightweight adapter can map intermediate representations from one model into another's feature space while preserving downstream performance. Early CNN results show that shallow linear layers can stitch models trained under different settings (Lenc & Vedaldi, 2015; Bansal et al., 2021). Bansal et al. (2021) argue that stitching complements statistical similarity metrics (e.g., CKA) by testing *functional* interchangeability rather than mere geometric resemblance. Related work further shows that independently trained CNN and face-recognition models can be linearly aligned at the final layer with minimal accuracy loss (McNeely-White et al., 2020; 2022). Recent work extends stitching to transformers and LLMs: Chen et al. (2025) align hidden states across language models of different sizes via linear maps to transfer features. Unlike our approach, theirs focuses on computational efficiency. Jiang & Li (2024) stitch autoregressive and bidirectional transformers (GPT and BERT) for look-ahead text understanding, supporting the feasibility of cross-LLM alignment.

Linear Identifiability. Roeder et al. (2020) show that for a broad class of models, including supervised, contrastive, and causal language models, representations learned on the same data and architecture are *linearly identifiable*: there exists an invertible matrix W such that $Z_B \approx W Z_A$. This result provides a theoretical foundation for the use of linear alignment methods like ours. However, the identifiability theorem is guaranteed only when architectures, objectives, and data distributions match. When these conditions differ, representational equivalence may be approximate.

Homomorphic encryption. Homomorphic Encryption (HE) is widely used to protect client inputs during outsourced inference. CryptoNets (Gilad-Bachrach et al., 2016b) first demonstrated HE inference with a public model, while Gazelle (Juvekar et al., 2018) combines HE with Secure Multi-Party Computation (MPC) to accelerate end-to-end interactive private inference. More recent systems target secure transformer execution: Nexus (Zhang et al., 2024a) proposes non-interactive HE protocols, Powerformer (Park et al., 2024) improves throughput via HE-friendly attention, and BOLT (Pang et al., 2024) and Nimbus (Li et al., 2024)

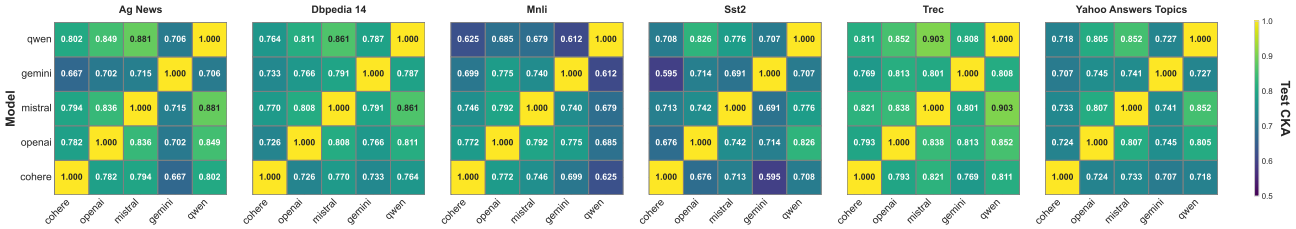

Figure 2. **Linear CKA similarity across embedding APIs.** We compute linear CKA (Kornblith et al., 2019) on vendor-provided embeddings over shared inputs from multiple datasets. CKA values range from 0.595 to 0.881, indicating substantial shared linear structure across independently trained models.

use MPC for online private inference. HETAL (Lee et al., 2023) enables encrypted fine-tuning on fixed backbones, and the Encryption-Friendly LLM architecture (Zhang et al., 2025) redesigns transformers for HE-compatible operations. All prior HE work encrypts the entire model. HELIX fundamentally differs because it extracts features locally and encrypts only the linear head.

3. Linear Alignment Characterization

In this section, we validate HELIX's core assumption by testing whether a simple linear map can align independently trained LLMs while preserving downstream task behavior.

Preliminaries For each experiment, we consider two embedding models: the target model $\mathcal{F}_A$ consists of a representation function $g_A : \mathcal{X} \to \mathbb{R}^{d_A}$ followed by a task head $f_A : \mathbb{R}^{d_A} \to \mathcal{Y}$. The source model $\mathcal{F}_B$ provides a representation function $g_B : \mathcal{X} \to \mathbb{R}^{d_B}$. Throughout, $g_A(\cdot)$ and $g_B(\cdot)$ correspond to the final hidden representations produced by their respective transformer encoders.

We focus on settings where the target head (f_A) is linear. For a K-class classification task, the head takes the form $f_A(z) = zV + c$, with parameters $V \in \mathbb{R}^{d_A \times K}$ and $c \in \mathbb{R}^K$ learned on labeled training data using $g_A(x)$.

To relate the two representation spaces, we learn a linear alignment from the source to the target space. Given an input $x \in \mathcal{X}$, the aligned representation is $\hat{z}_A = g_B(x)W + b$, where $W \in \mathbb{R}^{d_B \times d_A}$ and $b \in \mathbb{R}^{d_A}$ are learned parameters. For all experiments, predictions are obtained by applying the fixed classifier $f_A(\cdot)$ to $\hat{z}_A$ without retraining.

In our experiments, g_A and g_B are instantiated using a set of pretrained language models drawn from both vendor-provided embedding services and locally hosted autoregressive models. For embedding models, we use OpenAI's text-embedding-3-small, Cohere's embedding API, Google's Gemini embedding-001 model, Qwen3-Embedding-8B, and e5-mistral-7b-instruct models. In all cases, the representation functions $g_A(\cdot)$ and $g_B(\cdot)$ are kept fixed, and only the affine alignment parameters (W, b) are learned.

Representational Similarity. Before evaluating behavioral transfer, we first verify that independently trained models exhibit shared linear structure. Figure 2 reports representative CKA similarity across model pairs; full CKA/SVCCA methodology and results are deferred to the Appendix.

Party A (Target)	Party B (Source)	Classification Acc. Baseline	Classification Acc. Lin.Map	OOD Dataset	OOD AUROC Baseline	OOD AUROC Lin.Map
SST-2 (50%)						
Gemini	OpenAI	94.5	93.1	AGNews	0.826	0.801
Cohere	Gemini	94.4	91.7		0.819	0.870
OpenAI	Cohere	93.0	92.4		0.875	0.843
Mistral	Qwen	94.5	93.7		0.859	0.826
TREC (17%)						
OpenAI	Qwen	96.0	95.6	AGNews	0.738	0.916
Cohere	Gemini	97.0	91.2		0.921	0.766
OpenAI	Cohere	96.4	94.4		0.954	0.802
Mistral	Qwen	97.0	96.6		0.921	0.940
AG News (25%)						
Gemini	OpenAI	92.6	91.6	MNLI	0.908	0.891
OpenAI	Qwen	92.6	91.9		0.953	0.913
Cohere	Gemini	91.9	91.5		0.875	0.885
Mistral	Qwen	92.6	92.4		0.953	0.940

Table 1. **Downstream performance is preserved under task supervised linear alignment:** We train a linear classifier $f_A(\cdot)$ on target embeddings $g_A(\cdot)$ (**Baseline**), then fit a linear map from source embeddings $g_B(\cdot)$ into the target space and evaluate using the same frozen head (**Lin.Map**).

3.1. Cross-Model Alignment to Downstream Tasks

We next test whether a simple linear transformation between embedding models preserves downstream performance on supervised classification and OOD detection. Since embeddings are commonly used as features for linear classifiers (Devlin et al., 2018; Tunstall et al., 2022), this provides a natural testbed for cross-model representation compatibility.

Experimental setup. For each dataset, we designate a target model (PARTY A) and a source model (PARTY B). We train the target linear classifier $f_A(\cdot)$ on the training split using target embeddings , and learn an affine map (W^*, b^*) on the same split to project source embeddings $g_B(x)$ into the target feature space. At test time, we freeze $f_A(\cdot)$ and evaluate predictions using aligned source representations:

$$\hat{y} \;=\; f_A(g_B(x)\,W^* + b^*)\,.$$

OOD detection. OOD detection evaluates whether a model can separate in-distribution inputs from unseen data by probing its logits *confidence*. We use the *Energy score* (Liu et al., 2021) for logits $f_A(z) \in \mathbb{R}^K$:

$$\mathcal{E}(z) \;=\; -\log \sum_{k=1}^{K} \exp\big(f_A(z)_k\big).$$

Model 1	Model 2	Baseline Acc.		Mapped Acc.	
		M1	M2	M1→M2	M2→M1
Llama3-8B	Qwen2.5-7B	58	70	48	68
Gemma3-270M	Llama3-8B	22	58	20	49
Gemma3-270M	Qwen2.5-7B	22	70	21	68
Llama3.2-1B	Llama3-8B	42	58	36	58
Llama3.2-1B	Qwen2.5-7B	42	70	28	69

Table 2. **Cross-model linear alignment on MMLU:** Baseline shows native model accuracy; Mapped shows accuracy after linearly transforming Model 1's representations to Model 2's head (M1→M2), or vice versa. Mapping from stronger to weaker models preserves performance, while mapping from weaker to stronger models degrades substantially.

Lower Energy indicates higher confidence, while higher values are characteristic of OOD inputs. We report AUROC by thresholding $\mathcal{E}(z)$ to distinguish in- vs. out-of-distribution samples. Since Energy depends on the full logit distribution, it provides a sensitive test of whether linear alignment preserves the target model's confidence structure.

Results. We report in-distribution classification accuracy and OOD AUROC, where OOD samples come from an alternative dataset. **Baseline** trains and evaluates a linear classifier $f_A(\cdot)$ on target embeddings $g_A(\cdot)$, while **Lin.Map** applies the same classifier to linearly aligned source embeddings. Table 1 shows that linear alignment largely preserves classification accuracy and achieves competitive OOD detection performance, with AUROC often matching or exceeding the baseline, indicating that the mapping recovers both decision boundaries and confidence structure.

3.2. Text Generation

We next evaluate whether linear alignment extends to the more demanding setting of autoregressive text generation. Unlike classification, which requires a single forward pass, generation demands coherent sequential predictions over many decoding steps where errors can accumulate, making generation a stringent test of whether linear maps preserve the structure necessary for multi-step inference.

Experimental Setup. We evaluate cross-model generation using open-source instruction-tuned models ranging from 270M to 32B parameters. For each model pair, we learn an affine transformation that maps source model B's penultimate layer representations to target model A's prediction head space. The transformation is learned via ordinary least squares with ridge regularization ($\lambda = 10^{-4}$) on 4,000 training examples from either MMLU or Alpaca. We extract hidden states using token-level alignment based on character offsets, matching each token from model A to the corresponding token in model B whose ending position is equal to or immediately follows A's ending position.

Crucially, we use model A's prediction head without fine-tuning. At each decoding step, we extract model B's penultimate hidden state, apply the learned affine transformation to map into model A's representation space, and pass the result to model A's frozen LM head $f_A(\cdot)$ for next-token prediction. This tests whether linear mapping alone can enable zero-shot cross-model generation. We use greedy decoding for a maximum of 128 tokens.

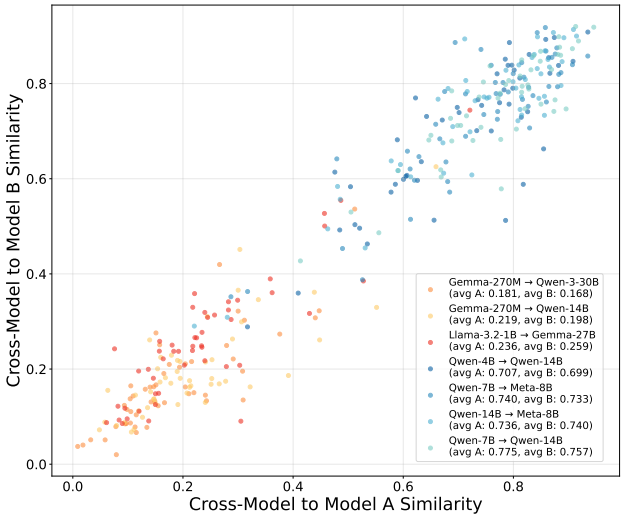

Figure 3. **Cross-Model Embedding Similarity to Native Models:** We compare Cross-Model text generation to the text produced by each base model using cosine similarity (using OpenAI's embedding-001). Each point represents an Alpaca test prompt. High-similarity pairs (upper right) produce coherent text, while low-similarity pairs (lower left) generate incoherent outputs.

3.2.1. MMLU: Structured Output Generation

Task and Evaluation. We evaluate accuracy on 100 MMLU test questions, using greedy decoding with early stopping when a valid answer letter (A, B, C, or D) is generated. For comparison, we report baseline accuracy for each model's native performance on the same questions.

Results. Table 2 shows results for five model pairs. Mapping from stronger to weaker models preserves substantial accuracy, while mapping from weaker to stronger models degrades significantly. This asymmetric transfer demonstrates that strong representations retain semantic structure after linear transformation and can be decoded zero-shot by weaker models' prediction heads without fine-tuning. The asymmetry confirms that source representational capacity, not the target head, is a limiting factor.

3.2.2. Alpaca: Instruction Following

Task and Evaluation. We evaluate cross-model generation on 100 held-out Alpaca test prompts using three complementary methods: (1) embedding similarity to native model outputs using OpenAI's text-embedding-3-small model; (2) LLM-as-a-Judge evaluation using GPT-5.1 and Claude-Sonnet-4, scoring relevance, quality, and helpfulness on a 1-10 scale; and (3) human evaluation on 200 samples (evenly split between single-model and cross-model) using a 1-5 scale based on proper English, quality, and coherence.

Results. We find that certain model pairs align substantially better than others, with two consistent trends. First,

Olmo - 7B → Qwen 2.5 - 14B
What are the benefits of social media for businesses? **The key benefits of using social media for businesses include increased brand awareness, improved customer engagement, enhanced advertising opportunities, better customer service, and cost-effective marketing...**
Evaluate the following expression: (6+2)*(2-2). **The result of the given expression is 0. First we calculate the addition and the subtraction, which are 8 and 0, and then the 8 is multiplied by 0...**
Describe how the given product works in 1-2 sentences: A portable charger. **A portable charger is a small, portable device with a high-capacity battery that recharges phones, tablets, or other electronics on the go. It has a set number of charge cycles and can recharge devices that...**
Llama 3 - 8B → Qwen 2.5 - 14B
Provide advice to a person who is considering becoming an entrepreneur. **First, make sure you have a solid business idea. This is the foundation of your business, and it's crucial that you have a clear and well-defined idea of what you want to do. Do your research, identify your target market, and make sure you have a unique value proposition that sets your business apart from...**
Writing a short article about a new technology that you think will revolutionize the world. **One new technology that has the potential to revolutionize the world is the development of a fully autonomous transportation system. This technology has been in the making for several years and is now on the verge of being fully implemented. The concept of a fully autonomous transportation system is one where vehicles are able to navigate and operate without the need for human input. This could be in the form of autonomous cars, drones...**

Table 3. Chat prompts and their corresponding completions for a given model pair using cross-model linear alignment.

smaller models ($\leq$ 1B parameters) yield poor text generation performance when mapped to larger models, even when both models belong to the same family (Figure 3). Second, *tokenizer compatibility* is a predictor of cross-model generation success. Analyzing 23 larger pairs ($\geq$4B parameters), we find that exact token match rate (i.e. the fraction of tokens aligning at corresponding positions when tokenizing identical text) strongly correlates with generation quality ($r = 0.898$, $p < 0.001$; Figure 4). High-quality pairs (LLM-judge score $\geq$3.5) consistently exhibit exact match $\geq$0.67, while failures show $\leq$0.24. Additionally, we analyze vocabulary overlap via Jaccard index, which shows similar predictive power ($r = 0.822$, $p < 0.001$; Appendix C). For example, Gemma's poor cross-family performance stems from low tokenizer compatibility (exact match <0.23, Jaccard <0.07). We find that pairs with exact match >0.7 succeed consistently, as evidenced by the clustering patterns in Figure 3. Full results are in Appendix C.

Embedding similarity. Figure 3 reveals a strong correlation between embedding similarity and generation quality. Model pairs with consistent high quality text generation cluster into either high-similarity (upper right, cosine similarity > 0.7 to both models), and model pairs with poor text generatino cluster into low-similarity (lower left, < 0.3) regions. High-similarity pairs like Qwen-7B$\leftrightarrow$Meta-8B and Qwen-14B$\rightarrow$Meta-8B produce outputs semantically similar to both native models, while low-similarity pairs like Gemma-270M$\rightarrow$Qwen-14B generate poor quality text.

LLM-as-a-Judge scores. Cross-model generation via linear alignment achieves LLM-judge scores of 4.0-4.7 for high-compatibility pairs (Qwen $\rightarrow$ Llama, Mistral-Nemo $\rightarrow$ Apertus), retaining 60-70% of single-model baseline. Low-compatibility pairs such as Gemma and small models (<2B) produce incoherent text (scores < 2.0), validating our embedding similarity analysis. Detailed scores and comparisons to single-model baselines are provided in the Appendix.

Human evaluation. Human evaluation scores corroborate the LLM-as-a-Judge findings. Cross-model pairs judged as low-quality by the LLM received an average human score of 1.1 ($\pm$0.2), while high-quality cross-model pairs scored 3.0 ($\pm$0.7). For comparison, low-quality baseline models scored 2.5 ($\pm$0.8) and high-quality baseline models scored 4.5 ($\pm$0.3). Human evaluators ranked outputs in the same order as the automated judge, validating that embedding similarity and LLM-as-a-Judge scores as reliable quality metrics.

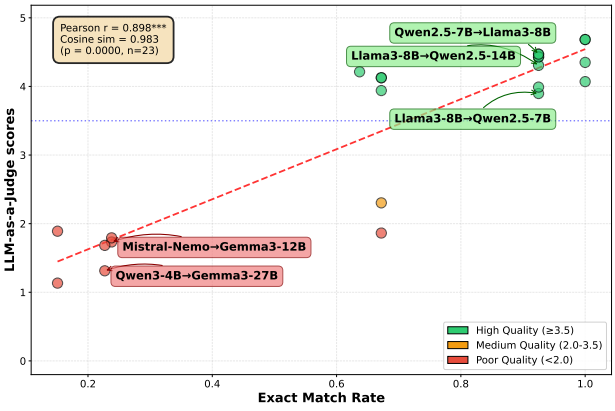

Figure 4. **Exact Token Match Rate Predicts Cross-Model Generation Quality.** Exact token match rate between two models predicts cross-model text generation quality across 23 model pairs. Quality is measured by LLM-as-a-Judge Scores ($r = 0.898$, $p < 0.001$).

Perplexity Analysis. Finally, we observe that larger model pairs (7–8B parameters) show lower perplexity degradation ($26.07\% \pm 12.26\%$) compared to smaller-to-larger mappings ($37.25\% \pm 25.84\%$), though small sample sizes ($N = 3$–4) limit statistical conclusions.

4. Privacy Framework

In this section we formalize the privacy-preserving execution model underlying HELIX, including the cryptographic assumptions, threat model, and protocol design. An extended definiton of the framework is in the Appendix.

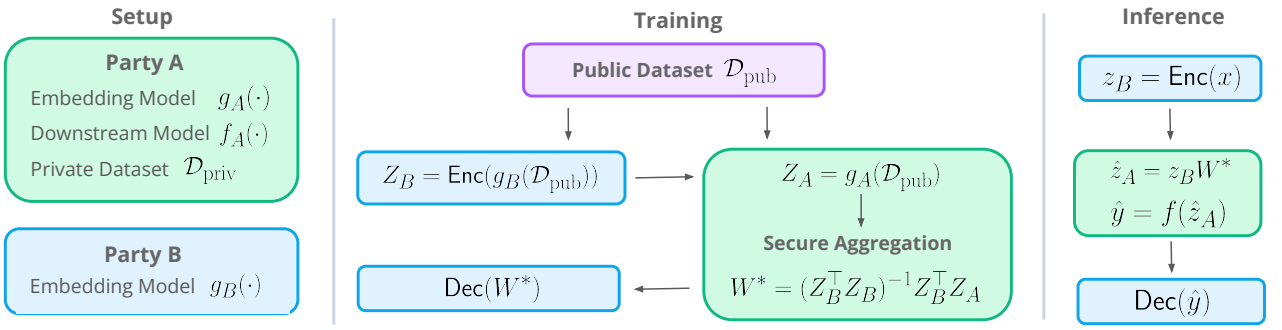

Figure 5. **Two-party privacy-preserving alignment and inference. Training:** PARTY B (client) encrypts embeddings $Z_B = g_B(\mathcal{D}_{\text{pub}})$ and sends $\text{Enc}(Z_B)$ to PARTY A (provider), who computes the encrypted cross-covariance $\text{Enc}(Z_A^\top Z_B)$ using plaintext $Z_A = g_A(\mathcal{D}_{\text{pub}})$ and returns $\text{Enc}(Z_A^\top Z_B)$ to PARTY B. PARTY B decrypts $Z_A^\top Z_B$ and computes W^* locally using Eq. (1). **Inference:** PARTY B computes aligned embedding $\hat{z}_A = z_B \cdot W^* + b^*$ locally, encrypts $\text{Enc}_I(\hat{z}_A)$, and sends to PARTY A, who applies the classifier homomorphically and returns the encrypted prediction for PARTY B to decrypt.

4.1. Problem Formulation and Entities

We consider a cross-silo inference scenario involving two parties: PARTY A (service provider), who owns a classification model, and PARTY B (client), who owns an independent embedding model. PARTY A provides classification-as-a-service (e.g., an API) by returning predictions without sharing model parameters. Due to privacy, regulatory, or competitive constraints, PARTY B cannot share raw data, query embeddings, or encoder parameters with PARTY A. Our goal is to learn an affine map that aligns PARTY B's representations to PARTY A's feature space using encrypted communication, enabling secure inference while protecting client confidentiality.

PARTY A (Service Provider). PARTY A holds a classification head $f_A : \mathbb{R}^{d_A} \to \mathcal{Y}$ trained on embeddings from their proprietary encoder $g_A : \mathcal{X} \to \mathbb{R}^{d_A}$ using private data $\mathcal{D}_{\text{priv}}$. PARTY A provides classification-as-a-service (e.g., an API) and does not share the classifier parameters (V, c) in plaintext. Instead, PARTY A evaluates f_A server-side on encrypted inputs, enabling PARTY B to obtain predictions without revealing queries while PARTY A retains model ownership. In our setting, f_A is a linear classifier.

PARTY B (Client). PARTY B holds a proprietary embedding model $g_B : \mathcal{X} \to \mathbb{R}^{d_B}$. PARTY B seeks to leverage PARTY A's classifier f_A for downstream predictions without revealing query data, embeddings $g_B(x)$, or encoder parameters to PARTY A.

Encrypted Computation. Our protocol uses homomorphic encryption (CKKS (Cheon et al., 2017)) to enable computation on encrypted data without revealing inputs. We restrict secure computation to linear operations, as approximate HE schemes evaluate these efficiently, following prior work that targets linear components for encrypted training and inference (Gilad-Bachrach et al., 2016a; Mohassel & Zhang, 2017; Juvekar et al., 2018; Lee et al., 2023). Our implementation returns full logits, however, prior work has shown that encrypted argmax operations (Jovanovic et al., 2022; Zhang et al., 2024a) can limit information leakage and defend against model extraction and membership inference attacks (Tramèr et al., 2016; Carlini et al., 2021). These techniques could be straightforwardly integrated with our linear alignment protocol for enhanced model provider privacy in production deployments.

4.2. Threat Model and Guarantees

We adopt a *semi-honest* (honest-but-curious) threat model (Goldreich, 2004) where both parties follow the protocol but may analyze communication to infer private information; we do not consider malicious adversaries. We prioritize client-side privacy with the following security guarantees:

- **Client input privacy (PARTY B).** During inference, PARTY A observes only CKKS ciphertexts (encrypted aligned embeddings and encrypted outputs). Without PARTY B's secret key, semantic security implies these ciphertexts reveal no information about queries x, embeddings $g_B(x)$, or encoder parameters beyond what is implied by the decrypted final predictions.
- **Provider classifier privacy (PARTY A).** PARTY A's classifier parameters (V, c) are never revealed in plaintext to PARTY B and are only used within encrypted linear inference. The base protocol returns encrypted prediction outputs to PARTY B. We treat adaptive attacks such as model extraction and membership inference as *out of scope* for this work.

Alignment map leakage. PARTY B retains the learned alignment map W^* in plaintext after decryption and uses it locally during inference. While W^* reveals structural information about PARTY A's embedding space (dimension d_A), it does not expose the classifier parameters (V, c) or training data $\mathcal{D}_{\text{priv}}$. However, W^* may enable property inference about PARTY A's model architecture. We evaluate membership inference attacks on W^* in Appendix J, demonstrating that W^* does not leak individual sample membership under standard geometric feature extraction.

Limitations. As in most HE deployments, structural information (embedding dimensions, communication volume)

may be revealed. Our semi-honest threat model assumes PARTY B makes honest inference queries rather than attempting model extraction through adaptive attacks.

4.3. Secure Linear Alignment

Given public data $\mathcal{D}_{\text{pub}}$ with embeddings $Z_A = g_A(\mathcal{D}_{\text{pub}})$ and $Z_B = g_B(\mathcal{D}_{\text{pub}})$, we compute the affine alignment from the sufficient statistics $Z_B^\top Z_B$ and $Z_B^\top Z_A$:

$$W^* = (Z_B^\top Z_B + \lambda I)^{-1} Z_B^\top Z_A. \tag{1}$$

To avoid revealing raw embeddings, we stream aggregates over mini-batches; the cross-covariance $Z_B^\top Z_A$ is computed under homomorphic encryption via secure aggregation.

Public data assumption. We assume a shared, non-sensitive public dataset $\mathcal{D}_{\text{pub}}$ for fitting W^*. When learned exclusively from $\mathcal{D}_{\text{pub}}$, the alignment reflects only public distributional structure and PARTY A's private data remains confidential. Optionally, PARTY A may augment $\mathcal{D}_{\text{pub}}$ with a small number of plaintext in-distribution samples from $\mathcal{D}_{\text{priv}}$ (64-128 samples) to improve alignment accuracy; this trades data confidentiality for improved performance, revealing limited task-specific information to PARTY B. While the shared samples are directly exposed, a secondary concern is whether W^* itself leaks information about which specific samples were included—for instance, if PARTY A curates the shared set to exclude sensitive examples, can an adversary infer their presence or absence from W^*? We evaluate this through membership inference analysis in Appendix J, demonstrating that W^* leakage is provably bounded: membership inference advantage is $O(\sqrt{d}/N) \approx 0.016$ for our configuration, yielding negligible privacy risk.

4.4. Two-Party Secure Training Protocol

To compute Eq. (1) while preserving client confidentiality:

1. **Encrypt and send.** PARTY B (client) generates keys (pk, sk), encrypts Z_B to obtain $\mathsf{Enc}_{pk}(Z_B)$, and sends it to PARTY A.
2. **Secure aggregation.** PARTY A (provider) computes $\mathsf{Enc}(Z_A^\top Z_B)$ via homomorphic matrix multiplication using plaintext Z_A and encrypted Z_B, then returns $\mathsf{Enc}(Z_A^\top Z_B)$ to PARTY B.
3. **Decrypt and solve.** PARTY B decrypts to obtain $Z_A^\top Z_B$, then computes $W^* = (Z_B^\top Z_B + \lambda I)^{-1} Z_A^\top Z_B$ in plaintext using their local Z_B. Optionally, PARTY B computes bias $b^* \leftarrow \frac{1}{N}\mathbf{1}^\top (Z_A - Z_B W^*)$ after obtaining Z_A via an additional encrypted transmission or using public data statistics.

4.5. Privacy-Preserving Inference

Inference uses the plaintext map W^* held by PARTY B (client). To preserve query confidentiality, PARTY B generates a fresh inference keypair (pk_I, sk_I) and shares pk_I with PARTY A. Assuming a linear classifier $f_A(z) = zV + c$, the protocol is as follows:

1. **Compute and encrypt aligned representation.** PARTY B computes $z_B = g_B(x)$, applies the alignment locally $\hat{z}_A = z_B \cdot W^* + b^*$, and sends $\mathsf{Enc}_{pk_I}(\hat{z}_A)$ to PARTY A.
2. **Homomorphic classification.** PARTY A computes $\mathsf{Enc}(\hat{y}) = \mathsf{Enc}(\hat{z}_A) \cdot V + c$ homomorphically.
3. **Decrypt output.** PARTY B decrypts $y = \mathsf{Dec}_{sk_I}(\mathsf{Enc}(\hat{y}))$.

5. Experiments

We apply the HELIX framework to large-scale embedding models to evaluate its effectiveness on downstream tasks. Embeddings are commonly used directly as features to train a linear classifier on labeled data (Devlin et al., 2018; Wang et al., 2022). Our experiments utilize five embedding models: EMBEDDING-001 (Google), TEXT-EMBEDDING-3-SMALL (OpenAI), E5-MISTRAL-7B-INSTRUCT, QWEN3-EMBEDDING-8B, and EMBED-ENGLISH-V3.0 (Cohere).

5.1. Embedding Classification

We investigate the cross-silo knowledge transfer capabilities of HELIX between two private embedding models. In contrast to Section 3, we concentrate on training a linear map on a public dataset, rather than the in-distribution dataset.

Experimental Setup. We choose an embedding model to represent PARTY A and train a linear classifier $f(\cdot)$ on the full training set using embeddings $Z_A = g_A(X)$ and labels Y. We evaluate across six datasets: TREC, MNLI, DBpedia, and AG News. We use embedding model $g_B(\cdot)$ to serve as PARTY B.

We then train the linear map W^* under two settings: (1) using only a public, *independent* dataset to estimate W^*, and (2) augmenting the public dataset with a small number of in-distribution samples (64, 128) in the dataset under test. For the public dataset, we use Wikipedia and IMDB. Our baseline are a linear classifier trained on the full training set ($f(\cdot)$), as well as a classifier trained on the few-shot in-distribution samples (in practice the client would have access to the few-shot samples to build their own classifier). Setting (2) serves as an upper-bound analysis demonstrating the performance ceiling when W^* uses limited in-distribution data (64-128 samples). However, this compromises data security by requiring PARTY A to share potentially sensitive samples with PARTY B during training, violating the zero-shot privacy guarantees of Setting (1). We argue that this performance gap quantifies the privacy-utility trade-off: how much accuracy is sacrificed to preserve full data confidentiality versus accepting limited data exposure for improved alignment.

Results. Table 4 shows that cross-model alignment achieves strong performance across model pairs and datasets. **Public + ID** (64-128 samples) consistently matches or exceeds baselines trained on the same few-shot data, with particularly strong results on TREC and MNLI. Critically, **Public Only** surpasses 64-shot baselines in many configurations.

Party A (Target)	Party B (Source)	Baseline Full Data (%)	Baseline 64	Baseline 128	Public Only	Public + ID 64	Public + ID 128	Baseline Full Data(%)	Baseline 64	Baseline 128	Public Only	Public + ID 64	Public + ID 128
TREC (17%)								**MNLI** (33%)					
Gemini	OpenAI	95.4	56.6	77.6	57.6	78.0	**81.0**	65.0	33.5	35.1	44.5	48.5	**48.9**
OpenAI	Qwen	96.4	59.0	82.8	58.4	77.6	**83.6**	62.4	57.7	61.9	64.0	64.9	**65.2**
Mistral	Cohere	96.6	52.0	73.2	55.8	74.4	**78.4**	77.6	34.4	35.8	39.5	41.9	**42.8**
OpenAI	Cohere	96.4	52.2	73.4	68.4	80.2	**80.6**	62.4	34.4	35.8	46.5	47.1	**47.9**
Mistral	Qwen	96.6	59.4	82.2	65.6	83.8	**87.4**	77.6	57.6	61.8	58.8	71.3	**72.9**
Qwen	OpenAI	97.0	57.0	77.8	75.4	82.8	**87.6**	87.9	33.4	35.2	42.0	42.9	**43.9**
DBpedia (7%)								**AG News** (25%)					
Gemini	OpenAI	99.1	65.0	77.8	53.5	89.3	**94.1**	92.1	79.6	87.0	85.9	88.4	**88.7**
OpenAI	Qwen	99.1	66.0	83.4	66.7	91.0	**95.8**	92.7	80.0	87.1	79.7	88.6	**89.2**
Mistral	Cohere	99.2	56.5	78.9	44.5	70.0	**86.7**	93.4	81.8	**85.0**	64.3	80.0	83.6
Gemini	Mistral	99.1	52.0	79.1	65.7	93.4	**96.6**	92.1	75.7	87.8	86.7	88.1	**88.7**
Mistral	Qwen	99.2	72.9	93.2	79.6	93.6	**96.5**	93.4	84.4	86.5	71.4	87.9	**88.2**
Qwen	OpenAI	99.1	74.1	88.9	66.4	81.3	**89.9**	92.9	82.0	86.1	84.6	86.2	**87.2**

Table 4. **Cross-model alignment classification.** A classifier $f(\cdot)$ is trained on private data (column *Baseline Full Data (%)*). We evaluate HELIXs by mapping representations from a public dataset, and from the public dataset augmented with few-shot in-distribution samples (64, 128), prior to classification by $f(\cdot)$. Additional baselines correspond to independently trained classifiers using *only* the few-shot samples. Best results are shown in **bold**, second-best underlined. Each experiment is repeated three times with different random seeds. The standard deviation across runs is ± 1.5 percentage points, indicating high reproducibility. Full results are in the Appendix.

Method	Security Scope	SST-2	MRPC	RTE	Latency (s)
Baseline	None	92.3	90.3	69.7	< 1
BOLT	Full (HE+MPC)	92.8	90.0	69.3	> 60
Nimbus	Full (HE+MPC)	92.6	89.8	66.8	> 20
MPCFormer	Full (MPC)	–	88.7	64.9	18
Enc.-Friendly	Full (HE)	81.9	81.5	59.3	26.5
PowerFormer	Full (HE)	92.0	87.8	69.8	> 20
Nexus	Full (HE)	92.1	–	69.9	37.3
HELIX* (Base)	Linear (HE)	92.3	77.8	59.6	< 1
HELIX (FT)	Linear (HE)	93.0	82.0	55.6	< 1

Table 5. Efficiency–utility comparison across *distinct secure inference settings* on GLUE tasks. **Security Scope** denotes which model components are evaluated under cryptographic protection. Prior methods secure full transformer inference, whereas HELIX encrypts only linear alignment and classification.

5.2. Computational Efficiency Analysis

We contextualize HELIX alongside prior private inference systems on GLUE tasks in a two-party setting, while emphasizing that these approaches target **different secure-inference scopes**. Prior systems protect end-to-end transformer inference under cryptographic protocols, whereas HELIX assumes the client can compute embeddings locally and requires secure computation only for the final linear classification head. We report task accuracy and end-to-end latency to characterize the tradeoff within each scope.

In addition, the evaluated model families differ across settings: prior work reports secure inference results for a single BERT model, while HELIX is assessed using randomly paired embedding models from Gemini, OpenAI, and Cohere, as well as a fine-tuned Llama-2-8b mapped to base Llama-2-8b. These experiments reflect HELIX's design goal of enabling cross-silo transfer across independent models with minimal encrypted computation.

Experimental Setup. We evaluate HELIX on SST-2, MRPC, and RTE. Reported latency includes plaintext embedding inference, CKKS encryption, encrypted evaluation of the linear classification head, and decryption. Baselines perform cryptographically protected end-to-end inference on BERT, as reported in their original works. Our results are averaged over three model pairs per task, with a low variance across runs. We implement HELIX using TenSEAL CKKS with poly_modulus_degree=8192, coefficient modulus [60, 40, 40, 60], and scale 2^{40}. These parameters achieve 128-bit security according to the Homomorphic Encryption Security Standard (HomomorphicEncryption.org, 2018). Our circuit has minimal multiplicative depth (depth-1: one ciphertext-plaintext multiplication), well within CKKS's noise budget and requiring no bootstrapping or modulus switching beyond standard rescaling. All measurements are run on CPU and reported as end-to-end latency.

Results. Across tasks, HELIX achieves strong accuracy with sub-second latency by encrypting only the final linear head. In contrast, prior systems incur substantially higher cryptographic overhead because they secure end-to-end transformer inference (often requiring online interaction), leading to multi-second to minute-scale latency.

In addition to low-latency inference, HELIX incurs less than 1MB of communication per sample, since it transmits only a single embedding vector (typically 1536–3072 dimensions) since CKKS uses SIMD packing to encrypt multiple values into a single ciphertext. With poly_modulus_degree=8192, each ciphertext can encode up to 4096 values.

6. Conclusion

We introduce a framework for linearly and securely transferring knowledge between LLMs. Several areas remain for future work. First, extending to multi-modal models and developing tokenizer-agnostic alignment methods could broaden applicability beyond text-only models with compatible vocabularies. Second, while our protocol protects client queries, better securing the provider's model parameters remains an open challenge. Finally, exploring efficient non-linear alignment methods could improve cross-model generation quality while maintaining computational efficiency.

Impact Statement

This work advances the practical understanding of representation alignment and secure computation for machine learning models. The proposed techniques enable cross-model knowledge transfer and privacy-preserving training and inference in settings where data or model sharing is restricted. We do not anticipate new societal risks beyond those commonly associated with the deployment of large language models; however, as with any enabling technology, these methods could be misused depending on the application context. We therefore encourage responsible deployment consistent with established privacy, security, and ethical guidelines.

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Bansal, Y., Nakkiran, P., and Barak, B. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeurIPS 34)*, pp. 225–236, 2021.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. Leveled fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3), 2014.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Carlini, N., Paleka, D., Dj Dvijotham, K., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., Yona, I., Wallace, E., Rolnick, D., and Tramèr, F. Stealing part of a production language model, 2024. URL https://arxiv.org/abs/2403.06634.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

Chen, A., Merullo, J., Stolfo, A., and Pavlick, E. Transferring features across language models with model stitching. *arXiv preprint arXiv:2506.06609*, 2025.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pp. 409–437. Springer, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

Diebold, G. Overcoming barriers to data sharing in the united states. Technical report, Center for Data Innovation, 2023. URL https://www2.datainnovation.org/2023-data-sharing-barriers.pdf. Report on legal, social, technical and economic obstacles to data sharing.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

et al., K. B. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS*, 2017.

et al., P. R. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017*, 2018.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born-again neural networks. *arXiv:1805.04770*, 2018.

Gentry, C. *A Fully Homomorphic Encryption Scheme*. Stanford University, 2009.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 201–210, 2016a.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016b.

Goldreich, O. *Foundations of Cryptography: Volume 2*. Cambridge University Press, 2004.

Gostin, L. O. and Hodge, J. G. Personal privacy and the health information portability and accountability act (hipaa): A comment on the supreme court's decision in Ferguson v. City of Charleston. *Journal of Law, Medicine & Ethics*, 28(2):210–213, 2000.

Hao, M., Li, H., Chen, H., Xing, P., Xu, G., and Zhang, T. Iron: Private inference on transformers. *Advances in neural information processing systems*, 35:15718–15731, 2022.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

HomomorphicEncryption.org. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, November 2018. URL https://homomorphicencryption.org/standard/. Technical Report.

Huh, M., Cheung, B., Wang, T., and Isola, P. Position: The platonic representation hypothesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20617–20642. PMLR, 2024. URL https://arxiv.org/abs/2405.07987.

Jiang, F. and Li, J. Stitching gpt to bert for look-ahead language understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Jovanovic, N., Fischer, M., Steffen, S., and Vechev, M. Private and reliable neural network inference. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1663–1677, 2022.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. Gazelle: A low latency framework for secure neural network inference. In *USENIX Security*, 2018.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., and et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chowdhery, A., Chen, R., Elhage, N., Ebrahimi, R., Musabi, R., Khan, G., et al. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*, 2020.

Karimireddy, S. P., Kale, S., Mohri, M., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 5132–5143, 2020.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In *EMNLP*, pp. 1317–1327, 2016.

Klabunde, M., Schumacher, T., Strohmaier, M., and Lemmerich, F. Similarity of neural network models: A survey of functional and representational measures. *ACM Computing Surveys*, 57(9):1–52, 2025.

Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., and van der Maaten, L. Crypten: Secure multi-party computation meets machine learning. In *NeurIPS*, 2021. URL https://papers.neurips.cc/paper/2021/file/2754518221cfbc8d25c13a06a4cb8421-Paper.pdf.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Lee, S., Lee, G., Kim, J. W., Shin, J., and Lee, M.-K. Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption. In *International conference on machine learning*, pp. 19010–19035. PMLR, 2023.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Li, D., Wang, H., Shao, R., Guo, H., Xing, E., and Zhang, H. MPCFORMER: FAST, PERFORMANT AND PRIVATE TRANSFORMER INFERENCE WITH MPC. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CWmvjOEhgH-.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

Li, Z., Yang, K., Tan, J., Lu, W.-j., Wu, H., Wang, X., Yu, Y., Zhao, D., Zheng, Y., Guo, M., and Leng, J. Nimbus: Secure and efficient two-party inference for transformers. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 2024. URL https://arxiv.org/abs/2411.15707. arXiv:2411.15707.

Liu, W. et al. Energy-based out-of-distribution detection. In *ICLR*, 2021.

Liu, Y., Kang, Y., Xing, C., Chen, T., and Yang, Q. A secure federated transfer learning framework. *IEEE Intelligent*

Systems, 35(4):70–82, 2020. URL https://arxiv.org/pdf/1812.03337.

Liu, Y., Chen, T., and Yang, Q. Hierarchical federated learning: Algorithms, applications and challenges. *arXiv preprint arXiv:2109.11846*, 2022.

Luo, J., Zhang, Y., Zhang, Z., Zhang, J., Mu, X., Wang, H., Yu, Y., and Xu, Z. Secformer: Fast and accurate privacy-preserving inference for transformer models via smpc. *arXiv preprint arXiv:2401.00793*, 2024. URL https://arxiv.org/abs/2401.00793.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of Machine Learning Research (AISTATS)*, volume 54, pp. 1273–1282, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf.

McNeely-White, D., Beveridge, J. R., and Draper, B. A. Inception and resnet features are (almost) equivalent. *Cognitive Systems Research*, 59:312–318, 2020.

McNeely-White, D., Sattelberg, B., Blanchard, N., and Beveridge, R. Canonical face embeddings. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(2): 197–209, 2022.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, 2017. URL https://www.ieee-security.org/TC/SP2017/papers/466.pdf.

Mora, A., Tenison, I., Bellavista, P., and Rish, I. Knowledge distillation for federated learning: a practical guide. *arXiv preprint arXiv:2211.04742*, 2022.

Morcos, A. S., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

Pang, Q., Zhu, J., Möllering, H., Zheng, W., and Schneider, T. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4753–4771. IEEE, 2024.

Park, D., Lee, E., and Lee, J.-W. Powerformer: Efficient privacy-preserving transformer with batch rectifier-power max function and optimized homomorphic attention. Technical report, Cryptology ePrint Archive, 2024.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. Svcca: Singular vector canonical correlation analysis for deep understanding and improvement. *stat*, 1050:19, 2017.

Reimsbach-Kounatze, C., Ishikawa, S., et al. Sharing trustworthy ai models with privacy-enhancing technologies. Technical Report No. 38, OECD, 2025. URL https://www.oecd.org/content/dam/oecd/en/publications/reports/2025/06/sharing-trustworthy-ai-models-with-privacy-enhancing-technologies_5df6fd05/a266160b-en.pdf.

Roeder, G., Wu, Y., Duvenaud, D., and Grosse, R. On linear identifiability of learned representations. *arXiv preprint arXiv:2007.00810*, 2020.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022. URL https://arxiv.org/abs/2206.04615.

Tang, Y. and Yang, Y. Pooling and attention: What are effective designs for llm-based embedding models? *arXiv preprint arXiv:2409.02727*, 2024.

Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. In *ICLR*, 2020.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pp. 601–618, USA, 2016. USENIX Association. ISBN 9781931971324.

Tunstall, L., Reimers, N., Jo, U. E. S., Bates, L., Korat, D., Wasserblat, M., and Pereg, O. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*, 2022.

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. In *ICLR 2019 Workshop on AI for Social Good*, 2019. URL https://aiforsocialgood.github.io/iclr2019/accepted/track1/pdfs/31_aisg_iclr2019.pdf.

Voigt, P. and Von dem Bussche, A. The european general data protection regulation (gdpr). *A Practical Guide. Springer International Publishing*, pp. 10–15, 2017.

Wang, J., Chen, Q., Sun, H., Shi, Z., and Yang, E. a. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33:7611–7623, 2020.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Wu, F., Cui, L., Yao, S., and Yu, S. Inference attacks in machine learning as a service: a taxonomy, review, and promising directions. *arXiv e-prints*, pp. arXiv–2406, 2024.

Zhang, J., Yang, X., He, L., Chen, K., Lu, W.-j., Wang, Y., Hou, X., Liu, J., Ren, K., and Yang, X. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024a.

Zhang, L., Li, M. Y., and Griffiths, T. L. What should embeddings embed? autoregressive models represent latent generating distributions. *arXiv preprint arXiv:2406.03707*, 2024b.

Zhang, Y., Wang, H., et al. Encryption-friendly large language model architectures. In *International Conference on Learning Representations (ICLR)*, 2025.

Appendix

A. LLM Similarity

To assess the extent to which independently trained embedding models learn compatible linear structure, we analyze representational similarity using two complementary metrics: linear Centered Kernel Alignment (CKA) and Singular Vector Canonical Correlation Analysis (SVCCA). Both measures operate directly on representation matrices computed over a shared set of inputs and quantify the degree of shared linear geometry across models.

A.1. CKA Heatmaps

Our primary similarity analysis uses linear CKA (Kornblith et al., 2019), which measures similarity between two representation matrices $Z_A \in \mathbb{R}^{n \times d_1}$ and $Z_B \in \mathbb{R}^{n \times d_2}$ computed over the same set of n inputs. Linear CKA is invariant to isotropic scaling and orthogonal transformations, making it a stable and widely used metric for comparing internal representations across models with different dimensionalities or parameterizations.

Formally, linear CKA is defined as

$$\mathrm{CKA}(Z_A, Z_B) = \frac{\|Z_A^\top Z_B\|_F^2}{\|Z_A^\top Z_A\|_F \, \|Z_B^\top Z_B\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm. A CKA value of 1 indicates identical representations up to orthogonal transformation and scaling, while a value of 0 indicates no shared linear structure.

Representation extraction. We compute Z_A and Z_B using a mixture of deployed embedding APIs and locally run autoregressive LLMs. For embedding APIs, we use the provider's default embedding outputs, which already reflect model-specific pooling and normalization strategies. For locally run autoregressive models, which are trained primarily for next-token prediction and often instruction fine-tuned, we adopt mean pooling over the final hidden layer to obtain fixed-dimensional sequence representations.

Concretely, given final-layer token embeddings $\{h_i\}_{i=1}^{L}$ for an input sequence of length L, we compute the sequence-level embedding as

$$z = \frac{1}{L} \sum_{i=1}^{L} h_i.$$

Prior work has shown that pooling hidden states of autoregressive models yields semantic representations competitive with those of dedicated embedding models (Zhang et al., 2024b; Tang & Yang, 2024), making this a reasonable and widely adopted choice for cross-model comparison.

Experimental Procedure. We compute linear CKA between pairs of embedding models using representations extracted on shared training and test splits. Embeddings are mean-centered prior to CKA computation. For embedding APIs, we use the provider's default outputs, while for instruction-tuned autoregressive models we apply mean pooling over final-layer token embeddings.

CKA is computed on both splits following (Kornblith et al., 2019), using 5,000 training and 2,000 test samples. Each experiment is repeated three times with different random subsamples, and results are averaged.

Results. Figure 6 reports linear CKA similarity across instruction-tuned autoregressive LLMs, computed on Alpaca and TriviaQA inputs. Each heatmap reports average test-set CKA computed using 5,000 samples to estimate representation covariance, with evaluation performed on 2,000 held-out samples. Results are averaged over three random subsampling runs to reduce variance.

We observe moderate to high CKA similarity across most autoregressive model pairs, indicating substantial shared linear structure despite differences in architecture, pretraining data, and instruction-tuning procedures. While variability across pairs is more pronounced than in encoder-style embedding models, many pairs still exhibit CKA values above 0.5, which corresponds to meaningful shared linear structure and suggests strong potential for linear alignment. We hypothesize that the increased variance arises from task-specific shifts introduced during instruction tuning, which may alter representation geometry while preserving a common semantic backbone.

A.2. SVCCA

To complement CKA, we analyze representations using Singular Vector Canonical Correlation Analysis (SVCCA) (Raghu et al., 2017), which measures correlation between low-rank subspaces extracted via singular value decomposition followed by canonical correlation analysis. Unlike CKA, which evaluates global similarity between full representation matrices, SVCCA emphasizes shared informative subspaces, making it particularly relevant for assessing the feasibility of linear alignment.

SVCCA Experimental Procedure For each model pair, we extract embeddings on shared training and test splits. Encoder-style embedding models use provider-default outputs, while instruction-tuned autoregressive models use mean pooling over final-layer token embeddings.

SVCCA is fit on the training embeddings by first applying PCA to each model's representations, followed by canonical correlation analysis (CCA) on the reduced features. We evaluate SVCCA on both the training and test splits using the fitted PCA and CCA transforms. We report results using fixed PCA dimensionalities of 64 and 128 components.

To ensure comparability, embeddings are aligned by truncating to the minimum number of available samples across models, with 10,000 training examples and 2,000 test examples. Each experiment is repeated three times with different random subsamples, and reported correlations are averaged across runs. As a control, we compute a random baseline by shuffling one model's embeddings prior to SVCCA.

Figures 8 and 9 report SVCCA results for encoder-style embedding models evaluated across multiple datasets and model pairings, using projections onto the top 64 and 128 components, respectively. Across datasets and model combinations, we observe consistently high SVCCA correlations, indicating a strong shared low-rank subspace among embedding models.

Figures **??** and **??** show corresponding SVCCA curves for instruction-tuned autoregressive LLMs. Although these models exhibit greater variability across pairs, the mean and median SVCCA correlations remain above 0.6 for a majority of components, even at higher dimensionalities. This suggests that instruction tuning does not eliminate a substantial shared subspace, but instead introduces controlled variation atop a common representational core.

Summary. Taken together, the CKA and SVCCA analyses provide evidence that independently trained embedding models learn compatible representations. Despite differences in training objectives, architectures, and fine-tuning procedures, these models preserve shared geometric structure that is amenable to linear alignment, motivating our subsequent investigation into whether such alignment suffices for downstream behavioral transfer.

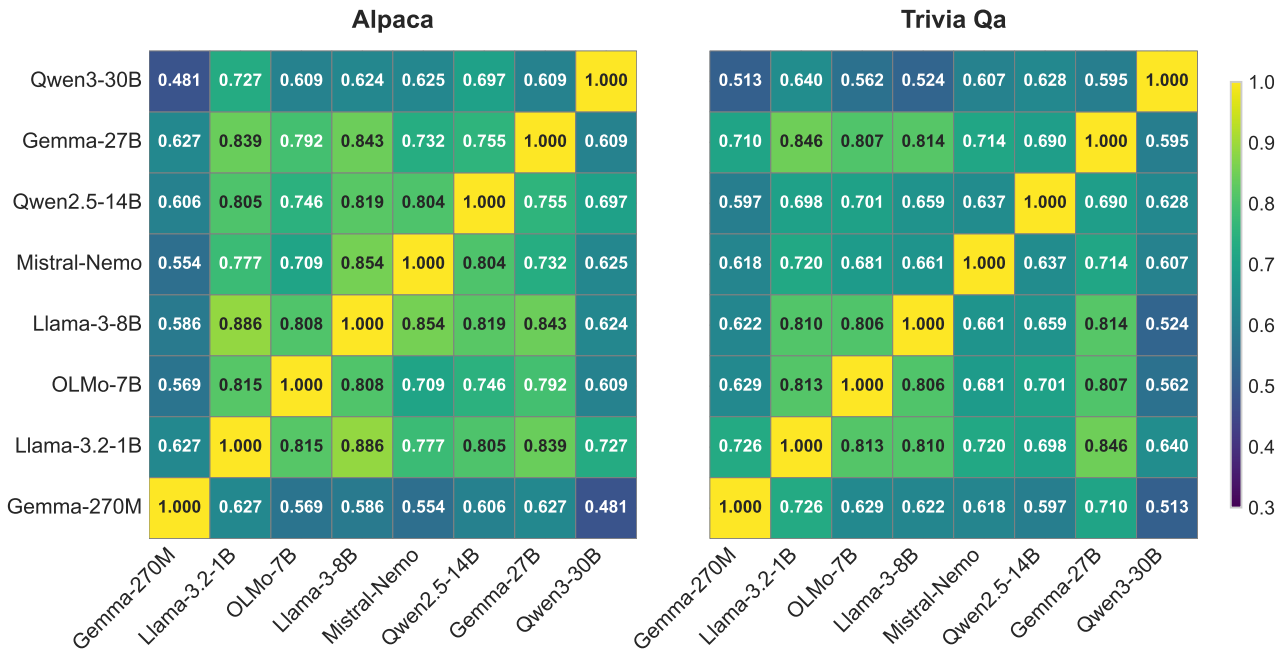

Figure 6. Test-set CKA between embeddings across Alpaca and Trivia QA datasets. Model embeddings were mean pooled at their final hidden state prior to calculating CKA with other models.

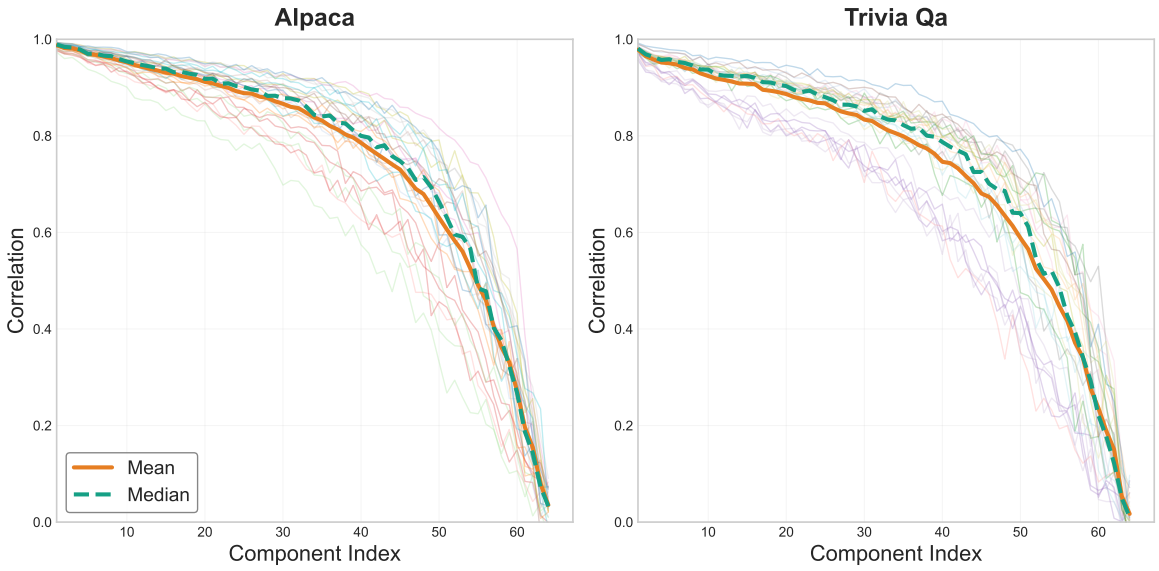

Figure 7. SVCCA of eight instruction tuned LLM combinations at 64 components. Mean and median are bolded.

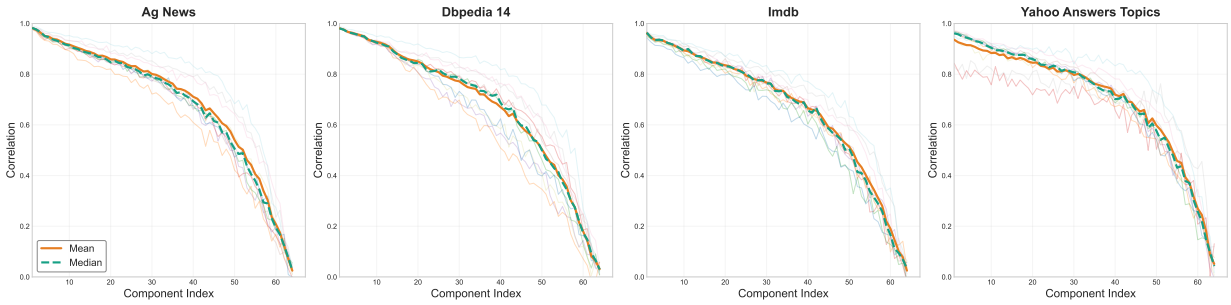

Figure 8. SVCCA of four datasets on five embedding model combinations.

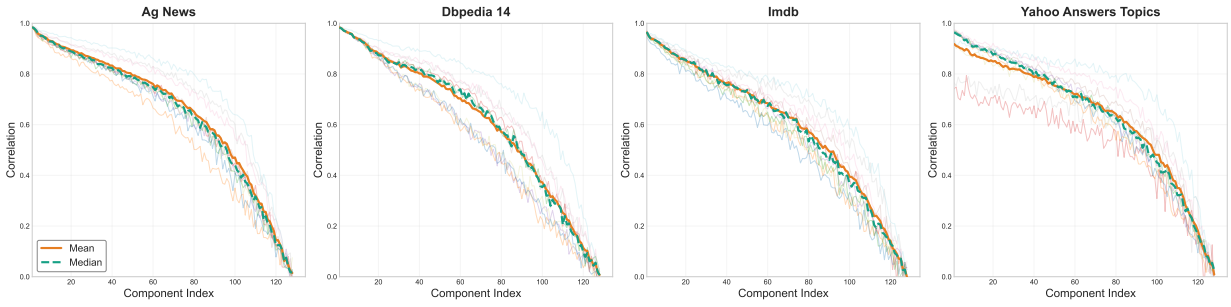

Figure 9. SVCCA of five embedding model combinations at 128 components (test set)

A.3. Task-Supervised Alignment Baseline

To characterize the best-case performance of linear alignment, we first evaluate an *in-distribution* setting where the mapping is trained directly on the task's training split. Concretely, we fit (W^*, b^*) using paired embeddings from the training set (computed by g_A and g_B on the same inputs) and evaluate on the held-out test split using the fixed classifier $f_A(\cdot)$ trained on target embeddings. This baseline does *not* use a separate public dataset $\mathcal{D}_{\text{pub}}$ for alignment; instead, it reflects a setting where the alignment data matches the downstream task distribution and provides a reference point for the public-data mapping results used in our privacy-preserving protocol.

Party A (Target)	Party B (Source)	Classification Acc. Baseline	Classification Acc. HELIX	OOD Dataset	OOD AUROC Baseline	OOD AUROC HELIX
AG News (25%)						
Gemini	OpenAI	92.6	91.6	MNLI	0.908	0.891
OpenAI	Qwen	92.6	91.9	MNLI	0.953	0.913
Cohere	Gemini	91.9	91.5	MNLI	0.875	0.885
OpenAI	Cohere	92.1	92.0	MNLI	0.881	0.880
Mistral	Qwen	92.6	92.4	MNLI	0.953	0.940
DBpedia (7%)						
Gemini	OpenAI	98.9	98.6	Yahoo	0.971	0.975
OpenAI	Qwen	99.0	98.7	Yahoo	0.979	0.974
Cohere	Gemini	98.9	98.7	Yahoo	0.973	0.980
OpenAI	Cohere	98.9	98.7	Yahoo	0.971	0.973
Mistral	Qwen	99.0	98.9	Yahoo	0.979	0.985
Yahoo Answers Topics (10%)						
Gemini	OpenAI	70.9	67.6	SST-2	0.417	0.612
OpenAI	Qwen	71.5	68.6	SST-2	0.623	0.491
Cohere	Gemini	70.2	67.3	SST-2	0.566	0.593
OpenAI	Cohere	69.7	69.4	SST-2	0.481	0.444
Mistral	Qwen	71.5	70.5	SST-2	0.623	0.482
MNLI (33%)						
Gemini	OpenAI	62.4	61.3	IMDB	0.367	0.092
OpenAI	Qwen	87.9	61.3	IMDB	0.905	0.693
Cohere	Gemini	65.0	56.8	IMDB	0.111	0.263
OpenAI	Cohere	59.1	58.3	IMDB	0.384	0.285
Mistral	Qwen	87.9	78.5	IMDB	0.905	0.650
SST-2 (50%)						
Gemini	OpenAI	94.5	93.1	AGNews	0.826	0.801
OpenAI	Qwen	94.5	93.8	AGNews	0.859	0.848
Cohere	Gemini	94.4	91.7	AGNews	0.819	0.870
OpenAI	Cohere	93.0	92.4	AGNews	0.875	0.843
Mistral	Qwen	94.5	93.7	AGNews	0.859	0.826
IMDB (50%)						
Gemini	OpenAI	94.9	95.9	Amazon	0.523	0.543
OpenAI	Qwen	95.7	94.8	Amazon	0.535	0.535
Cohere	Gemini	96.4	94.5	Amazon	0.572	0.501
OpenAI	Cohere	94.8	94.9	Amazon	0.379	0.534
Mistral	Qwen	95.7	95.1	Amazon	0.535	0.617
TREC (16.7%)						
Gemini	OpenAI	95.4	94.0	AGNews	0.412	0.718
OpenAI	Qwen	96.0	95.6	AGNews	0.738	0.916
Cohere	Gemini	97.0	91.2	AGNews	0.921	0.766
OpenAI	Cohere	96.4	94.4	AGNews	0.954	0.802
Mistral	Qwen	97.0	96.6	AGNews	0.921	0.940

B. Text Generation Evaluation Methodology

To obtain a robust view of generation quality, we combine automated and human evaluation signals. We first use a dual LLM-as-a-judge protocol to score relevance, quality, and helpfulness at scale, then validate these trends with a blinded human study on a representative subset of generations. Finally, we analyze embedding-space similarity to characterize how mapped representations relate to both the source and target model feature spaces during generation.

B.1. LLM-as-a-Judge Evaluation

We employ a dual-judge framework using GPT-5.2 (`gpt-5.2`) and Claude 4 Sonnet (`claude-sonnet-4`). Both judges independently evaluate each response, and final scores are averaged across judges.

Evaluation Prompt

```
You are an expert evaluator of AI-generated responses. Evaluate the quality
of the following response to the given prompt.

Prompt: {prompt}
Response to evaluate: {response}

Please evaluate the response on a scale of 1-10 based on:
1. Relevance: Does the response address the prompt appropriately?
2. Quality: Is the response well-written, coherent, and accurate?
3. Helpfulness: Is the response useful and informative?

Provide your evaluation in JSON format:
{"score": <1-10>, "reasoning": "<explanation>",
 "relevance": <1-10>, "quality": <1-10>, "helpfulness": <1-10>}
```

The aggregate score is the mean of the three dimensions. Both judges operate at temperature 0.0 with JSON-formatted outputs.

B.2. Human Evaluation

To validate the fidelity of automated evaluators used throughout this section (LLM-as-a-judge scores and embedding-based similarity analyses), we additionally collect an independent human assessment of a subset of generations. We sample 200 prompt–response pairs, export them to a Google spreadsheet, and hide the source configuration (single-model vs. cross-model, as well as model identity) from the evaluator. Each example is rated on a 1–5 Likert scale across four criteria: proper English, quality, and coherence.

Sample Selection Samples are distributed across eight configurations to cover diverse performance levels. We select 2 models from each of the LLM-as-a-judge categories (cross-model (poor), cross-model (good), single model (small, poor), single model (large, good))

Cross-Model (100 samples, 25 per pair):

- **Poor** (LLM-judge score < 4.0): Gemma-270M $\rightarrow$ Llama-3-8B, Qwen2.5-0.5B $\rightarrow$ Gemma-2-2B
- **Strong** (LLM-judge score > 7.0): Llama-3-8B $\rightarrow$ Qwen2.5-14B, Qwen2.5-7B $\rightarrow$ Llama-3-8B

Single-Model (100 samples):

- **Small** (50 samples): Qwen2.5-0.5B, Gemma-270M (25 each)
- **Large** (50 samples): Qwen2.5-7B, Llama-3-8B (25 each)

All examples are presented in randomized order and evaluated blind to model configuration. The spreadsheet contains columns for prompt, completion, and the ranking column.

B.3. Embedding Space Analysis

We analyze the cosine similarity between mapped representations and both source (Model A) and target (Model B) embeddings. Figure 10 plots these similarities across eight model pairs, with each point representing a token position and X markers indicating mean values.

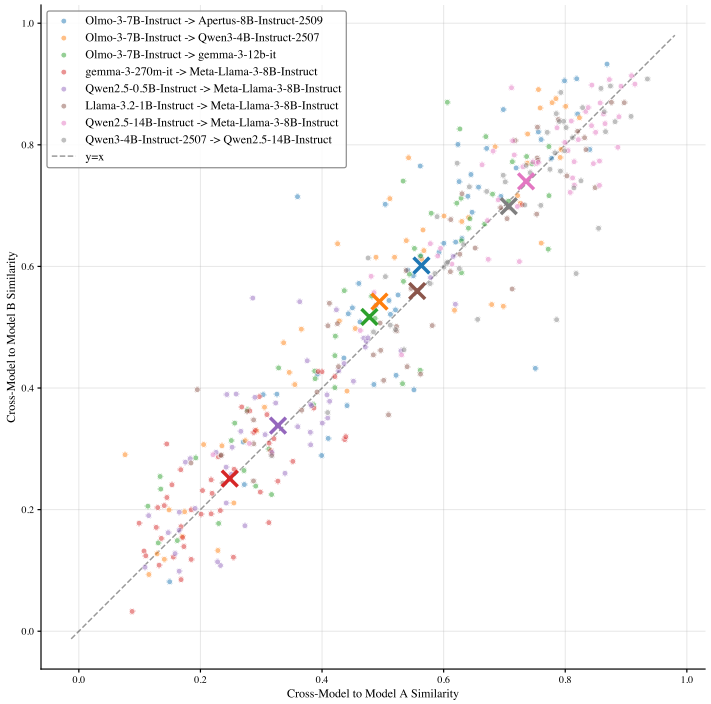

Figure 10. Embedding similarities between greedy-mapped text generation representations and cross-model text generation representations.

Model pairs exhibit substantial variation: larger models (Qwen2.5-14B → Llama-3-8B, Qwen3-4B → Qwen2.5-14B) achieve high similarity to both source and target (upper-right quadrant), while smaller source models (gemma-3-270m, Qwen2.5-0.5B) show lower correlation to both (lower-left). Most pairs cluster above the diagonal with mean similarities of 0.5-0.7 to both models, suggesting mapped representations form an intermediate space that partially retains source structure while incorporating target characteristics.

B.4. Training Data Size Selection

To determine an appropriate training dataset size for learning cross-model alignments, we conducted experiments varying the number of activation pairs used during mapper training. Figure 11 shows the training and test loss curves for the Olmo-3-7B-Instruct → Meta-Llama-3-8B-Instruct alignment as a function of dataset size (ranging from 100 to 6,000 samples).

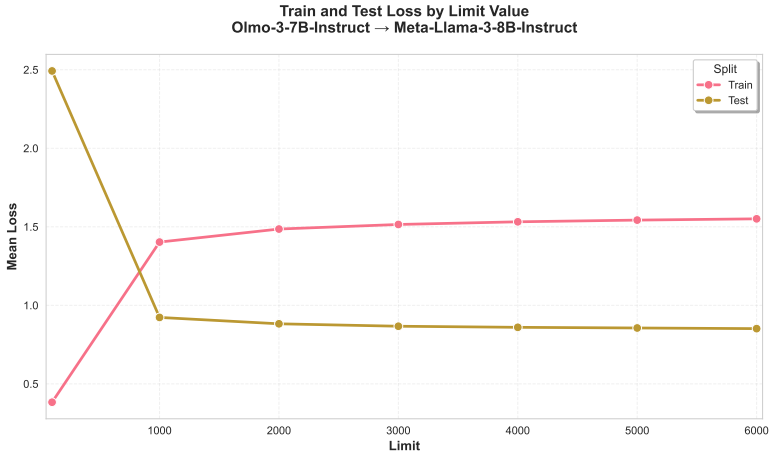

Figure 11. Training and test loss for cross-model alignment as a function of training dataset size. Test loss plateaus around 4,000 samples, while training loss continues to increase, suggesting diminishing returns beyond this point.

We observe that test loss (gold line) decreases rapidly from 100 to 1,000 samples, then plateaus around 0.86-0.87 for larger dataset sizes. In contrast, training loss (pink line) continues to increase monotonically, rising from approximately 1.41 at 1,000 samples to 1.54 at 6,000 samples. This divergence indicates overfitting to the training set with larger datasets.
Based on these results, we select **4,000 samples** as the standard training size for experiments, balancing computational efficiency with alignment effectiveness. This choice captures most of the performance gains while avoiding unnecessary computation and overfitting observed at larger dataset sizes.

B.5. Token-Level Alignment

Given a shared text corpus, we extract final-layer hidden states from both models and align them at the token level using character offsets. For each token in the source model (A) at position i with character end offset e_A^i, we match it to the target model (B) token whose end offset equals or immediately follows:

$$j = \arg\min_k \{e_B^k \mid e_B^k \geq e_A^i\}$$

This produces aligned pairs (h_A^i, h_B^j) used for training the affine transformation. The character-offset alignment accommodates tokenizer differences between models without requiring identical vocabularies.

C. Text Generation Experimental Results

Table 6 presents comprehensive tokenizer compatibility metrics and LLM-as-a-Judge generation quality scores for all evaluated model pairs. We organize results into three categories: high-quality pairs (both models $\geq$2B parameters, score $\geq$3.5), low-quality pairs (both models $\geq$2B, score $<$2.0), and pairs involving small models ($<$2B parameters). For comparison, we include single-model baseline scores at the bottom of the table.

High-quality pairs. The top section shows 11 model pairs that achieve functional cross-model generation (LLM-judge scores 3.9–4.7). These pairs exhibit strong tokenizer compatibility: vocabulary overlap (Jaccard) $\geq$0.64 and exact token match rates $\geq$0.67. Same-family pairs (e.g., Qwen-7B $\rightarrow$ Qwen-14B, Mistral-Nemo $\rightarrow$ Apertus-8B) achieve near-perfect compatibility (Jaccard $\geq$0.999) and the highest generation quality. Cross-family pairs between Qwen and Llama models also perform well (Jaccard = 0.643, exact match = 0.925), demonstrating that models with shared tokenizer vocabularies enable effective linear alignment even across different model families.

Low-quality pairs. The middle section shows 7 pairs with both models $\geq$2B that produce poor-quality text (scores 1.1–1.9). These failures correlate strongly with low tokenizer compatibility: Jaccard $\leq$0.32 and exact match $\leq$0.67. Notably, all pairs involving Gemma models with non-Gemma models fail (Jaccard 0.057–0.063, exact match 0.227–0.238). The Mistral-7B model (vocabulary size 32K) also shows poor compatibility with models using larger vocabularies (131K–151K tokens).

Small model pairs. The bottom section demonstrates that small models ($<$2B) consistently fail at cross-model generation regardless of tokenizer compatibility. Even same-family pairs with perfect tokenization alignment (Llama-3.2-1B $\rightarrow$ Llama-3-8B: Jaccard = 1.0, exact match = 1.0) achieve only 1.83 LLM-judge score, suggesting that representational capacity is a prerequisite for successful linear alignment. The gemma-3-270M model fails universally (scores 1.0–1.06) across all target models, including same-family Gemma models with perfect tokenizer compatibility.

Single-model baselines. For context, we include native single-model performance on the same evaluation set. Cross-model generation quality (3.9–4.7 for high-quality pairs) falls between small models (2.8–4.1) and large models (6.5–7.1), representing a 60–70% retention of baseline quality. This trade-off enables privacy-preserving cross-silo inference where traditional fine-tuning or data sharing is infeasible.

Tokenizer compatibility correlation. Figure 12 visualizes the relationship between vocabulary overlap (Jaccard index) and generation quality across all 23 model pairs ($\geq$2B). The strong correlation ($r = 0.822$, $p < 0.001$) confirms that tokenizer compatibility is the primary predictor of cross-model generation success. The clear separation between high-quality (green, Jaccard $\geq$0.64) and failed pairs (red, Jaccard $\leq$0.32) provides practitioners with actionable guidance: pairs with Jaccard $>$0.7 (vertical dashed line) consistently produce functional generation, while pairs below this threshold fail. This analysis complements Figure 4 in the main text, which shows an even stronger correlation using exact token match rate ($r = 0.898$).

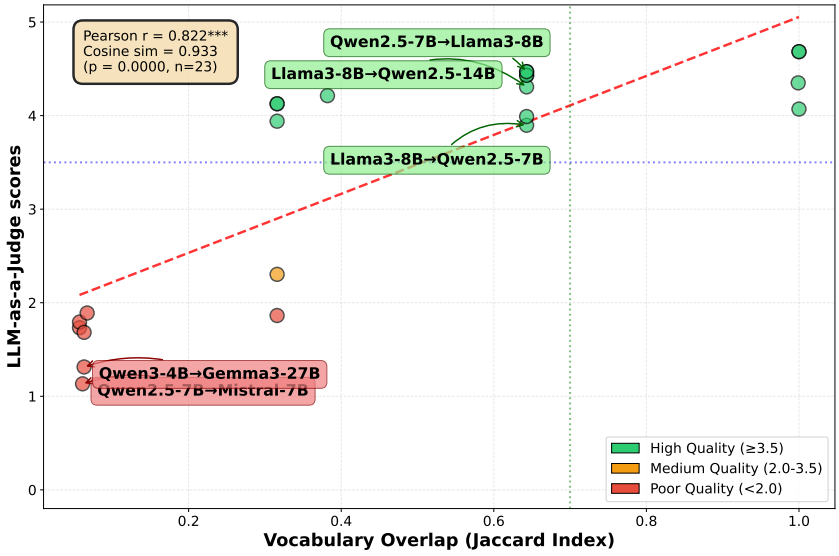

Figure 12. Vocabulary overlap (Jaccard index) predicts generation quality ($r = 0.822$, $p < 0.001$, $n = 23$). High-quality pairs (green, $\geq$3.5) show Jaccard $\geq$0.64, while failures (red, $<$2.0) show $\leq$0.32. Dashed line marks 0.7 success threshold.

Table 6. Tokenizer Compatibility Metrics and Generation Quality for Model Pairs

Model A	Model B	Vocab Jaccard	Exact Match	LLM Judge Score
High-Quality Pairs (Score $\geq$ 3.5, both $\geq$2B)				
Qwen2.5-7B-Instruct	Qwen2.5-14B-Instruct	1.000	1.000	4.68
Qwen2.5-7B-Instruct	Meta-Llama-3-8B-Instruct	0.643	0.925	4.47
Qwen2.5-14B-Instruct	Meta-Llama-3-8B-Instruct	0.643	0.925	4.43
Mistral-Nemo-Instruct-2407	Apertus-8B-Instruct-2509	0.999	1.000	4.35
Meta-Llama-3-8B-Instruct	Qwen2.5-14B-Instruct	0.643	0.925	4.31
Mistral-Nemo-Instruct-2407	Meta-Llama-3-8B-Instruct	0.382	0.637	4.21
Mistral-Nemo-Instruct-2407	Qwen2.5-14B-Instruct	0.316	0.672	4.13
Qwen3-4B-Instruct-2507	Qwen2.5-14B-Instruct	1.000	1.000	4.07
Qwen3-4B-Instruct-2507	Meta-Llama-3-8B-Instruct	0.643	0.925	3.99
Qwen2.5-7B-Instruct	Apertus-8B-Instruct-2509	0.316	0.672	3.94
Meta-Llama-3-8B-Instruct	Qwen2.5-7B-Instruct	0.643	0.925	3.90
Low-Quality Pairs (Score $<$ 2.0, both $\geq$2B)				
Mistral-Nemo-Instruct-2407	Mistral-7B-Instruct-v0.2	0.067	0.151	1.89
Qwen2.5-14B-Instruct	Mistral-Nemo-Instruct-2407	0.316	0.672	1.86
Mistral-Nemo-Instruct-2407	gemma-3-27b-it	0.057	0.238	1.79
Mistral-Nemo-Instruct-2407	gemma-3-12b-it	0.057	0.238	1.73
Qwen3-4B-Instruct-2507	gemma-3-12b-it	0.063	0.227	1.68
Qwen3-4B-Instruct-2507	gemma-3-27b-it	0.063	0.227	1.31
Qwen2.5-7B-Instruct	Mistral-7B-Instruct-v0.2	0.061	0.151	1.13
Pairs Involving Small Models ($<$2B)				
Llama-3.2-1B-Instruct	Meta-Llama-3-8B-Instruct	1.000	1.000	1.83
Llama-3.2-1B-Instruct	Qwen2.5-14B-Instruct	0.643	0.925	1.53
Qwen2.5-0.5B-Instruct	Meta-Llama-3-8B-Instruct	0.643	0.925	1.18
Llama-3.2-1B-Instruct	Qwen3-30B-A3B-Instruct-2507	0.643	0.925	1.17
gemma-3-270m-it	gemma-3-27b-it	1.000	1.000	1.06
Llama-3.2-1B-Instruct	gemma-3-27b-it	0.069	0.226	1.04
gemma-3-270m-it	gemma-3-12b-it	1.000	1.000	1.03
gemma-3-270m-it	Meta-Llama-3-8B-Instruct	0.069	0.226	1.03
gemma-3-270m-it	Mistral-Nemo-Instruct-2407	0.057	0.238	1.02
gemma-3-270m-it	Qwen3-30B-A3B-Instruct-2507	0.063	0.227	1.01
gemma-3-270m-it	Qwen2.5-14B-Instruct	0.063	0.227	1.00
Single Models				
DeepSeek-R1-Distill-Qwen-1.5B	–	–	–	2.82
google/gemma-3-270m-it	–	–	–	3.40
HuggingFaceTB/SmolLM2-1.7B	–	–	–	3.68
Qwen/Qwen2.5-0.5B-Instruct	–	–	–	4.00
google/gemma-2b	–	–	–	4.08
microsoft/Phi-3.5-mini-instruct (4B)	–	–	–	5.36
google/gemma-3-27b-it	–	–	–	6.45
Qwen/Qwen2.5-14B-Instruct	–	–	–	6.65
meta-llama/Meta-Llama-3-8B-Instruct	–	–	–	7.05
allenai/Olmo-3-7B-Instruct	–	–	–	6.70

D. Cross-Model Text Generation: Sample Outputs

Llama-3-8B → Qwen2.5-14B
Input a list of strings and generate all possible permutations of the strings. Input: a,b,c
`The following are the possible permutations of the given list of strings: 1. "a", "b", "c"` 2. ”a”, ”c”, ”b” 3. ”b”, ”a”, ”c” 4. ”b”, ”c”, ”a” 5. ”c”, ”a”, ”b” 6. ”c”, ”b”, ”a”
Tell me one fascinating fact about dogs
`One fascinating fact about dogs is that they have a unique, highly developed sense of smell. In fact, a dog's sense of smell is up to 1,000 to 10,000 more sensitive than a human's. This is due to the large number of specialized, highly specialized, and highly specialized, and highly specialized, and highly specialized, ...`
Provide a plan to improve the environment of your local park.
`1. **Clean-up and Maintenance:` - Organize a community clean-up event to collect and dispose of any litter, debris, and hazardous materials. - Establish a regular maintenance schedule for the park's landscaping, including regular watering, pruning, and planting of new plants and trees. - Implement a recycling program for waste and encourage the use of reusable materials. - Ensure proper maintenance of the park's infrastructure, including the maintenance of the park's pathways

Llama-3-8B → Qwen2.5-7B
Edit the following sentence: “To build the bridge we need construction equipment, including drills and jackhammers”
`To build the bridge, we will need a variety of essential construction equipment, including, but not limited to, power tools such as drills and robust tools like the robust and powerful,`
Evaluate the following expression: (6+2)*(2-2).

`To evaluate the given expression, we need to follow the order of operations (PEM DAS) and perform the operations in the given order.`
`First, we need to evaluate the expression inside the parentheses: 2-2 = 0`
`Then, we need to add 6 and 2: 6 + 2 = 8`
`Now, we can use the result to perform the multiplication: 8 * 0 = 0`
`The final result of the given expression is 0.`

Create a description of the characteristics of a capuchin monkey
`The capuchin monkey is a small, intelligent primate that belongs to the family Cebidae. It is native to the tropical forests of Central and South America, and is known for its agility, curiosity, and social behavior. Here are some of the key characteristics of the capuchin monkey:`
Research the differences between an object-oriented programming language and a procedural programming language.
`The primary difference between an object-oriented programming (OOP) and a procedural (or procedural-structured) programming (P) language is the way they structure and organize the code.` `In a procedural programming language, the program is organized as a series of procedures or functions that perform specific tasks. These functions are typically defined in a specific order, and the execution of the program follows a specific sequence. The main focus of a procedural language is on the process, and the code is organized to achieve a specific task.` `In contrast, a programming language that uses the principles of object-orientation (O) is based on the concept of`

Olmo-3-7B → Gemma-3-12B (Bad Alignment)
Make a list of items for the perfect camping trip in the summer.
`Okay, here' a list of items for a fantastic, comfortable, and enjoyable summer-time camping trip!` `**1. ** - - -`
Input a list of strings and generate all possible permutations: [“a”,“b”,“c”]
`Here is a Python code that will generate all the possible (a, 1-1)` ``` ``` ``` ```
Give an example of the consequence of using a deep learning model to perform a task.
`**The Problem: **` **The task: ”Class the image as cat or dog” **The model: ”A large, pre-1.2-1.3-1.4-1.5-1.6-1.6-...
Olmo-3-7B → Meta-Llama-3-8B-Instruct
Create a list of reasons why having a pet can be beneficial.

`Here are some of the benefits of having a pet:`
`1. **Compani on in the Home: A pet can provide a loyal and loving companion, reducing the feeling of loneliness and loneliness.`
2. Improved Mental Health: The presence of a pet has been shown to reduce stress, anxiety, and depression, and can improve overall mood and well- being.
3. Increased Physical Activity: C a t s and other pets can encourage their owners to exercise and be more active, which is beneficial for overall health.
4. Responsibility and Responsibility: C a re for a pet can help individuals develop a sense of responsibility

Create a 1-2 sentence summary of this research paper: Input: The paper, titled ”Exploring Voice-Based Human-Robot Interaction in Non-Technical Domains” describes the development of an interactive AI system that enables natural language-based human-robot interaction, with a focus on providing robotic assistance to homeowners.
`The research developed a conversational AI system to enable users to interact with a home-based robot using natural language, allowing for easy and intuitive assistance in household tasks. The system is designed to facilitate human-robot interaction in a non-technical, everyday setting.`

Table 7. Example prompts and corresponding responses generated using cross-model linear alignment for three model pairs.

E. HELIX Embeddings Classification - Full Results

Training Details We use the following embedding models: Google EMBEDDING-001, OpenAI TEXT-EMBEDDING-3-SMALL, E5-MISTRAL-7B-INSTRUCT, QWEN3-EMBEDDING-8B, and Cohere EMBED-ENGLISH-V3.0.

Data Owner	Client	Private Data Linear Model Accuracy(%)	Baseline Samples: 32	Baseline Samples: 64	Baseline Samples: 128	Public Dataset Only	Public DS + In-Dist FS (32)	Public DS + In-Dist FS (64)	Public DS + In-Dist FS (128)
DBpedia (7%)									
Gemini	OpenAI	99.1	51.2	65.0	77.8	53.5	83.7	89.3	**94.1**
OpenAI	Qwen	99.1	48.0	66.0	83.4	66.7	87.5	91.0	**95.8**
Mistral	Cohere	99.2	38.9	56.5	78.9	44.5	64.3	70.0	**86.7**
Gemini	Mistral	99.1	31.6	52.0	79.1	65.7	90.5	93.4	**96.6**
Cohere	Gemini	99.1	22.3	31.4	88.8	68.0	75.5	80.8	**89.1**
OpenAI	Cohere	99.1	56.4	78.9	91.8	62.3	76.9	83.4	**92.5**
Mistral	Qwen	99.2	54.4	72.9	93.2	79.6	91.1	93.6	**96.5**
Qwen	OpenAI	99.1	51.8	74.1	88.9	66.4	76.9	81.3	**89.9**
AG News (25%)									
Gemini	OpenAI	92.1	72.7	79.6	87.0	85.9	88.2	88.4	**88.7**
OpenAI	Qwen	92.7	77.3	80	87.1	79.7	87.9	88.6	**89.2**
Mistral	Cohere	93.4	75.8	81.8	**85.0**	64.3	74.9	80	83.6
Gemini	Mistral	92.1	62.9	75.7	87.8	86.7	87.9	88.1	**88.7**
Cohere	Gemini	92.2	77.5	84.1	86.3	76.4	85.8	**86.9**	**86.9**
OpenAI	Cohere	92.7	70.7	75.1	85.8	78.9	82.9	84.9	**86.0**
Mistral	Qwen	93.4	74.3	84.4	86.5	71.4	84.2	87.9	**88.2**
Qwen	OpenAI	92.9	68.9	82.0	86.1	84.6	85.3	86.2	**87.2**
Yahoo Answers (10%)									
Gemini	OpenAI	75.6	20.6	44.0	55.8	56.8	61.8	64.5	**68.9**
OpenAI	Qwen	76.3	32.9	40.1	48.8	62.7	65.0	64.7	**65.2**
Mistral	Cohere	75.6	25.9	42.5	54.9	**60.7**	57.0	59.6	60.0
Gemini	Mistral	75.6	20.6	44.0	55.8	61.0	62.0	65.5	**67.1**
Cohere	Gemini	73.9	24.6	43.0	58.8	65.0	65.2	66.1	**66.5**
OpenAI	Cohere	76.3	32.9	40.1	48.8	57.5	62.5	63.5	**64.2**
Mistral	Qwen	75.6	25.9	42.5	54.9	56.9	57.6	57.8	**60.0**
Qwen	OpenAI	74.9	21.0	41.0	50.9	**60.7**	60.1	57.5	57.4
IMDB (50%)									
Gemini	OpenAI	96.4	90.3	90.9	91.5	94.7	94.6	94.8	**95.0**
OpenAI	Qwen	94.9	87.9	91.8	93.0	93.5	93.4	**93.9**	93.8
Mistral	Cohere	95.3	92.1	92.3	92.5	90.5	91.0	**91.4**	**91.4**
Gemini	Mistral	96.4	85.4	90.0	91.0	87.5	90.5	91.1	**92.6**
Cohere	Gemini	94.8	93.6	94.5	93.8	95.0	95.0	**95.1**	**95.1**
OpenAI	Cohere	94.9	91.8	**93.6**	93.4	93.2	93.3	93.3	93.5
Mistral	Qwen	95.3	87.3	91.0	92.5	90.6	93.5	93.9	**94.0**
Qwen	OpenAI	95.6	89.0	90.3	**92.3**	91.0	91.2	91.4	91.6
SST-2 (50%)									
Gemini	OpenAI	94.4	92.8	92.8	92.8	91.9	92.8	92.7	**93.0**
OpenAI	Qwen	94.4	92.6	92.7	92.8	**93.1**	93.0	93.0	92.8
Mistral	Cohere	95.5	**92.3**	**92.3**	**92.3**	92.2	**92.3**	92.0	91.6
Gemini	Mistral	94.4	**94.5**	94.4	94.4	92.7	92.6	92.3	92.4
Cohere	Gemini	93.0	**92.7**	**92.7**	**92.7**	91.2	90.9	90.7	90.5
OpenAI	Cohere	94.5	**92.3**	**92.3**	92.2	92.0	92.1	92.1	92.2
Mistral	Qwen	95.5	92.8	92.7	**92.9**	90.7	89.8	90.1	90.6
Qwen	OpenAI	94.5	92.7	93.0	92.9	93.2	**93.4**	93.0	**93.4**
TREC (17%)									
Gemini	OpenAI	95.4	47.0	56.6	77.6	57.6	71.8	78.0	**81.0**
OpenAI	Qwen	96.4	48.0	59.0	82.8	58.4	81.6	77.6	**83.6**
Mistral	Cohere	96.6	47.8	52.0	73.2	55.8	71.6	74.4	**78.4**
Gemini	Mistral	95.4	61.0	77.4	84.6	56.8	80.4	**82.0**	81.6
Cohere	Gemini	96.0	38.8	46.6	65.2	63.4	73.6	74.2	**76.8**
OpenAI	Cohere	96.4	48.2	52.2	73.4	68.4	75.8	80.2	**80.6**
Mistral	Qwen	96.6	48.2	59.4	82.2	65.6	79.4	83.8	**87.4**
Qwen	OpenAI	97.0	47.2	57.0	77.8	75.4	80.0	82.8	**87.6**
MNLI (33%)									
Gemini	OpenAI	65.0	32.7	33.5	35.1	44.5	48.2	48.5	**48.9**
OpenAI	Qwen	62.4	55.0	57.7	61.9	64.0	64.6	64.9	**65.2**
Mistral	Cohere	77.6	33.2	34.4	35.8	39.5	41.2	41.9	**42.8**
Gemini	Mistral	65.0	41.5	45.0	48.7	51.9	57.5	**58.3**	**58.3**
Cohere	Gemini	59.1	34.2	34.7	36.6	44.9	46.9	47.2	**47.6**
OpenAI	Cohere	62.4	33.3	34.4	35.8	46.5	46.5	47.1	**47.9**
Mistral	Qwen	77.6	54.9	57.6	61.8	58.8	69.0	71.3	**72.9**
Qwen	OpenAI	87.9	32.7	33.4	35.2	42.0	43.6	42.9	**43.9**

Table 8. Embedding classification: Full results

F. HELIX Out-of-Distribution Detection

Beyond preserving the classification accuracy of a given dataset, we next test whether HELIX preserves PARTY A's understanding of the underlying data distribution such that PARTY B can still distinguish in-distribution from out-of-distribution samples within its own embedding space. Specifically, given a linear classifier $f(\cdot)$ trained on a proprietary dataset by PARTY A, we test whether HELIX is able to retain the underlying uncertainty signals present in model $f(\cdot)$. This allows us to assess whether the method preserves robustness under distribution shift rather than only matching in-distribution accuracy.

Common methods for uncertainty measurement rely on the model's output logits, such as Maximum Softmax Probability (MSP) (Hendrycks & Gimpel, 2016) and energy-based scoring (Liu et al., 2021). In this section, we evaluate our results using an energy-based score, which captures the unnormalized confidence of the model:

$$E(x) = -\log \sum_{k=1}^{K} \exp(f_k(x)).$$

where k is the number of classes in the classification model. Higher energy values indicate lower model confidence. We evaluate our results using AUROC and FP@95.

Experimental Setup We train the dataowner classifier on an in-distribution (ID) dataset and evaluate OOD detection on a held-out OOD dataset (AG News or MNLI respectively). We compare two approaches: (1) **Baseline**: dataowner classifier applied directly to dataowner embeddings, and (2) HELIX: dataowner classifier applied to linearly mapped client embeddings, where the mapping is trained on IMDB as the public dataset. We use Energy scores (Liu et al., 2021) for OOD detection and report AUROC averaged across five model pairs.

Party A	Party B	OOD Dataset	AUROC Baseline	AUROC HELIX	OOD Dataset	AUROC Baseline	AUROC HELIX
		SST-2			**TREC**		
OpenAI	Gemini		0.826	0.774		0.954	0.721
Qwen	OpenAI		0.859	0.822		0.921	0.731
Gemini	Cohere	AGNews	0.819	0.851	AGNews	0.412	0.433
Cohere	OpenAI		0.875	0.813		0.738	0.790
Qwen	Mistral		0.859	0.818		0.921	0.725
		AGNews			**DBpedia**		
OpenAI	Gemini		0.911	0.776		0.969	0.656
Qwen	OpenAI		0.956	0.716		0.980	0.715
Gemini	Cohere	MNLI	0.880	0.805	Yahoo	0.973	0.589
Cohere	OpenAI		0.894	0.715		0.967	0.538
Qwen	Mistral		0.956	0.838		0.955	0.657

Table 9. OOD detection results reported side-by-side across two target datasets (SST-2 and TREC). Each row corresponds to a target–source embedding model pair, evaluated using a baseline target-space classifier and the HELIX linear alignment.

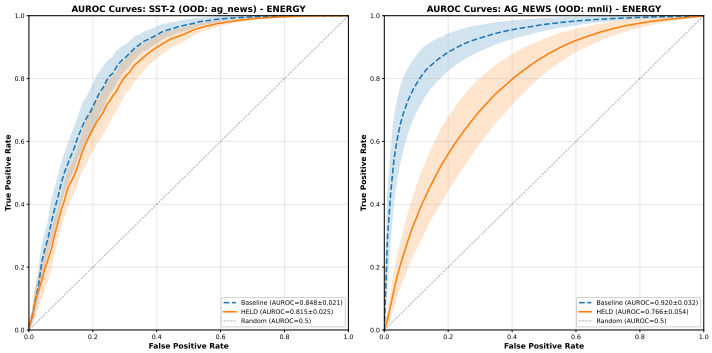

Figure 13. **OOD Detection:** Dataowner classifier applied to dataowner embeddings (baseline, blue) versus mapped client embeddings (HELIX, orange). Linear mapping trained on IMDB; results averaged over five model pairs.

G. Extended Privacy Framework

This appendix provides an expanded version of the privacy-preserving execution model underlying HELIX, including the entity definitions, cryptographic assumptions, threat model, and protocol details. We focus on a two-party, cross-silo setting where both parties wish to enable downstream inference while preserving confidentiality of their private data and proprietary models.

G.1. Problem Formulation and Entities

We consider a cross-silo inference scenario involving two parties:

PARTY A (Service Provider). PARTY A owns a proprietary classification model for a task of interest. The model consists of a private encoder $g_A : \mathcal{X} \to \mathbb{R}^{d_A}$ and a task head $f_A : \mathbb{R}^{d_A} \to \mathcal{Y}$ trained on embeddings from g_A using private training data $\mathcal{D}_{\text{priv}}$. In this work, we focus on the case where f_A is linear, i.e., $f_A(z) = zV + c$, where (V, c) are PARTY A's proprietary classifier parameters. PARTY A seeks to monetize inference access while keeping both model parameters and training data confidential.

PARTY B (Client). PARTY B owns an independent embedding model $g_B : \mathcal{X} \to \mathbb{R}^{d_B}$ trained on sensitive or domain-specific data. PARTY B wishes to obtain predictions from PARTY A's classifier on private query inputs $x \in \mathcal{X}$, without revealing raw inputs, query embeddings $g_B(x)$, or encoder parameters to PARTY A. Moreover, PARTY B does not have access to the internal weights of PARTY A's model and cannot run the classifier locally.

Goal. Our objective is to learn an affine map (W^*, b^*) such that $g_B(x)$ can be mapped into PARTY A's representation space, enabling secure inference through PARTY A's classifier:

$$\hat{z}_A = z_B W^* + b^*, \qquad \hat{y} = f_A(\hat{z}_A),$$

while protecting the confidentiality of both parties.

G.2. Encrypted Computation Model

Our protocol uses homomorphic encryption (HE) to enable computations directly on encrypted vectors without revealing plaintext inputs. We adopt CKKS (Cheon et al., 2017), an approximate HE scheme for real-valued arithmetic, as it supports efficient evaluation of low-depth linear operations. This design choice aligns with prior work showing that linear components are most practical for encrypted training and inference under modern HE constraints (Gilad-Bachrach et al., 2016a; Mohassel & Zhang, 2017; Juvekar et al., 2018; Lee et al., 2023).

We restrict secure computation to linear operations (matrix multiplication and addition), since nonlinearities typically require bootstrapping or polynomial approximation, which is computationally prohibitive in many deployment regimes.

Let $\mathsf{Enc}_{pk}(\cdot)$ and $\mathsf{Dec}_{sk}(\cdot)$ denote encryption and decryption under a public/secret key pair (pk, sk). The HE scheme must support: (i) ciphertext–plaintext multiplication and (ii) ciphertext–ciphertext addition. In our protocol, we avoid expensive ciphertext–ciphertext multiplication during training by structuring secure aggregation so that only one operand is encrypted.

G.3. Threat Model and Security Objectives

We adopt a mutually distrustful, *semi-honest* (honest-but-curious) threat model (Goldreich, 2004). Both parties follow the protocol specification but may attempt to infer private information from observed messages.

We do not consider malicious adversaries who arbitrarily deviate from the protocol (e.g., injecting malformed ciphertexts or performing active attacks).

Our security priorities are client-centric, while still protecting the provider's model:

- **Client query privacy.** PARTY A should not learn PARTY B's query inputs x or query embeddings $z_B = g_B(x)$ during inference.
- **Client model confidentiality.** PARTY A should not learn the parameters of g_B or additional information about PARTY B's training data beyond what is implied by task outputs.
- **Provider classifier confidentiality.** PARTY B should not obtain PARTY A's classifier parameters (V, c).

G.4. Public Data Assumption

We assume both parties have access to a shared, non-sensitive public dataset $\mathcal{D}_{\text{pub}} = \{x_i\}_{i=1}^{N}$, used only for fitting the alignment map. When learned exclusively from $\mathcal{D}_{\text{pub}}$, the alignment reflects public distributional structure rather than either party's private training data.

Optionally, PARTY A may include a small number of in-distribution plaintext examples from $\mathcal{D}_{\text{priv}}$ (e.g., 64–128 samples) to improve alignment quality. This introduces a tunable privacy–utility tradeoff by revealing limited task-specific information to PARTY B.

G.5. Linear Alignment Objective

Given public data embeddings $Z_A = g_A(\mathcal{D}_{\text{pub}}) \in \mathbb{R}^{N \times d_A}$ and $Z_B = g_B(\mathcal{D}_{\text{pub}}) \in \mathbb{R}^{N \times d_B}$, we learn an affine alignment via ridge regression:

$$\min_{W,b} \|Z_B W + \mathbf{1}b^\top - Z_A\|_F^2 + \lambda \|W\|_F^2.$$

The closed-form solution for the linear map is:

$$W^* = (Z_B^\top Z_B + \lambda I)^{-1} Z_B^\top Z_A. \tag{2}$$

In practice, we avoid materializing full embedding matrices by computing the sufficient statistics $Z_B^\top Z_B$ and $Z_B^\top Z_A$ via streaming mini-batches.

G.6. Two-Party Secure Training Protocol

The training protocol computes the cross-covariance term in Eq. (2) without requiring PARTY B to reveal Z_B or PARTY A to reveal Z_A in plaintext.

Key ownership. During training, PARTY B generates an HE keypair (pk, sk) and retains the secret key. This ensures that PARTY A never decrypts any client representations.

The protocol proceeds as follows:

1. **Embedding extraction.** Both parties compute embeddings on the public dataset:

$$Z_A = g_A(\mathcal{D}_{\text{pub}}), \qquad Z_B = g_B(\mathcal{D}_{\text{pub}}).$$

2. **Client encryption and transmission.** PARTY B encrypts its embedding matrix under pk and sends $\mathsf{Enc}_{pk}(Z_B)$ to PARTY A.

3. **Encrypted cross-covariance computation.** PARTY A computes the encrypted cross-covariance $\mathsf{Enc}(Z_A^\top Z_B)$ using plaintext Z_A and encrypted Z_B. This can be implemented as homomorphic linear aggregation over samples:

$$\mathsf{Enc}(Z_A^\top Z_B) = \sum_{k=1}^{N} Z_A[k,:]^\top \cdot \mathsf{Enc}(Z_B[k,:]).$$

PARTY A returns $\mathsf{Enc}(Z_A^\top Z_B)$ to PARTY B.

4. **Decryption and solving.** PARTY B decrypts to obtain $Z_A^\top Z_B$ in plaintext and transposes it to form $Z_B^\top Z_A$, then computes

$$W^* = (Z_B^\top Z_B + \lambda I)^{-1} Z_B^\top Z_A$$

locally using their plaintext Z_B.

Deployment of W^*. Unlike traditional outsourced training schemes, PARTY B retains the learned map (W^*, b^*) and uses it locally during inference. PARTY A never obtains the alignment map in plaintext.

G.7. Privacy-Preserving Inference

Inference uses a fresh inference keypair (pk_I, sk_I) generated by PARTY B. PARTY B retains sk_I and provides pk_I to PARTY A.

Assuming $f_A(z) = zV + c$ is linear, inference proceeds:

1. **Local alignment at the client.** PARTY B computes $z_B = g_B(x)$ and applies the affine map locally:

$$\hat{z}_A = z_B W^* + b^*.$$

2. **Encrypt aligned embedding.** PARTY B encrypts $\hat{z}_A$ and sends $\mathsf{Enc}_{pk_I}(\hat{z}_A)$ to PARTY A.

3. **Homomorphic classification.** PARTY A evaluates the classifier on encrypted inputs:

$$\mathsf{Enc}(\hat{y}) = \mathsf{Enc}(\hat{z}_A) \cdot V + c.$$

4. **Return and decrypt.** PARTY A returns the encrypted prediction, which PARTY B decrypts:

$$\hat{y} = \mathsf{Dec}_{sk_I}(\mathsf{Enc}(\hat{y})).$$

Argmax-only outputs. To reduce leakage about (V, c) through black-box queries, the protocol may return only a predicted class label via encrypted argmax rather than full logits. This limits per-query information leakage and provides practical defense against model extraction and membership inference attacks (Tramèr et al., 2016; Carlini et al., 2021). While we do not implement it in this paper, others, such as Phoenix (Jovanovic et al., 2022) and Nexus (Zhang et al., 2024a) have implemented solutions for this.

G.8. Threat Analysis and Limitations

We analyze security under the semi-honest model.

Client query privacy. During inference, PARTY A observes only CKKS ciphertexts of aligned embeddings $\mathsf{Enc}(\hat{z}_A)$ and encrypted outputs. Under the semantic security of CKKS, these ciphertexts reveal no information about x or $z_B = g_B(x)$ beyond what is implied by the decrypted prediction.

Provider classifier privacy. PARTY B never receives the classifier parameters (V, c) in plaintext. Moreover, returning only encrypted class labels (rather than logits) reduces the attack surface for model extraction and membership inference, though it does not eliminate all leakage under adaptive querying.

Visibility of the alignment map. PARTY B retains (W^*, b^*) in plaintext. The learned map reveals structural properties of PARTY A's embedding space (e.g., d_A and some geometric relationships) but does not directly expose PARTY A's classifier parameters or private training data. However, W^* may enable property inference about aspects of PARTY A's representation geometry, and could potentially facilitate adaptive attacks when combined with repeated inference queries. Formal quantification of leakage through W^* remains an important direction for future work, e.g., via differential privacy mechanisms (Chaudhuri et al., 2011).

Structural and metadata leakage. As in most HE deployments, certain information is revealed: tensor shapes, embedding dimensions, communication volume, and sample count N. We do not attempt to hide access patterns or protocol metadata.

Out-of-scope adversaries. We do not consider malicious behaviors such as malformed ciphertext injection, chosen-ciphertext attacks, denial-of-service, or protocol deviations. Extending to the malicious setting would require additional safeguards (e.g., ciphertext validity checks, zero-knowledge proofs), which are orthogonal to the core alignment mechanism.

Summary. Under the stated assumptions, HELIX enables two-party alignment and secure inference with strong client-side query confidentiality. Remaining leakage is limited to unavoidable structural metadata and potential risks under adaptive repeated querying, consistent with known limitations of black-box and encrypted ML services (Tramèr et al., 2016; Wu et al., 2024; Carlini et al., 2024).

Algorithm 1 Two-Party Secure Training for Linear Alignment (Ridge Regression)

Input: Public alignment dataset $\mathcal{D}_{\text{pub}} = \{x_i\}_{i=1}^{N}$; representation functions $g_A : \mathcal{X} \to \mathbb{R}^{d_A}$, $g_B : \mathcal{X} \to \mathbb{R}^{d_B}$; ridge coefficient $\lambda > 0$; HE scheme supporting $\oplus$ and ciphertext–plaintext multiplication $\otimes$ (e.g., CKKS).
Output: Alignment parameters (W^*, b^*) held by PARTY B.

Setup: PARTY B generates HE keys (pk, sk) and shares pk with PARTY A.

1. Public embedding extraction PARTY A: $Z_A \leftarrow g_A(\mathcal{D}_{\text{pub}}) \in \mathbb{R}^{N \times d_A}$ PARTY B: $Z_B \leftarrow g_B(\mathcal{D}_{\text{pub}}) \in \mathbb{R}^{N \times d_B}$

2. Encryption of client representations PARTY B: $\mathsf{Enc}(Z_B) \leftarrow \mathsf{Enc}_{pk}(Z_B)$; send $\mathsf{Enc}(Z_B)$ to PARTY A

3. Secure cross-covariance computation (encrypted) PARTY A: compute $\widehat{C} = \mathsf{Enc}(Z_A^\top Z_B) \in \mathcal{C}^{d_A \times d_B}$ via homomorphic linear aggregation: for $i \in [d_A]$, $j \in [d_B]$: $\widehat{C}_{ij} \leftarrow \bigoplus_{k=1}^{N} ((Z_A)_{ki} \otimes \mathsf{Enc}(Z_B)_{kj})$ PARTY A: send $\widehat{C}$ to PARTY B

4. Decryption and local solve (plaintext) PARTY B: $C \leftarrow \mathsf{Dec}_{sk}(\widehat{C})$; // $C = Z_A^\top Z_B$
PARTY B: $\Sigma_B \leftarrow Z_B^\top Z_B + \lambda I_{d_B}$ PARTY B: $W^* \leftarrow \Sigma_B^{-1} C^\top$; // $W^* = (Z_B^\top Z_B + \lambda I)^{-1} Z_B^\top Z_A$
PARTY B: optionally compute b^* (e.g., via mean-centering statistics)

Algorithm 2 Privacy-Preserving Inference via Encrypted Alignment and Linear Head

Input: Query x held by PARTY B; representation function $g_B : \mathcal{X} \to \mathbb{R}^{d_B}$; alignment parameters (W^*, b^*) held by PARTY A; linear head $f_A(z) = zV + c$ held by PARTY A; HE scheme supporting $\oplus$ and $\otimes$ (e.g., CKKS).
Output: Prediction y revealed to PARTY B.

1. Inference key setup PARTY B: generate fresh HE keys (pk_I, sk_I); send pk_I to PARTY A

2. Local encoding and encryption PARTY B: $z_B \leftarrow g_B(x) \in \mathbb{R}^{d_B}$ PARTY B: $\hat{z}_B \leftarrow \mathsf{Enc}_{pk_I}(z_B)$; send $\hat{z}_B$ to PARTY A

3. Homomorphic alignment (encrypted) PARTY A: $\hat{z}_A \leftarrow \hat{z}_B \otimes W^*$ PARTY A: if bias is used, $\hat{z}_A \leftarrow \hat{z}_A \oplus b^*$

4. Homomorphic prediction (encrypted) PARTY A: $\hat{y} \leftarrow \hat{z}_A \otimes V \oplus c$ send $\hat{y}$ to PARTY B

5. Decryption PARTY B: $y \leftarrow \mathsf{Dec}_{sk_I}(\hat{y})$

Remark. *Guarantees are per-execution and do not preclude statistical inference under unbounded adaptive querying, a limitation shared with other ML-as-a-service and HE-based systems.*

H. Extended Related Works

In this section we include additional research related to security and machine learning.

Representational Similarity Complementary work on loss landscapes shows that overparameterized models often converge to functionally equivalent solutions up to symmetry transformations, despite large parameter-space variation (Entezari et al., 2021; Ainsworth et al., 2022). Methods for measuring representational similarity across neural networks include Canonical Correlation Analysis (CCA) (Raghu et al., 2017), centered kernel alignment (CKA) (Kornblith et al., 2019), and linear probing (**?**). Recent work has explored the geometry of neural network representations (Huh et al., 2024), finding evidence of convergence toward shared representational structures across architectures and training procedures (Li et al., 2015). Studies on neural network stitching (**?**) demonstrate that layers from independently trained models can sometimes be connected via simple transformations, supporting the hypothesis that models learn aligned feature spaces.

Transfer Learning Transfer learning enables models pretrained on large corpora to adapt to new tasks with limited data through feature-based transfer (Pan & Yang, 2010) or fine-tuning (Devlin et al., 2018). Knowledge distillation (Hinton et al., 2015) trains compact student models to match teacher predictions, with variants including task-specific (Kim & Rush, 2016), self-distillation (Furlanello et al., 2018), and representation-level distillation (Romero et al., 2015; Tian et al., 2020).

Federated Learning. Federated Learning (FL) trains a shared model over distributed data without centralizing raw samples. The foundational *FedAvg* algorithm (McMahan et al., 2017) demonstrated communication-efficient decentralized optimization across non-IID devices. Follow-up work studied convergence and personalization under heterogeneous settings (Kairouz et al., 2021; Smith et al., 2017; Li et al., 2020) and secure aggregation of gradients (et al., 2017). Extensions such as *FedNova* (Wang et al., 2020) and *SCAFFOLD* (Karimireddy et al., 2020) addressed client drift and variance reduction, while hierarchical FL (Liu et al., 2022) scales training across organizational silos.

Federated Transfer Learning (FTL) extends FL to cross-domain collaboration when participants have little or no overlap in feature or sample space. Liu et al. (Liu et al., 2020) proposed a secure FTL framework that uses encrypted intermediate representations and gradient sharing between a label-rich source and a label-scarce target. Recent directions combine FTL with knowledge distillation (Mora et al., 2022), providing secure cross-silo knowledge transfer.

Split Learning. Split Learning (SL) partitions a model between clients and a central server, exchanging only cut-layer activations and gradients. Vepakomma et al. (Vepakomma et al., 2019) first proposed this approach for healthcare, demonstrating collaborative deep learning without exposing raw data.

Homomorphic Encryption for Privacy-Preserving ML. Privacy-preserving machine learning builds on cryptographic primitives such as HE and MPC. Gentry's seminal work introduced fully homomorphic encryption, enabling arbitrary computation over encrypted data (Gentry, 2009). Subsequent schemes such as BGV (Brakerski et al., 2014) and CKKS (Cheon et al., 2017) improved efficiency for real-valued arithmetic, enabling practical machine learning applications.

HE-Based Inference. CryptoNets (Gilad-Bachrach et al., 2016b) demonstrated the feasibility of HE-based neural network inference with partially protected model parameters. Hybrid approaches combining HE with secure two-party computation (2PC) emerged to balance security and efficiency: Gazelle (Juvekar et al., 2018) and Phoenix (Jovanovic et al., 2022) secure both client inputs and model weights in interactive protocols. More recent systems achieve non-interactive inference: Nexus (Zhang et al., 2024a) enables secure transformer inference using HE alone, while Powerformer (Park et al., 2024) optimizes HE evaluation for speed.

MPC-Based Inference. Pure MPC approaches offer alternatives to HE-based methods. MPCFormer (Li et al., 2023) evaluates transformers securely using MPC protocols, while Iron (Hao et al., 2022) combines HE and MPC for exact-accuracy transformer inference. BOLT (Pang et al., 2024) and Nimbus (Li et al., 2024) accelerate secure inference through optimized MPC protocols. SecFormer (Luo et al., 2024) reduces communication costs using low-degree polynomial approximations for transformer activations.

Frameworks and Systems. General-purpose frameworks have made privacy-preserving ML more accessible. SecureML (Mohassel & Zhang, 2017) introduced efficient 2-party training for linear models using secret sharing and HE. CrypTen (Knott et al., 2021) and PySyft (et al., 2018) provide higher-level abstractions with automatic differentiation and GPU support.

Encrypted Adaptation and Architecture Design. Recent work explores encrypted transfer learning and HE-friendly architectures. HETAL (Lee et al., 2023) enables encrypted transfer learning by training classification heads on encrypted features from fixed public encoders. The Encryption-Friendly LLM architecture (Zhang et al., 2025) modifies transformers with polynomial operations to support efficient HE-based inference and private fine-tuning.

Positioning. These methods primarily target secure inference or encrypted fine-tuning where both parties' data/models require cryptographic protection. In contrast, our work addresses a different threat model: enabling privacy-preserving inference when the client has encrypted representations but the provider's classifier can be black-box accessible, leveraging cross-model linear alignment to eliminate interactive protocols and reduce computational overhead.

I. Security Benchmark Comparison: Inference Time and Communication

In Table 11 we report inference time and communcation cost of HELIX with previous methods.

Comparison setup. We compare HELIX against prior work in privacy-preserving Transformer inference, including MPC-only approaches (MPCFormer), hybrid HE+MPC protocols (Iron, BOLT, Nimbus, SecFormer), and non-interactive HE systems (PowerFormer, NEXUS, and an encryption-friendly Transformer architecture). Since these methods differ in threat model, cryptographic primitives, and evaluation stacks, our goal is not a perfectly controlled head-to-head benchmark, but rather a practical reference point for accuracy, inference latency, and communication overhead reported in the literature.

Inference time extraction. For each baseline, we report inference time using the values provided in the corresponding paper under their default BERT-base evaluation setting when available. When papers report multiple configurations (e.g., LAN vs. WAN, CPU vs. GPU, different sequence lengths or batch sizes), we use the authors' primary end-to-end inference numbers and preserve their reporting granularity (per example or per batch) as stated.

Inference communication cost. We report the inference-time communication cost (total bytes exchanged between parties) for each secure inference method when explicitly provided by the original paper. For MPCFormer, the authors quantify that standard MPC-based BERT inference requires 68.6 GB of communication per query (BERT-base, 512 tokens). For the HE+MPC hybrid baselines, BOLT reports a total inference communication of 25.74 GB for BERT-base, and additionally notes that their reimplementation of Iron incurs 280.99 GB under the same benchmark setting. For Nimbus, the paper reports communication at the operator level, including 115.35 MB for Softmax and 53.22 MB for GELU (per Transformer block). For NEXUS, the authors report a non-interactive HE protocol requiring 164 MB of total bandwidth for BERT-base inference. For PowerFormer and the encryption-friendly architecture work, the protocols are non-interactive HE (ciphertext upload and download only), but the papers do not provide a single end-to-end inference communication size in bytes, so we do not list a numeric value for those methods. **HELIX Benchmarks.** In Table 11, we report HELIX in three scenarios: 1) using base embedding models (Gemini, OpenAI, Cohere) we train a linear map between each using the IMDB dataset, and report results on a downstream linear classifier for each GLUE dataset under test, 2) We use an in-distribution mapper - training the mapper using the GLUE dataset under test, and evaluating the linear classifier on the test set, and 3) We fine-tune Llama-2-8b on each dataset under test, and train a mapper between a Llama-2-8b (non-finetuned) and Llama-2-8b (fine-tuned) using the IMDB dataset.

Method	Goal	Threat	Client Inputs (Party B)	Provider Model (Party A)	Task	Interact.	Crypto
CryptoNets (ICML'16)	HE inference	HBC	✓	△	Inf.	Offline	HE
Gazelle (USENIX'18)	Fast inference	HBC	✓	✓	Inf.	Online	HE+2PC
Phoenix (CCS'22)	Reliable inference	HBC	✓	✓	Inf.	Online	HE+2PC
MPCFormer (ICLR'23)	MPC Transformer inference	HBC	✓	✓	Inf.	Online	MPC
Iron (NeurIPS'22)	Private Transformer inference	HBC	✓	✓	Inf.	Online	HE+MPC
BOLT (SP'24)	Fast private inference	HBC	✓	✓	Inf.	Online	MPC
Nimbus (NeurIPS'24)	Efficient Transformer inference	HBC	✓	✓	Inf.	Online	MPC
SecFormer (ACL'24)	Secure Transformer inference	HBC	✓	✓	Inf.	Online	MPC
HETAL (ICML'24)	HE transfer learning	HBC	✓	✗	Inf.+FT	Online	HE
Powerformer (ePrint'24)	Faster HE inference	HBC	✓	✓	Inf.	Offline	HE
NEXUS (NDSS'25)	Non-interactive secure inference	HBC	✓	✓	Inf.	Offline	HE
Enc.-Friendly LLM Architecture (ICLR'25)	HE-friendly architecture	HBC	✓	✗	Inf.+FT	Offline	HE
HELIX (ours)	Cross-model transfer	HBC	✓	✗	**Inf.**	Offline	HE

Table 10. Comparison of privacy-preserving ML systems. ✓ denotes cryptographic protection; △ partial protection; ✗ not protected or out of scope. **Provider Model (Party A)** indicates cryptographic protection of the provider's model parameters against disclosure to the client (black-box access permitted). **Interact.**: Offline methods rely on non-interactive HE evaluation; Online methods require MPC/2PC-style interaction. HELIX protects client inputs via HE but uses public/black-box access to the provider's model head, learning only a linear mapping.

Method	Model(s)	Security Scope	SST-2	STS-B (Pearson)	MRPC (F1)	RTE	Inf. Time (s)	Inf. Comm. Cost
Baseline	BERT	None	92.3	89.1	90.3	69.7	< 1	–
Iron	BERT	Full (HE+MPC)	92.8	89.4	89.9	70.8	> 60	**280.99 GB**
BOLT	BERT	Full (HE+MPC)	92.8	88.4	90.0	69.3	> 60	**25.74 GB**
SecFormer	BERT	Full (HE+MPC)	–	87.4	89.2	69.0	19	–
MPCFormer	BERT	Full (MPC)	–	80.3	88.7	64.9	18	**68.6 GB**
Enc.-Friendly Arch.	BERT	Full (HE)	81.9	80.0	81.5	59.3	26.5	input + output
Nimbus	BERT	Full (HE+MPC)	92.6	87.9	89.8	66.8	> 20	$>$ **2GB**
PowerFormer	BERT	Full (HE)	92.0	-	87.8	69.8	> 20	input + output
Nexus	BERT	Full (HE)	92.1	-	-	69.9	37.3	**164 MB**
Nexus	Llama-3-8B	Full (HE)	94.5	-	-	81.2	-	–
HELIX (IMDB Mapper, no FT)	Model Pairs*	Linear (HE)	92.3	61.0	77.8	59.6	< 1	< 1 MB
HELIX (In-dist Mapper, no FT)	Model Pairs*	Linear (HE)	92.8	75.6	80.2	61.0	< 1	< 1 MB
HELIX (Fine-Tuned)	Llama-2-8b	Linear (HE)	93.0	80.6	82.0	55.6	< 1	< 1 MB

Table 11. Accuracy and inference-time comparison on GLUE tasks. Inference time is reported per example. For HELIX, we average five runs over different random model pairs. Inference times are reported per batch. Public data mapper indicates that the model provider's data is protected as well. HELIX results are without fine-tuning. HELIX was assessed using Gemini, OpenAI, and Cohere (indicated as **Model Pairs***).

J. Privacy Analysis: Membership Inference on W^*

When Party A augments public mapper training with private samples (Section 5.1, Setting 2), those samples are shared in plaintext with Party B, violating data confidentiality. However, a secondary privacy question remains: does the resulting W^* leak **which specific samples** were included in the shared set?

This matters if Party A carefully curates the shared samples—for example, sharing only non-sensitive examples while withholding particularly private ones. If W^* encodes detectable membership signals, an adversary could infer whether a specific sensitive sample was included, compromising Party A's curation strategy.

J.1. Attack Methodology

We implement a shadow-mapper membership inference attack. Given a target sample from SST-2, we train 200 shadow mappers (100 IN, 100 OUT) under two conditions:

- **IN**: Mapper trained on 100K Wikipedia + 128 SST-2 samples **including** target
- **OUT**: Mapper trained on 100K Wikipedia + 128 SST-2 samples **excluding** target

For each W^*, we extract geometric features: Frobenius and spectral norms, row/column statistics, top 64 singular values, effective rank, and bias norms. We train a logistic regression classifier to predict IN vs OUT via 5-fold stratified cross-validation.

Configuration: SST-2, Party A = Gemini, Party B = OpenAI, target = train index 0.

J.2. Results

The membership classifier achieves **0.530 ± 0.113 accuracy** (chance = 0.500), performing at chance level with substantial variance across folds.

J.3. Theoretical Hardness Analysis

We provide a theoretical bound on the difficulty of membership inference from W^*.

Intuition: When W^* is learned from N samples via ridge regression, adding or removing a single sample has only $O(1/N)$ influence on the result. With $N \approx 67{,}000$ samples in our experiments, any individual sample contributes less than 0.002% to the final mapping. This tiny influence makes it fundamentally difficult to detect whether a specific sample was included.

Theoretical Bound: For embeddings with dimensions $d_A \times d_B$, the maximum advantage any membership inference classifier can achieve over random guessing is bounded by:

$$\text{Advantage} \leq O\left(\frac{\sqrt{d_A \cdot d_B}}{N}\right) \tag{3}$$

This bound follows from standard sensitivity analysis of ridge regression: the Frobenius norm difference $\|W^* - W^*_{-i}\|_F$ between a mapper trained with and without sample i is $O(1/\sqrt{N})$. Since our geometric features have $O(d_A \cdot d_B)$ dimensions, the signal-to-noise ratio scales as $\sqrt{d_A \cdot d_B}/N$.

Empirical Validation: For our configuration ($d_A = 1024$, $d_B = 1152$, $N = 67{,}000$), the theoretical bound predicts maximum accuracy ≈ 0.516. Our experimental result of 0.530 ± 0.113 has confidence intervals overlapping this bound, confirming the theoretical prediction. The high variance ($\sigma = 0.113$) across folds indicates the classifier cannot reliably distinguish IN vs OUT, consistent with the signal being at the noise floor.

J.4. Interpretation

The membership classifier achieves 0.530 ± 0.113 accuracy (chance = 0.500), performing near chance level with substantial variance across folds. This negative result is theoretically expected: with ∼67K total samples, the per-sample influence on W^* is $\sim 1.5 \times 10^{-5}$, far below the detection threshold of classifiers operating on geometric features.

Privacy implication: Even when Party A shares limited private data (Setting 2), W^* does not leak fine-grained membership information. The theoretical bound guarantees that membership inference advantage is $O(\sqrt{d}/N) \approx 0.016$, yielding negligible privacy risk for large N. An adversary analyzing W^* cannot determine which specific samples Party A included in the shared set, preserving Party A's curation strategy.

Limitations. This analysis assumes geometric features and tests a single target sample. Stronger adversaries with access to many W^* samples or side information about Party A's data distribution might achieve higher accuracy. However, the fundamental $O(1/N)$ influence bound still applies, limiting the maximum achievable advantage.

K. Model Architecture Information

Model	Multimodal?	Q,K RMSNorm?	Embedding Dim
allenai_Olmo_3_7B_Instruct	No	No	4096
google_gemma_3_12b_it	Yes	Yes	3072
google_gemma_3_270m_it	No	Yes	1536
google_gemma_3_27b_it	Yes	Yes	4608
meta_llama_Llama_3.2_1B_Instruct	No	No	2048
meta_llama_Meta_Llama_3_8B_Instruct	No	No	4096
mistralai_Ministral_3_14B_Instruct_2512	Yes	No	5120
mistralai_Mistral_7B_Instruct_v0.2	No	No	4096
mistralai_Mistral_Nemo_Instruct_2407	No	No	5120
Qwen_Qwen2.5_0.5B_Instruct	No	No	896
Qwen_Qwen2.5_14B_Instruct	No	No	5120
Qwen_Qwen2.5_32B_Instruct	No	No	5120
Qwen_Qwen2.5_7B_Instruct	No	No	3584
Qwen_Qwen3_30B_A3B_Instruct_2507	No	Yes	2048
Qwen_Qwen3_4B_Instruct_2507	No	Yes	2560
swiss_ai_Apertus_8B_Instruct_2509	No	Yes	4096

Table 12. All models used in the text generation experiments.