Interoperability of ML models and systems

Week 9 – Machine Learning Systems (Fall 2026)

Yang Cao

University of Edinburgh

Recall



Week 2: data encoding (*e.g.,* recall webpages/crawl example in Week8).

Week 7: index embedding vectors.

Week 8: from RAG to memories (interactive, maintained, generalised RAG)

This week: what if encoder B replaces A?

Must we re-encode every page and reprocess the entire conversation?

What's to expect

1. Consistency between different model representations
2. From representation consistency to embedding translation
3. Impact on ML systems (retrieval)
4. Impact on LLM inference (KV Cache)

Motivation: we do we care about model change?

Encoder upgrade

- old and new embeddings cannot mix

Model deprecate:

- old models discontinued

heterogeneity

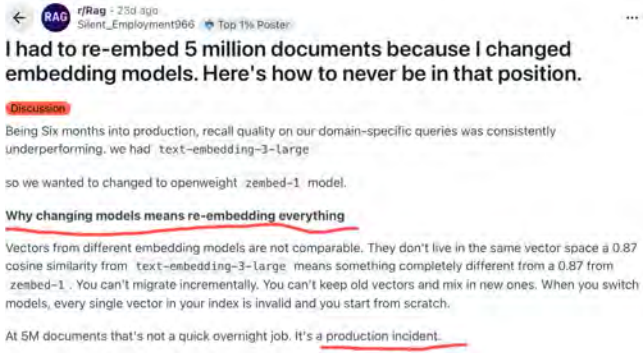
- different users use distinct encoders

Decoder hand-off

- inference-time handoff

Embeddings/states outlive models

Consequence of model change



r/Rag • 23d ago
Silent_Employment966 Top 1% Poster

I had to re-embed 5 million documents because I changed embedding models. Here's how to never be in that position.

Discussion

Being Six months into production, recall quality on our domain-specific queries was consistently underperforming. we had `text-embedding-3-large`

so we wanted to changed to openweight `zembed-1` model.

Why changing models means re-embedding everything

Vectors from different embedding models are not comparable. They don't live in the same vector space a 0.87 cosine similarity from `text-embedding-3-large` means something completely different from a 0.87 from `zembed-1` . You can't migrate incrementally. You can't keep old vectors and mix in new ones. When you switch models, every single vector in your index is invalid and you start from scratch.

At 5M documents that's not a quick overnight job. It's a production incident.

change encoder:

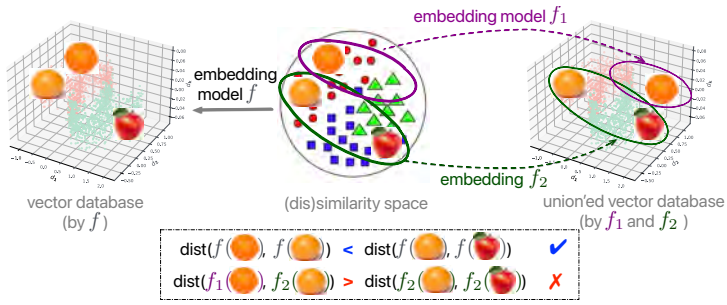
stale embeddings

embedding backfill:

re-encode every document

vector store is a silo of a model

Consequences: isolated embedding stores

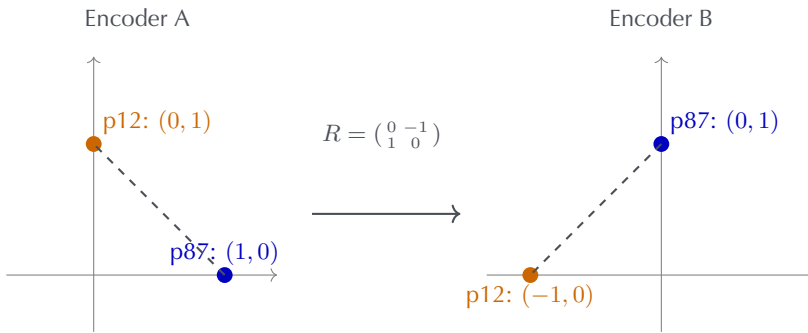


inside one model, two oranges are closer than an orange and an apple.

across two models the comparison fails

vector databases (RAG stores) are isolated data stores

. . . even if both encoders are exact



a B-query for p87 is $(0, 1)$

in the A-store it exactly matches the *wrong* page, p12.

Training does not fix this

A retrieval encoder f is trained through similarities $f(a)^\top f(b)$ only.

Take any orthogonal matrix R, so that $R^\top R = I$.

$$(Rf(a))^\top (Rf(b)) = f(a)^\top R^\top R\, f(b) = f(a)^\top f(b).$$

The encoders f and Rf reach the same loss on every batch.

Random initialisation and data order decide which one training returns.

Models map data records to high-dimensional points and training governs their geometry but not fix these points

Current practice: avoid the problem …

backfill

- embed every record again

text communication

- use text between agents (LLMs)

freeze the model

- pin a model version

KV cache

- session-only

Can we solve the problem and make use of abundant embeddings we already have?

Do models discover the same structure?

How do systems benefit from intrinsic consistency between models?

- VDBMS/RAG/retrieval systems
- KV Cache of LLMs

Comparing model representations

A *representation* is judged by the similarities it induces

Model (M) representation: maps an input x to a vector $f(x) \in \mathbb{R}^d$

Representation *kernel*: similarity that it assigns to every pair of inputs

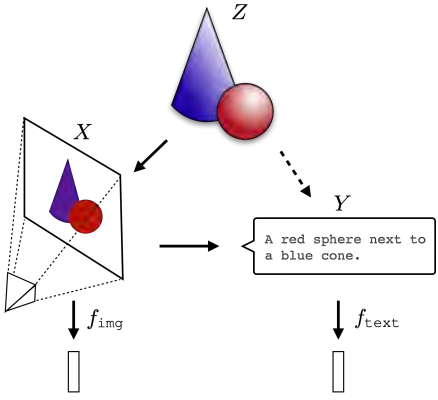
$$K(x_i, x_j) = \langle f(x_i), f(x_j) \rangle.$$

If two models have the same representation kernel, they have the same view of the ”world”

Representational alignment is a similarity score between two kernels

Huh et al., The Platonic Representation Hypothesis, ICML (2024).

The Platonic Representation Hypothesis



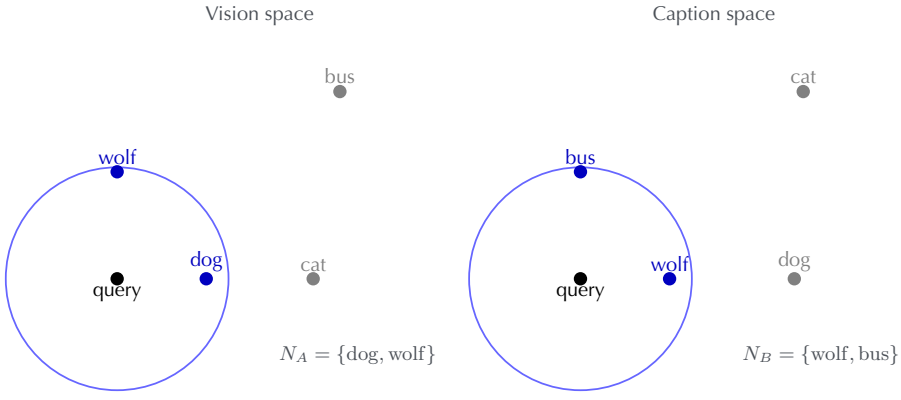
- Images X and text Y are projections of one underlying reality Z.
- Huh et al. (2024) conjecture that representation learners converge on a shared representation of Z.
- Larger models, more data and more diverse tasks drive the convergence.

A hypothesis about representation kernels

different observations can preserve shared relationships

Figure 1 of Huh et al., The Platonic Representation Hypothesis, ICML (2024)

How do we compare model representations?



For this query, the two nearest-neighbour sets share wolf: $\texttt{agreement} = |N_A \cap N_B|/2 = 1/2$.

How do we evaluate consistency between models?

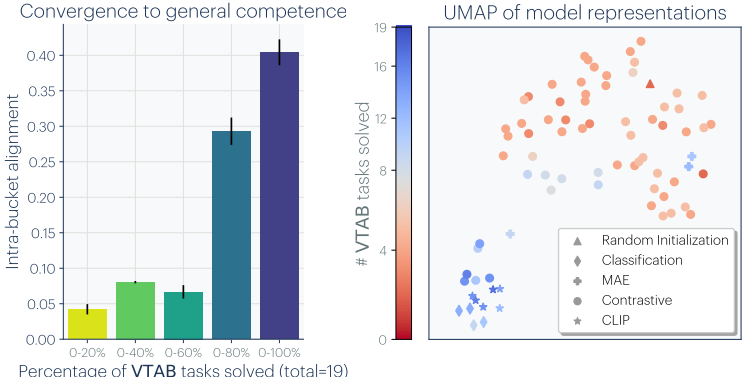


For n pairs and neighbourhood size k, the `mutual-kNN score` is

$$s = \frac{1}{n} \sum_{i=1}^{n} \frac{|N_A(i) \cap N_B(i)|}{k}$$

mutual-kNN measures shared neighbours, indicating how similar two models (kernels) are.

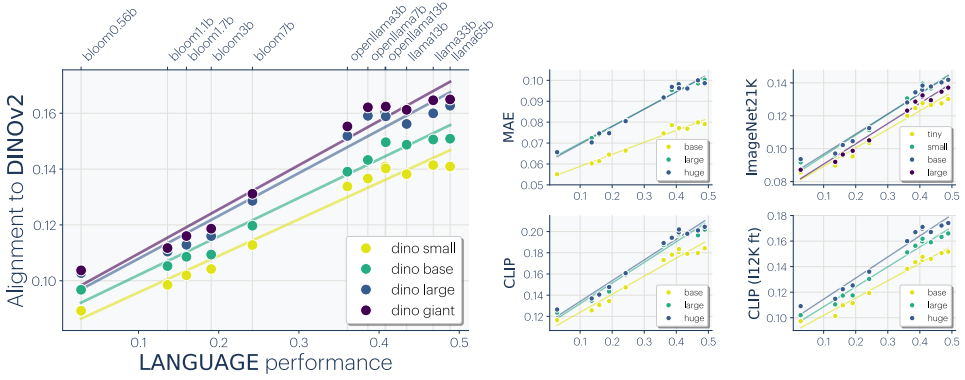
Model capabilities vs. Similarity: more capable ⇒ agree more



78 vision models are grouped by the number of VTAB tasks that they solve. Mutual-kNN alignment on Places-365 images is higher inside the more competent groups.

Figure 2 of Huh et al., The Platonic Representation Hypothesis, ICML (2024).

Cross-modality: text-only models align with image-only models



better language prediction ⇒ more shared image-caption neighbours

Figure 3 of Huh et al., The Platonic Representation Hypothesis, ICML (2024).

Is alignment 0.16 good or bad?

Comparing with ”chance”:

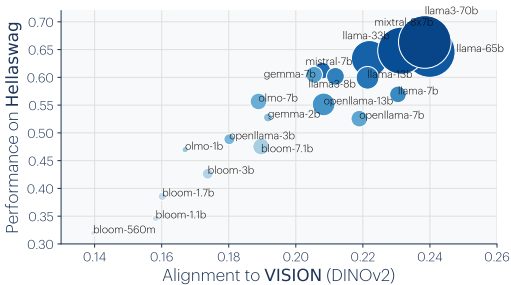
- fix one input among $n = 1024$ pairs
- each unrelated model chooses $k = 10$ neighbours from the other 1023 identifiers
- for each of A’s ten neighbours, B selects it with probability $10/1023$

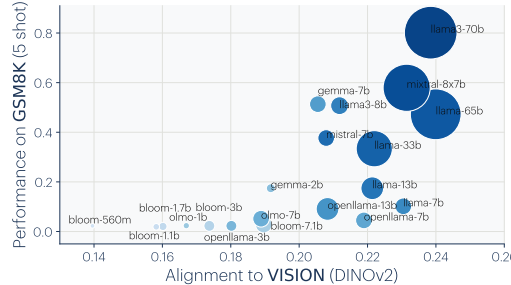
Mean overlap score for ”chance”:

- expected intersection: $100/1023$
- so chance is $100/1023/10 = 0.0098$.

Agreement score of 0.16 is above chance, though not close to one (exact consistent)

LMs: alignment with vision tracks language-task performance



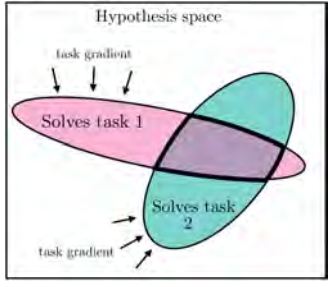


Each point is one language model. Horizontal: its alignment with the DINOv2 vision model. Vertical: its score on a language benchmark. Better-aligned language models score higher.

Figure 4 of Huh et al., The Platonic Representation Hypothesis, ICML (2024)

Why would different models converge (intuitive explanations)?

More tasks can rule out easy shortcuts



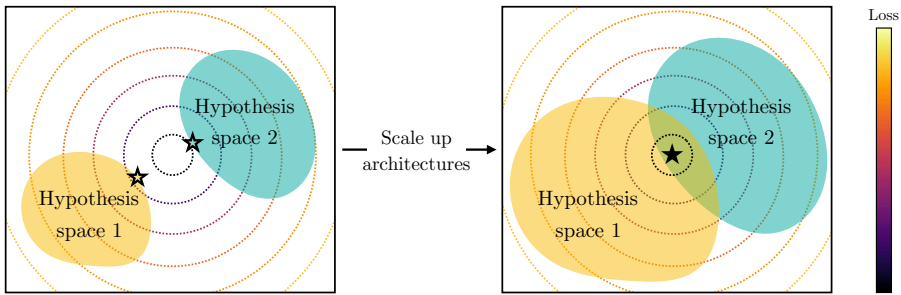
An animal classifier may ignore position; localisation cannot.

Requiring both tasks narrows the acceptable representations.

Figure 6 of Huh et al., The Platonic Representation Hypothesis, ICML (2024)

Why would different models converge (intuitive explanations)?

A larger function class is more likely to hold the optimum

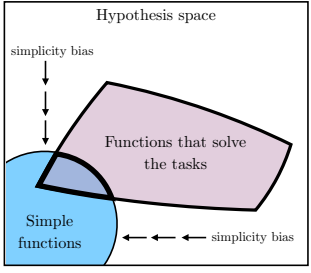


Two small function classes may each stop at a different best point. Two large classes can both reach the same global optimum, if one exists and training finds it.

Figure 5 of Huh et al., The Platonic Representation Hypothesis, ICML (2024)

Why would different models converge (intuitive explanations)?

Simplicity bias narrows the solutions further



Many functions fit the training data equally well

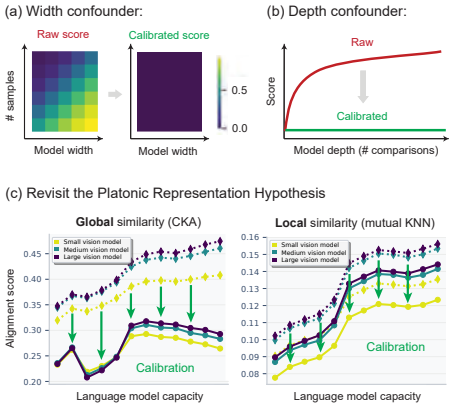
Deep networks tend to return a simple one among them

preference grows with model size (conjecture)

A shared preference selects a shared solution

Figure 7 of Huh et al., The Platonic Representation Hypothesis, ICML (2024)

Being critical: confounders with model size



- Larger models are wider. Width raises the chance level of global scores.
- Larger models are deeper. Depth offers more layer pairs to choose the best from.

Figure 2 of Groeger et al., Revisiting the Platonic Representation Hypothesis: An Aristotelian View, ICML (2026)

Centred kernel alignment for global agreement

Put the n paired embeddings in rows of X and Y. Subtract each model's mean vector.

The centred Gram matrices $K = XX^\top$ and $L = YY^\top$ contain all pairwise inner products.

Linear centred kernel alignment is their normalized overlap:

$$\mathrm{CKA}(X, Y) = \frac{\langle K, L \rangle_F}{\|K\|_F \, \|L\|_F}.$$

CKA equals one when K and L are proportional (a rotation preserves K).

Mutual-kNN needs only matching neighbour sets, and is weaker than CKA (global)

Huh et al., The Platonic Representation Hypothesis, ICML (2024).
Groeger et al., Revisiting the Platonic Representation Hypothesis: An Aristotelian View, ICML (2026)

Example: using linear CKA to compare model representations

Three inputs have one-dimensional representations

model A: $x = (-1, 0, 1)$

model B: $y = -2x = (-2, 0, 2)$

model C: $z = x^2 = (1, 0, 1)$

Evaluating similarity among models A, B, C using CKA:

Use $\langle xx^\top, yy^\top \rangle_F = (x^\top y)^2$, we compute linear CKA of A with B, and of A with C, smaller the better (more similar):

A with B: $(x^\top y)^2 = 16$ and $\|xx^\top\|_F \|yy^\top\|_F = 2 \times 8$, so CKA is 1.

A with C: centred $z = (\frac{1}{3}, -\frac{2}{3}, \frac{1}{3})$ gives $x^\top z = 0$, so CKA is 0.

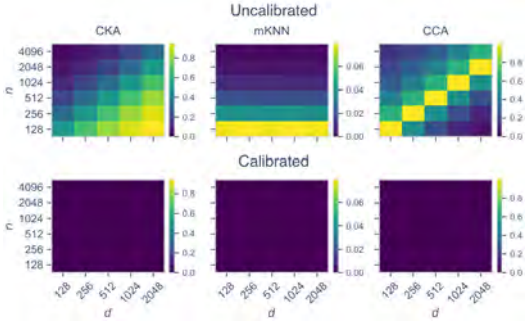
Fewer inputs or more noise can raise CKA

Continue example: consider CKA of A with C ($z = x^2$)

Inputs	Features per input (width d)	Raw CKA
$n = 3$	A: $(-1, 0, 1)$; C: $(1, 0, 1)$	0
$n = 2$	First two, centred: $(-0.5, 0.5)$ and its negative	1
$n = 3$	Append independent Gaussian noise	$\to 1$ as d grows

CKA is sensitive to data samples and dimensions

Impact of width

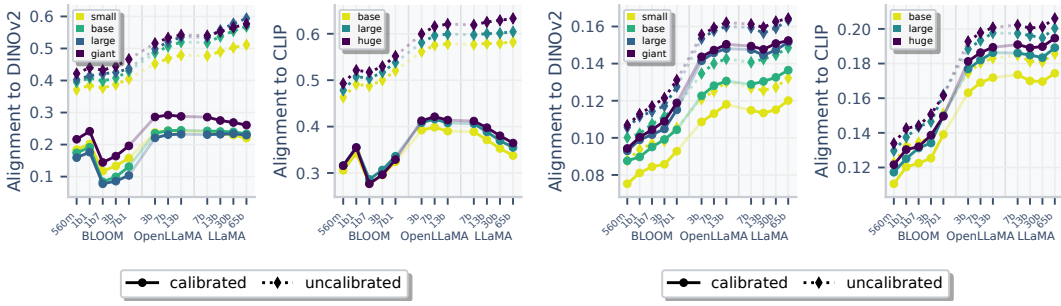


- Both representations are independent noise with n samples and width d.
- Raw CKA rises with d and falls with n.
- Raw mutual-kNN depends on n only.
- After calibration every cell is zero.

Width (dimensions d) alone can produce a high global score.

Figure 3 of Groeger et al., Revisiting the Platonic Representation Hypothesis: An Aristotelian View, ICML (2026)

A more complicated view on model convergence



(a) CKA RBF: Global spectral alignment.

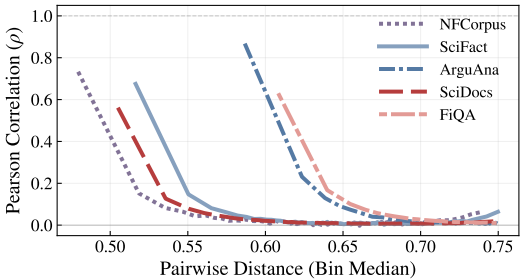
(b) mKNN: Local neighborhood overlap.

(Language models are ordered by performance on the horizontal axis.)

Model are converging; but there is no clear “scaling law”

Figure 5 of Groeger et al., Revisiting the Platonic Representation Hypothesis: An Aristotelian View, ICML (2026)

One possible explanation



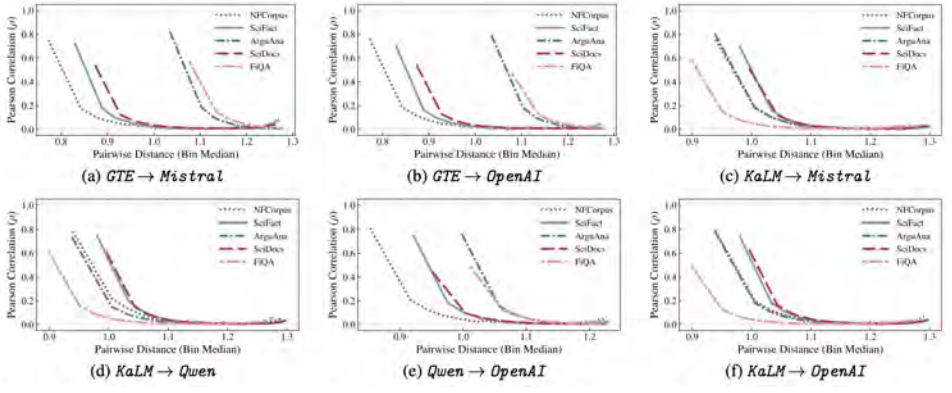
Mistral and OpenAI distances of `Arguana` group document pairs by Mistral distance

Pearson correlation within each group's Mistra and OpenAI distances:

- near 1 means the two distances rise together;
- near 0 means little linear agreement.

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

One possible explanation: contrastive learning



Short distances are more consistent across model pairs than long distances

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Local isometric consistency of contrastive encoders

Contrastive learning drives encoders toward a scaled isometry of the data manifold in the short-distance regime.

Let f_1 and f_2 be two encoders locally optimal at point $x \in \mathcal{M}$ with parameters λ_1 and λ_2, respectively. For any $y \in \mathcal{M}$ with $d_{\mathcal{M}}(x, y) < \delta_{\mathcal{M}}(x)$:

$$\|f_1(x) - f_1(y)\| = \kappa \cdot \|f_2(x) - f_2(y)\| + \mathcal{O}(d_{\mathcal{M}}(x, y)^2),$$

where $\kappa = \sqrt{\lambda_1/\lambda_2}$.

Implication: conformal translation between encoders, or equivalently, *rigid* (*i.e.,* scaled isometric) translation between encoders for short distances

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Do models discover the same structure?

How do systems benefit from intrinsic consistency between models?

- VDBMS/RAG/retrieval systems
- KV Cache of LLMs

Recall: a vector store answers by distance alone

A vector store holds $E = \{f(o) : o \in O\}$ for a set of objects O and one encoder f. A query is embedded by the same f.

top-$k(q)$ = the k vectors $e \in E$ with the smallest $\|f(q) - e\|$.

- The answer depends on distances between vectors and on nothing else.
- An index such as IVF or HNSW (Week 7) speeds up the same comparison.
- The raw objects are often stored elsewhere, or not at all.

Whoever preserves the distances preserves the answers.

Vectors of two encoders in one space?

- Backfilling

  A new encoder serves queries. Old vectors move forward without the raw data.

- Federated search

  Two organisations search both stores. Neither shares documents or models.

- Edge and cloud

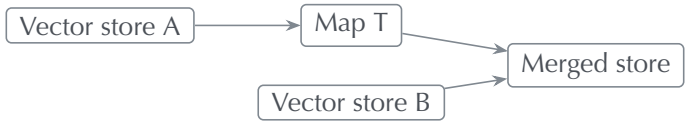
  A small on-device encoder and a large cloud encoder serve one collection.

- …

Interoperability across encoders' output is key to such scenarios

Problem: search two stores embedded by two encoders

Vector database (RAG index) integration across embedding models:



- Given: both vector sets and a few pairs of one page in both stores.
- Not given: the original pages and the two encoders.
- Wanted: a B-query finds the relevant pages of both stores.

The embedding pairs are the only bridge between the two encoder spaces

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Align two stores on their overlap, then merge them

Idea: we stitch two photographs on their shared region; the same for two vector stores?



- Stores $E_1 = f_1(O_1)$ and $E_2 = f_2(O_2)$ share some objects.
- Known pairs S hold both vectors of one object.
- Fit a map T on S. The merged store is $T(E_1) \cup E_2$.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

A2M: Align-to-Merge (just like align images)

If paired point clouds have the same distances up to one positive scale, try a map

$$T(x) = sRx + t, \qquad R^\top R = I, \quad s > 0.$$

- R: rotation or reflection
- s: uniform scaling
- t: placement/shifts.

Shared documents give vector pairs (x_i, y_i).

`A2M`: Fit the map on these pairs; test retrieval on other documents.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

How do we fit the map: Procrustes analysis

Stack the paired vectors as columns of X and Y. Centre each by its own mean.

$$\text{Objective:} \quad \min_{R^\top R = I} \|RX - Y\|_F^2$$

Orthogonality keeps $\|RX\|_F = \|X\|_F$, so

$$\|RX - Y\|_F^2 = \|X\|_F^2 + \|Y\|_F^2 - 2\,\mathrm{tr}(R\,XY^\top).$$

The best rotation maximises $\mathrm{tr}(R\,XY^\top)$.

How do we fit the map: Procrustes analysis

Take the singular value decomposition $YX^\top = U\Sigma V^\top$ and write $W = U^\top R\, V$.

$$\operatorname{tr}(R\,XY^\top) = \operatorname{tr}(W\Sigma) = \sum_i W_{ii}\,\sigma_i \le \sum_i \sigma_i$$

W is orthogonal, so $W_{ii} \le 1$. Equality holds at $W = I$:

$$R = UV^\top, \qquad s = \frac{\operatorname{tr}(\Sigma)}{\|X\|_F^2}, \qquad T(x) = sR\,(x - \bar{x}) + \bar{y}$$

A2M is no better than the plain union

Table 1: Recall@100 over integrated vector databases

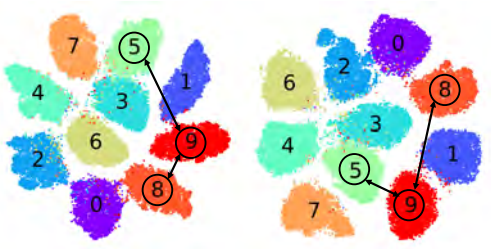
emb_1 vs emb_2	method	Dataset			
		SciFact	NFCorpus	ArguAna	SciDocs
Fast Text vs NV-embed V2	Union	26.33	13.62	30.34	10.80
	A2M	18.33	8.05	25.91	5.70
Fast Text vs Openai Ada	Union	25.33	13.93	29.55	12.20
	A2M	16.67	10.53	25.34	6.70
NV-embed vs Openai Ada	Union	32.00	14.24	28.19	16.70
	A2M	32.00	14.24	28.19	16.70
gte-Qwen2 vs NV-embed V2	Union	31.67	18.27	28.48	19.60
	A2M	32.33	15.48	28.48	18.40
Mistral vs Openai Ada	Union	32.00	15.17	28.19	17.60
	A2M	27.00	10.84	29.91	12.60

- Five encoder pairs and four BEIR benchmarks. Queries use the second encoder.
- A2M is below Union in 13 of 20 cells and equal in 5.
- Its largest gain is 1.72 points.

One global rigid map does not integrate these stores.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Diagnose: two encoders arrange the same classes differently



T-SNE visualization of CIFAR-10 embeddings using **VGGNet** (left) and **DenseNet** (right)

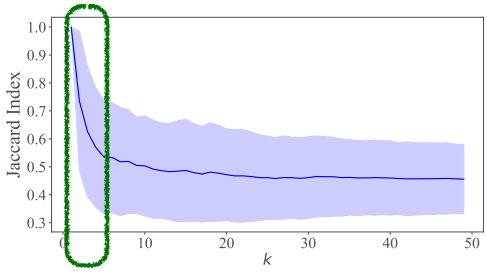
- Both encoders separate the ten CIFAR-10 classes.
- Left: class 9 lies next to class 8 and far from class 5.
- Right: class 9 lies next to class 5 and far from class 8.

The clusters agree. Their global arrangement does not.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Top-k results of two encoders agree only for small k

Local consistency



Jaccard index $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ of top-k search over SciFact with **NV-Embed-V2** and **OpenAI-Ada** embedding vectors

- One corpus, SciFact, embedded by NV-Embed-V2 and by OpenAI-Ada.
- For each query, compare the two top-k sets A and B by $|A \cap B|/|A \cup B|$.
- The mean is 1 at $k = 1$ and near 0.5 from $k = 10$ on.

Consistency between the two spaces is local.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Recall: local scaled isometric consistency

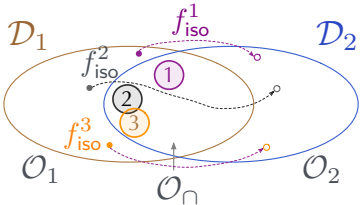
Let f_1 and f_2 be two contrastive encoders locally optimal at point $x \in \mathcal{M}$ with parameters λ_1 and λ_2, respectively. For any $y \in \mathcal{M}$ with $d_{\mathcal{M}}(x, y) < \delta_{\mathcal{M}}(x)$:

$$\|f_1(x) - f_1(y)\| = \kappa \cdot \|f_2(x) - f_2(y)\| + \mathcal{O}(d_{\mathcal{M}}(x, y)^2),$$

where $\kappa = \sqrt{\lambda_1/\lambda_2}$.

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Implication: from one map to a collection of local maps



Localized align-to-merge (LA2M)

- A2M fits one map on the whole overlap.
- LA2M splits the overlap into small regions 1, 2 and 3.
- Each region receives its own rigid map.

Rigid inside a region, flexible across regions.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

LA2M: one rigid map per neighbourhood



- Cluster the paired source vectors. Each cluster has a centroid c_i.
- Fit one Procrustes map per cluster from its own pairs.
- Send a new vector to the map of its nearest centroid.

Seamlessly integrated with vector indices: IVF cells (Week 7) now carry the map

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

From A2M to LA2M

Table 1: Recall@100 over integrated vector databases

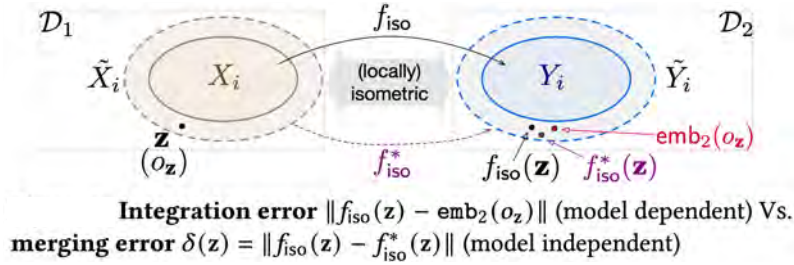
emb_1 vs emb_2	method	Dataset			
		SciFact	NFCorpus	ArguAna	SciDocs
Fast Text vs NV-embed V2	Union	26.33	13.62	30.34	10.80
	A2M	18.33	8.05	25.91	5.70
	LA2M	76.67	34.06	64.66	39.30
Fast Text vs Openai Ada	Union	25.33	13.93	29.55	12.20
	A2M	16.67	10.53	25.34	6.70
	LA2M	67.00	28.63	68.24	39.30
NV-embed vs Openai Ada	Union	32.00	14.24	28.19	16.70
	A2M	32.00	14.24	28.19	16.70
	LA2M	79.33	35.29	85.22	41.10
gte-Qwen2 vs NV-embed V2	Union	31.67	18.27	28.48	19.60
	A2M	32.33	15.48	28.48	18.40
	LA2M	85.33	31.27	86.65	44.60
Mistral vs Openai Ada	Union	32.00	15.17	28.19	17.60
	A2M	27.00	10.84	29.91	12.60
	LA2M	76.67	34.06	79.25	42.00

- same setup as before.
- LA2M tops in all cases
- improvement is evident: SciFact rises from about 30 to between 67 and 85.

LA2M: the map family stayed the same as A2M but only changes its scope

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Bounding the error of each local map

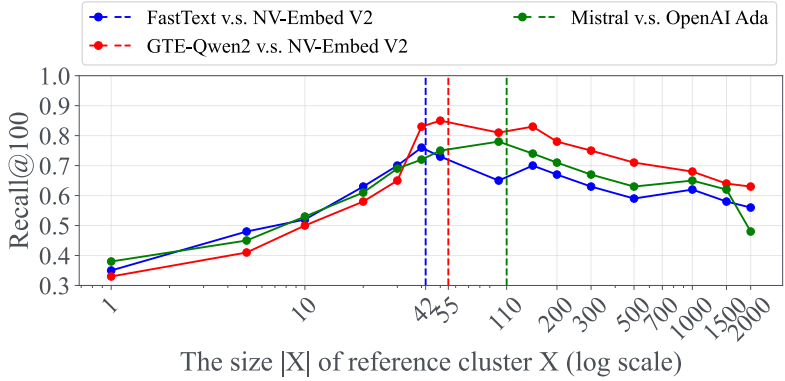


Integration error needs the unknown vector $f_2(o_z)$.

Merging error $\delta(z)$ compares the fitted map with the ideal rigid map of the enlarged region.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

How local is local for LA2M?

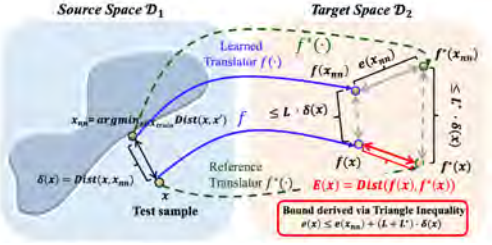


Recall@100 against the number of pairs per region, for three encoder pairs.

Dashed lines mark estimated intrinsic dimensions of 42, 55 and 110.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

From merging bound to integration bound

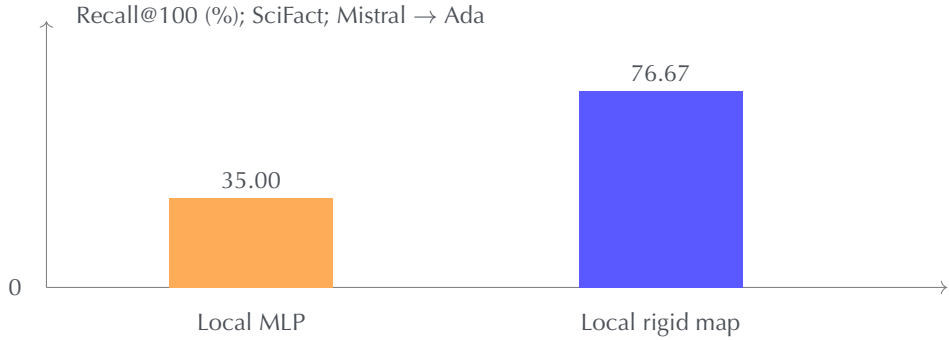


- Test vector x lies $\delta(x)$ from its nearest training vector x_{nn}.
- Training term: the error $e(x_{nn})$ at that neighbour.
- Coverage term: $(L + L^*)\,\delta(x)$.

A flexible map can lower the first term and raise the second.

Yang and Cao, Generalizable and Composable Multi-Model Embedding Translation, ICML (2026)

More flexible maps are not automatically better



For Mistral-to-Ada on SciFact, local rigid maps reach 76.67 recall; local MLPs reach 35.00. Extra flexibility alone does not explain LA2M.

Yang et al., Integrating vector databases across embedding models, SIGMOD (2026).

Where do the pairs come from?

Cross-encoder integration (LA2M) needs paired vectors, which may not be available

- The overlap $O_1 \cap O_2$ is unknown.
- Inside the overlap, no identifier says which x matches which y.
- A handful of pairs may be known, for instance 15 pairs for 57,638 documents.

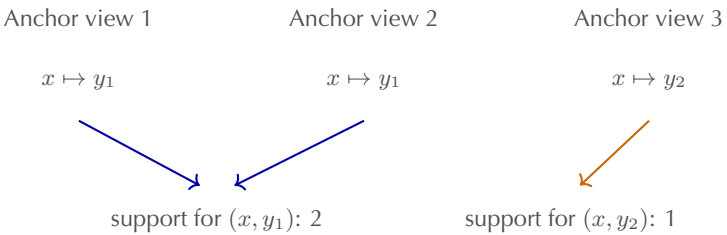
Vector linking:

- identify pairs of vectors across encoders that represent the same object/entity
- given only the vector sets (and a few seed pairs), no access to datasets or encoders

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Anchor signatures: distance-to-anchors hashing + voting

Distance-to-anchors as stable signatures of objects across encoders

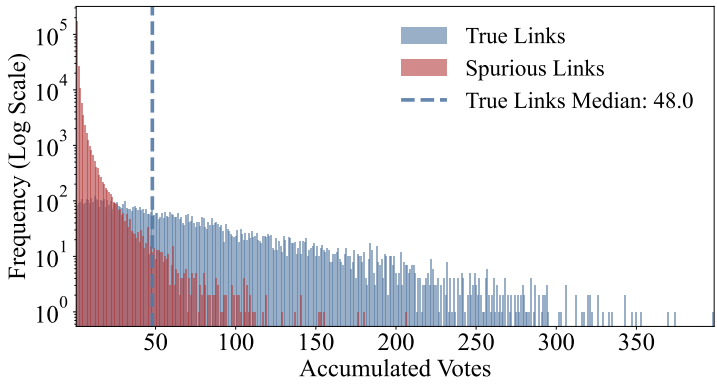


Different anchor subsets give different nearest-signature proposals

Count repeated support for each candidate pair; voting for true links

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Voting for true links



ArguAna, GTE/OpenAI: 500 fixed anchors, 500 views of 30 anchors. False links mostly get few votes; true links have median support 48.

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

Effectiveness

Fifteen seed pairs recover most links on five benchmarks

Table 1. **Vector linking at overlap α=0.3 and seeds $|\mathcal{S}|$=15:** each cell shows **precision/recall/F1(%); bold** indicates the best value.

Method	Qwen-OpenAI NFCorpus	GTE-OpenAI SciFact	GTE-Mistral ArguAna	Mistral-OpenAI SciDocs	Qwen-Kalm FiQA
Linear	2.8/7.2/4.0	2.2/5.0/3.1	0.4/0.3/0.3	2.2/1.7/1.9	0.7/0.8/0.8
CCA	46.7/10.6/17.3	29.3/5.2/8.8	25.2/3.9/6.8	11.0/1.4/2.4	12.5/0.7/1.4
MLP	36.1/3.3/6.0	14.2/1.4/2.5	11.0/0.7/1.2	10.4/0.3/0.5	9.2/0.1/0.3
RCSLS	38.9/3.0/5.6	28.6/2.1/3.8	26.2/0.2/0.4	32.6/0.3/0.6	15.8/0.2/0.4
Proc	52.5/11.8/19.3	37.9/5.7/9.9	30.8/4.8/8.4	15.6/1.6/2.9	20.6/1.3/2.4
AO	22.5/22.5/22.5	5.6/5.6/5.6	0.6/0.6/0.6	0.2/0.2/0.2	0.1/0.1/0.1
GEH	**82.1/95.6/88.3**	**83.2/89.1/86.0**	**77.1/84.5/80.7**	**82.8/81.6/82.2**	**79.8/79.9/79.8**

- Setting: overlap ratio 0.3 and 15 seed pairs.
- Cells: precision, recall and F1 of the recovered links, in percent.
- Procrustes: F1 of 2.4 to 19.3. GEH: 79.8 to 88.3.

Anchor signature (relative representation) is effective when used with iterative voting

Chen et al., Vector Linking via Cross-Model Local Isometric Consistency, ICML (2026).

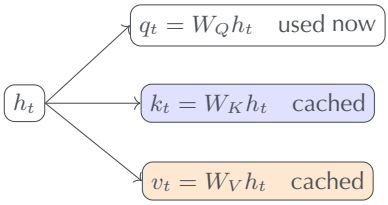
Do models discover the same structure?

How do systems benefit from intrinsic consistency between models?}

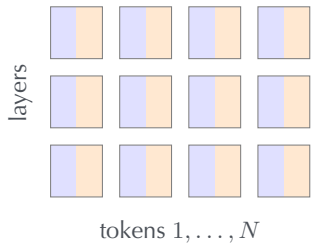
- VDBMS/RAG/retrieval systems
- **KV Cache of LLMs**

Recall: KV cache of LLMs

One layer, one head, token t



KV cache: layers × tokens



Each layer token t has a state h_t. The layer derives a query, a key k_t and a value v_t. Later tokens read k_t and v_t again, so the server keeps them. Prefill fills the grid for the whole prompt; decoding appends one column per new token.

What if we switch the model?

Every model switch triggers a full prefill:

- Cascade: a small model answers easy turns. A hard turn goes to a large model.
- Agents: several fine-tuned agents read one shared conversation history.
- Model update: a new version of a model takes over running sessions.
- Hand-back: a large model returns routine work to a cheap one.

Receiver model processes accumulated context again, although another model already did

Why process the same history twice?

Baseline: send the history as text



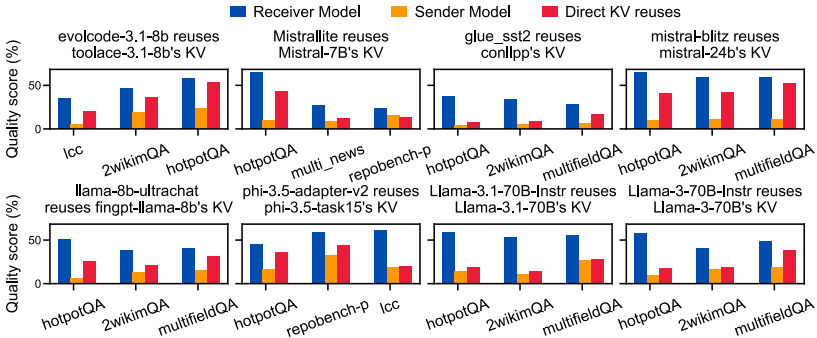
Reuse: S has already processed the history



S's cache already exists. A text hand-off repeats L's prefill; the translated-cache route avoids that pass. L must still decode the answer.

Mechanism adapted from Castin et al., KV-Lingo, arXiv:2609.32610 (2026), Figure 1.

Copying a cache fails even between fine-tuned variants



Eight model pairs of the same architecture. Blue: receiver with its own cache. Orange: sender alone. Red: receiver with the sender's cache copied in.

Figure 6 of Liu et al., DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving, ICLR (2026)

DroidSpeak profiles one foreign layer at a time

Hold five illustrated cache layers native; replace only layer 4



Decode the task; compare F1 with native target prefill

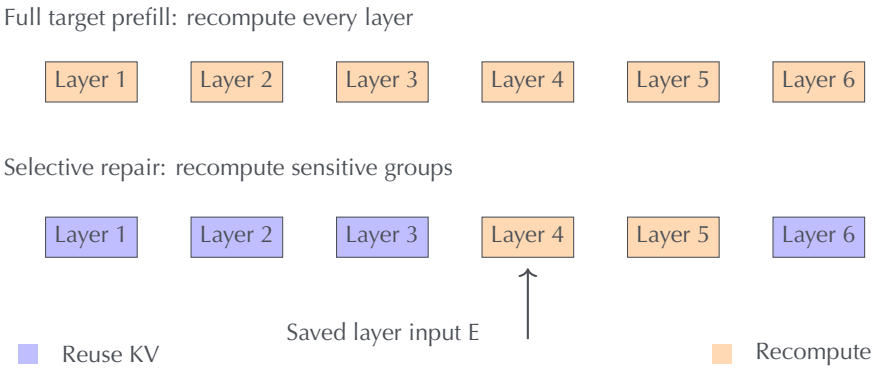


11% critical layers on average across eight model pairs

Replace one cache layer while keeping the others native. Measure the receiver's quality loss, then repeat. Only 11% of layers are critical on average in this study.

Liu et al., DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving, ICLR (2026)

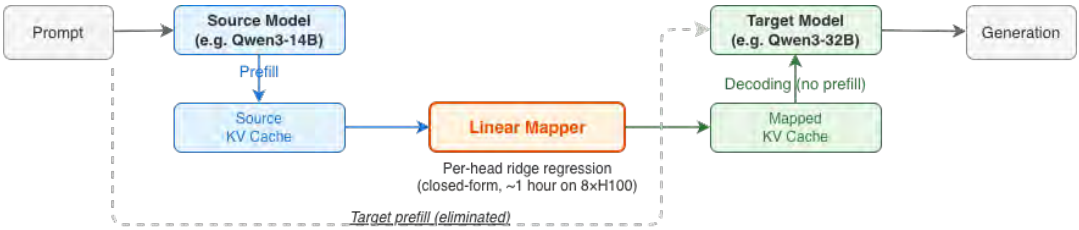
Recompute sensitive layers, reuse the remaining layers



Six illustrative layers: reuse blue KV layers and recompute the orange group. Saved layer input E restarts the target computation. Loading and computation can overlap.

Liu et al., DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving, ICLR (2026)

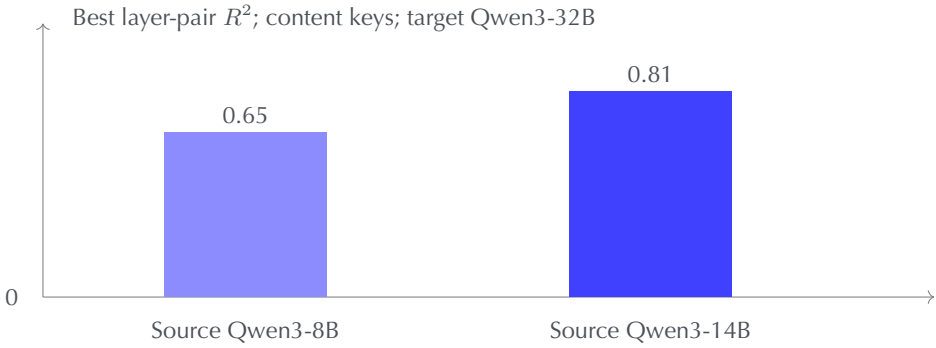
NVIDIA: linear map translation



The source model prefills the prompt. A linear mapper converts its cache into the format that the target expects. The target decodes from the mapped cache and never prefills.

Figure 1 of Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

Predictable KV cache structure

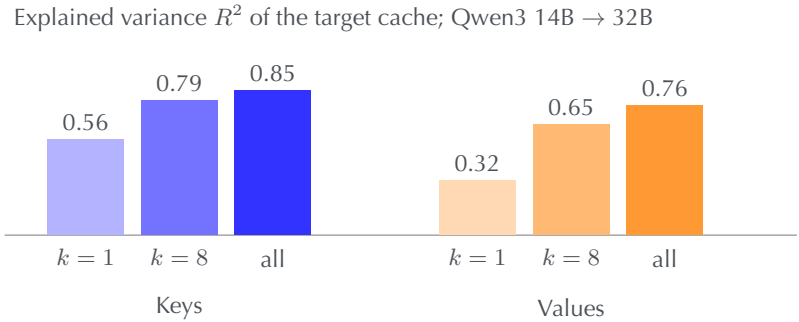


Best single-layer fits on RoPE-stripped keys. R^2 measures explained variance.

The best single source/target layer pairs predict substantial content-key structure.

Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

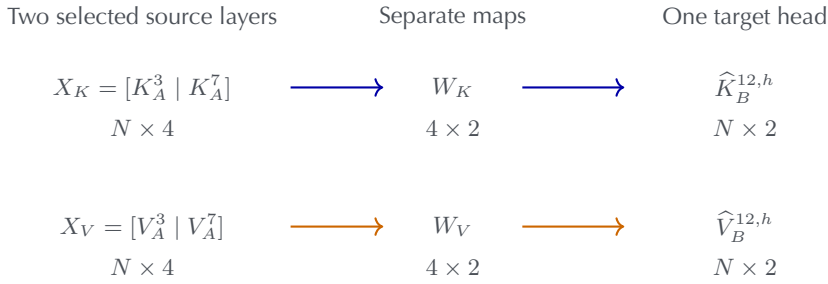
More source layers can help aligning with the target layer



One source layer explains 56% (resp. 32%) of key (resp. value) variance; eight layers explain 79% and 65%. So using the k best source layers for each target layer helps.

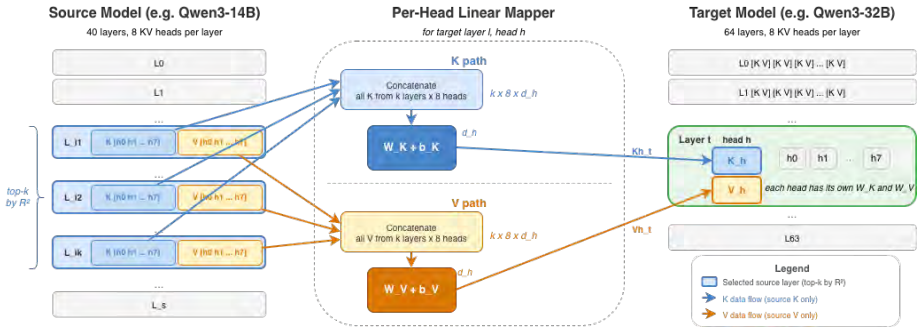
Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

Build each target head from several source layers



Illustratively, each selected layer supplies two features per token. Concatenation gives $N \times 4$ inputs; separate maps produce $N \times 2$ keys and values for one target head.

KV Cache translation: separate K and V maps per head

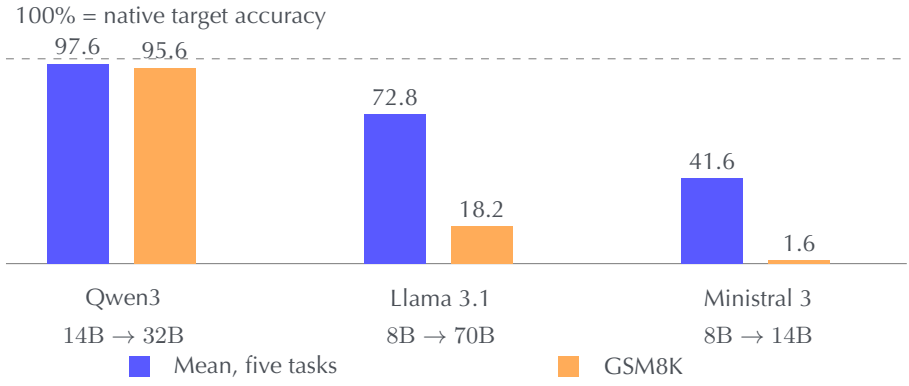


40 source layers and 64 target layers, each with 8 KV heads.

For target layer l and head h, the keys of the k selected source layers are concatenated and mapped by W_K. Values take a separate path through W_V.

Figure 3 of Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

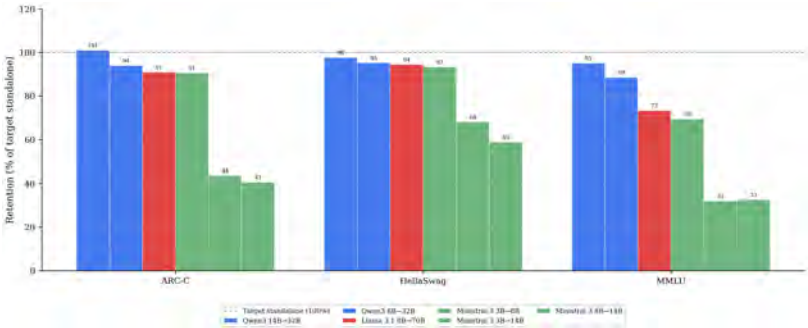
Effectiveness



Retention is a ratio to native-target accuracy. Mean retention hides generation failure: Llama 8B-to-70B keeps 72.8% on average, but only 18.2% on GSM8K.

Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

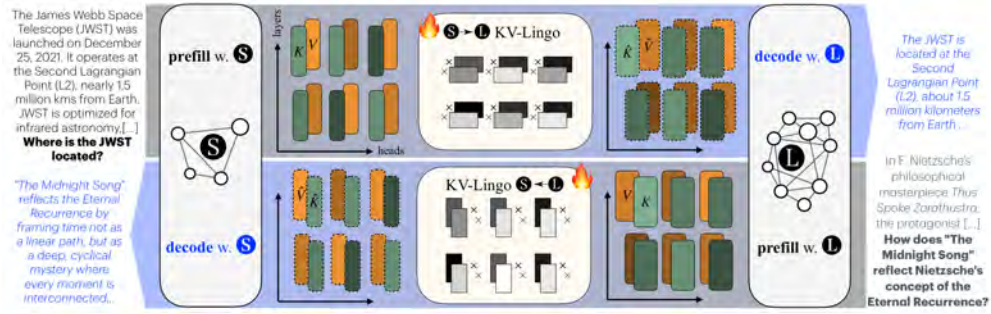
Effectiveness



Retention is the transferred accuracy as a percentage of the target's own accuracy.

Heo et al., Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse, arXiv:2608.03893 (2026)

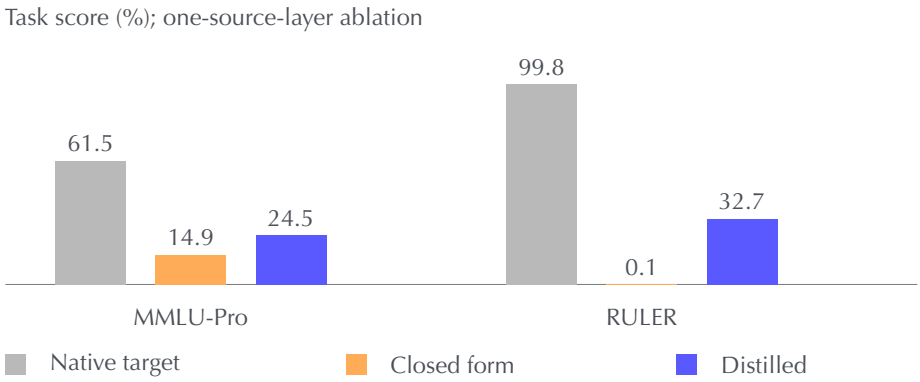
KV-Lingo: small-large model hand-off via KV translation



Top: the small model S prefills a passage and its question, and the large model L answers from the translated cache. Bottom: the reverse direction with a second translator.

Figure 1 of Castin et al., KV-Lingo: Learning KV-Cache Translators with Distillation, arXiv:2609.32610 (2026)

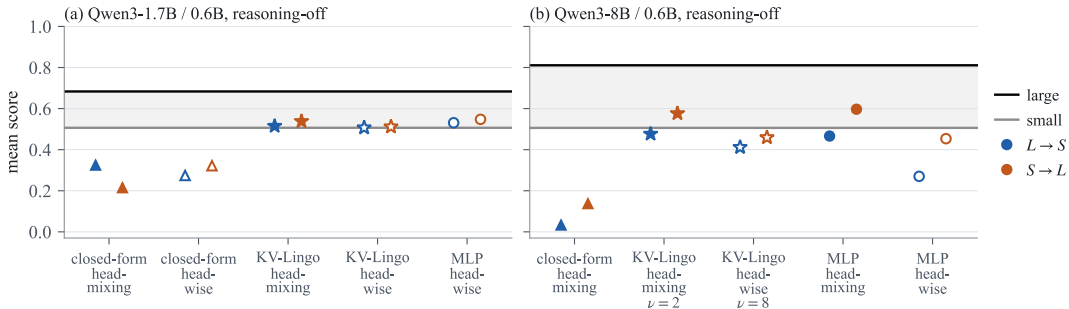
Distillation improves the map, but inspect the gap



Qwen3-0.6B to 8B, reasoning off, one selected source layer. Distillation repairs part of the loss, not all of it. This is an ablation, not the best translator.

Castin et al., KV-Lingo: Learning KV-Cache Translators with Distillation, arXiv:2609.32610 (2026)

Effectiveness



Black line: the large model alone. Grey line: the small model alone. Triangles are closed-form maps, stars are distilled KV-Lingo maps and circles are MLP translators.

Figure 3 of Castin et al., KV-Lingo: Learning KV-Cache Translators with Distillation, arXiv:2609.32610 (2026)