CROSS-MODEL STITCHING TRANSFERS ANSWERS, NOT COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation stitching describes the technique of grafting one model's hidden state into another's through a learned map. Its motivation is related to universality, under which models may learn shared representations that enable transfer. Prior work demonstrates that features, probes, and steering vectors can survive such transfers. However, what actually moves through the stitch, a question with substantial implications for universality, remains unresolved. We examine this question by grafting a large donor's residual state into a smaller recipient using a map trained with cross-entropy on the gold output label. We spread this evaluation across three same-family model pairs (Gemma-2 9B→2B, Qwen2.5 7B→0.5B, Llama-3.1 8B→3.2 1B) and two cross-family pairs (Qwen2.5 7B and Llama-3.1 8B into Gemma-2 2B). We evaluate across five task families: arithmetic, symbolic binding, state tracking, naturalistic factual recall, and GSM8K. Our experiments find that this task-supervised stitch is a working channel that confers the donor's answer on up to 88.6% of otherwise unsolvable problems in our primary arithmetic evaluation. Additionally, a battery of controls indicates that conferral is achieved through the transfer of a precomputed answer, and success is dictated primarily by the answer's presence at the donor stitch site. We also find that this presence alone is not sufficient, as it must then be lossily encoded into the recipient. Further ablations indicate that the task-supervised map learns to mitigate this effect by writing the signal redundantly across hundreds of dimensions of the graft, even though the signal itself is low-rank. Finally, while the first answer token transfers exceptionally well from the prefill state, the rest of the answer does not until prior answer tokens have been emitted. Transfer is bottlenecked by answer presence, and the donor must undergo additional serial computation before that presence is sufficient for transfer. Together, these results suggest a boundary in cross-model universality, where models appear sufficiently aligned for precomputed answers to transfer through a stitch, while showing no measurable evidence that the computations producing those answers transfer with them.

1 INTRODUCTION

Recent work makes cross-model activation stitching look like a surprisingly practical way to transfer representations. Features, probes, and steering vectors can be translated between models, and hidden states can be read out by patching them into new contexts (Chen et al., 2025; Oozeer et al., 2025; Ghandeharioun et al., 2024). What these results do not settle is what a stitch actually transfers, which has implications for universality and where internal representations align best between models (Olah et al., 2020; Chughtai et al., 2023). Work on cross-model transfer validates alignment through downstream performance, which does not distinguish computational transfer from the transfer of precomputed products. Our grafting experiments separate these possibilities causally.

Using a task-supervised linear stitch at a single layer pair, we graft a donor residual state onto a smaller recipient across five task families, including few-step arithmetic, symbolic binding, state tracking, naturalistic factual recall, and GSM8K. We evaluate these tasks by stitching larger models into same-family smaller recipients (Gemma-2 9B→2B, Qwen2.5 7B→0.5B, and Llama-3.1 8B→3.2 1B), with additional cross-family (Qwen and Llama donors into a common Gemma recipient) and instruction-tuned arms. We evaluate only on problems in the *unsolvable bin*, which is the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

subset of problems the recipient could not solve but the donor could, ensuring every correct answer is conferred from the donor.

First-token conferral for the same-family pairs on this bin reaches 88.6% for Gemma, 79.4% for Qwen, and 27.8% for Llama, far above the shuffle controls of Fig. 2. The Llama pair scores lowest because packing multi-digit answers into a single token forces each map to predict hundreds of answer classes rather than nine. This leads to undertraining (as all pairs are given the same amount of training data), and expanding the training data six-fold raises its conferral by a factor of 2.7 (§A.18). These results hold across model families, with Qwen2.5 7B and Llama-3.1 8B donors both lifting a Gemma-2 2B recipient's leading-digit accuracy (§8). Our symbolic binding and state tracking tasks behave the same way, with multi-digit computation and GSM8K bounding transfer to what the donor has already computed.

We define a transfer of computation by the stitch's ability to transfer something beyond a precomputed answer at the donor's graft site. A recipient that uses the stitched layer to compute answers or digits beyond what the donor had in its graft site would satisfy this definition. We call it a transfer of an answer when the output is fixed by the donor's precomputed content and the recipient only reads it out. We estimate this answer content with a linear probe, which is conservative as it can only track linear presence (§9). Our experimental suite consistently reinforces the idea that once the answer is stitched, nothing else in the recipient does measurable work on that stitched content beyond decoding the answer. Across our evaluations, improvement beyond the recipient's native computation can be traced back to content produced by the donor prior to its stitch site, a result which holds as we isolate that content, erase it, strengthen the map, and widen the graft to every prompt position (§5, §6, §7, §A.16). These results suggest a substantial boundary in cross-model universality: answer encodings are sufficiently aligned to transfer between models with relative ease, while the computations that produce those answers appear substantially less aligned, even under this generous mapping regime.

We synthesize our findings into four contributions:

1. We find that a task-supervised linear stitch is a viable channel that confers donor answers that the recipient cannot produce natively. This was reproduced across five model pairs (including two cross-family pairs) and four of the five task families.
2. We find that stronger maps and objectives do not close the full-answer gap, with a whole-answer objective roughly doubling full-answer conferral and still leaving it far below the donor's, yet a single free vector is able to drive the recipient to the full answer.
3. We find that an encoded version of the donor's precomputed answer is causally responsible for transfer.
4. We separate two often-conflated notions of universality by finding answer representations to be sufficiently aligned to transfer reliably, while intermediate computational states show little evidence of being aligned enough for useful continuation.

2 Related Work

Early work underlying model stitching comes from vision (Lenc & Vedaldi, 2015; Bansal et al., 2021). More recent work shows that learned features and activation-space interventions can transfer across models (Chen et al., 2025; Oozeer et al., 2025), while Patchscopes demonstrates that hidden states can be decoded through another model (Ghandeharioun et al., 2024). Linear probes and decoders can reveal content from hidden states (Alain & Bengio, 2016; Belrose et al., 2023). Successful task-supervised stitching does not necessarily imply similarity (Balogh & Jelasity, 2025) and can lead to off-manifold states (Smith et al., 2025). These works show that representations can be mapped, not what is transferred.

Closest to our setup, Ramesh & Li (2025) graft a donor's last-token activation into a recipient using a reconstruction map analogous to our control. StateBridge and a universal activation interface demonstrate cross-model hidden-state alignment without our task-supervised single-site overwrite (Peng et al., 2026; Kim et al., 2026). Cache-to-Cache (Fu et al., 2026) utilizes task supervision, but for layerwise KV-cache fusion rather than a single residual-state overwrite at one layer and one token position. All test the usefulness of cross-model transfer, but do not investigate what exactly the

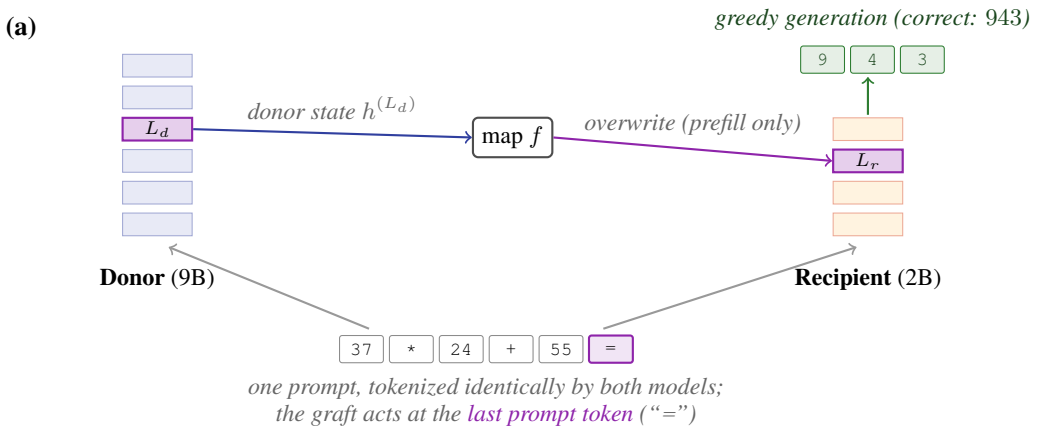

Figure 1: A worked example of our stitching procedure on a sample problem ($37 \times 24 + 55$). Both models read the same prompt. The donor's residual-stream state $h^{(L_d)}$ at the last prompt token is transferred through a learned map f and overwrites the recipient's activations at layer L_r during prefill. The recipient then generates greedily (here, the correct 943).

transferred state contains. Most report full-benchmark accuracy. We instead score only problems the donor answers and the recipient does not. An audit of such systems finds example-specific transfer ranging from ceiling to undetectable (Cheng et al., 2026), which benchmark accuracy does not distinguish.

A recent negative result reports that grafting one model's activations into another with a reconstruction-style objective does not improve multi-hop reasoning in Pythia models (Zhang & Xin, 2026), while cross-architecture activation injection has shown mixed results across model pairs (Piepereit, 2026).

Additional work shows how well a given hidden state can predict subsequent tokens (Pal et al., 2023), and recent analyses identify the mechanisms models use for arithmetic tasks (Nikankin et al., 2025; Kantamneni & Tegmark, 2025). Separate work finds that vectors representing a specific task (function vectors) (Todd et al., 2024) or task vectors induced by in-context demonstrations (Hendel et al., 2023) can be injected into a model to make it perform the task. We show that a vector translated between models instead carries the answer itself. Work on representation similarity shows that latent spaces can be made comparable (Moschella et al., 2023), while broader work finds shared geometry and features across models (Huh et al., 2024; Lan et al., 2024). These works establish cross-model alignment, but do not isolate whether it reflects already-computed content or an encoding that supports continued computation.

3 Setup and Preliminaries

3.1 The stitch and its maps.

We evaluate transfer through a stitch that grafts a large donor model's activations into a smaller recipient. Figure 1 demonstrates the stitching process, where the donor's residual state at layer L_d and at the final prompt token is passed through a map f and used to overwrite the recipient's original activations at layer L_r during prefill. The recipient is then allowed to continue its forward pass greedily. While our three primary pairs share a tokenizer, allowing for positional grafting, our cross-family pairs do not, forcing us to tokenize independently and score on decoded answer text.

The *task map* is a linear map trained with cross-entropy on the ground-truth first-answer token, and is our primary map. It is initialized from the reconstruction map and trained at full rank. Both models stay frozen throughout. The *reconstructive map* is our baseline, trained on the mean-squared error between the stitched donor activations and the recipient's original activations. The remaining maps test whether conferral is limited by a suboptimal map or objective, and the *free vector*, an unconstrained vector optimized per problem at the graft site with access to the gold answer (§5),

bounds what any single-site injection could achieve. Table 2 lists the pairs, evaluated in bfloat16, with their layer pairs. The recipient layer is set per model by arithmetic activation patching (Meng et al., 2022), using the elbow where the swap percentage rises sharply and then begins to plateau. We then pick the donor layer based on a combination of centered kernel alignment (Kornblith et al., 2019) with L_r, how much accuracy patching that layer restores, and how reliably a linear probe reads the answer off it, which improves toward later layers (§A.4).

3.2 Tasks and evaluation.

We primarily test on basic arithmetic of the form $a \cdot b + c$, which consistently gives a reasonable number of problems in the unsolvable bin. We additionally evaluate model pairs on symbolic binding, where the model resolves a chain of up to 14 alias definitions to a color label, and state tracking, where up to 10 swaps are applied to containers holding different colors, as non-arithmetic supplements. We also run depth-controlled arithmetic, which establishes a relationship between task depth and model performance (Table 8), and GSM8K (Cobbe et al., 2021) (full $1{,}319$-problem test set), whose answers are generated token by token rather than in a single forward pass.

Our primary metrics include first-token and full-answer conferral on the unsolvable bin, which is filtered on native first-token correctness for same-family pairs and full-answer correctness on our cross-family pairs (since they do not share a tokenizer). For the cross-family pairs, since the recipient's native leading digit accuracy is nonzero, its conferral must be read against its native floor (0.54 with the Qwen donor, 0.45 with Llama). Additionally, Llama-3.1's tokenizer packs three-digit sequences into a single token, meaning we must measure answer presence with a full-answer probe rather than a first-token one (which is used for Gemma and Qwen). Our train/eval splits are $3{,}000/2{,}000$ for arithmetic and $500/350$ for symbolic binding and state tracking ($250/220$ at the deepest). We report Wilson 95% intervals, bootstrap intervals, and t-intervals across five seeds. All decoding is greedy. Objective definitions and prompt templates are given in §A.21.

4 The Stitch Is a Working Channel

Our task map conferred the correct first token on 88.6% of problems for Gemma, 79.4% for Qwen, and 27.8% for Llama (Fig. 2; intervals and supplementary controls in Table 4). This result holds when looking at all problems the donor solves, as all three recipients report an increase on this set (30%, 18%, and 25% on Gemma, Qwen and Llama).

Four controls validate this conferral. First, our reconstructive stitch, despite maintaining a high reconstruction cosine of ≈ 0.98, confers only 0.191. An orthogonal Procrustes map, which is better able to conserve the donor layer's original geometry, still confers only 0.441 (§A.17). Conferral on our shuffled control, which feeds the task map an unrelated problem's donor state, falls to 0.102, confirming that problem-specific content is transferred. And, finally, grafting the recipient's own state leaves accuracy at zero.

Two additional characteristics of task map conferral are worth noting. First, the grafted vector (L_r) has a reconstruction cosine of only 0.22, meaning the task map did not learn to construct something in the natural manifold of vectors the recipient would normally produce. We return to what this displacement reflects in §6. Second, although first-token conferral is high, full-answer conferral is much lower (only 0.13 for Gemma). This result still outperforms native (0.00) and reconstructive (0.03) full-answer conferral, but it raises the question of why the stitch cannot deliver the full computed answer.

5 The Gap Between First-Token and Full-Answer Conferral

Even with stronger maps, full-answer conferral remains well below the donor's full-answer solve rate. Our two primary results (run on the Gemma same-family pair) are stitching with a nonlinear MLP task map, which leaves first-token conferral unchanged (0.89) and lifts full-answer conferral from 0.13 to 0.17, and a linear map trained on the full-answer objective, which lifts it further to 0.23. A combination of the two confers at 0.25, nearly doubling the linear task map rate. While these results are an improvement, they remain well below the donor's rate on this bin (0.42; §7). Widening

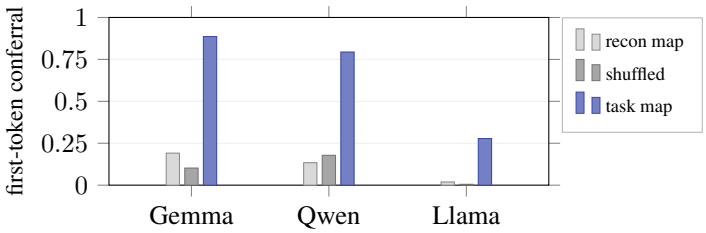

Figure 2: First-token conferral on the unsolvable bin, where native and self-graft accuracy are 0.000 by construction. The task map clears both controls in every pair.

Table 1: Full-answer conferral on the Gemma unsolvable bin. Every map sits far below the donor's own completion rate. Intervals in Appendix A.3.

Condition	full-answer conferral
Native recipient	0.00
Linear task map	0.13
MLP map	0.17
Oracle map (fit on eval, to convergence)	0.12
Linear map, whole-answer objective	0.23
MLP map, whole-answer objective	0.25
Donor, own generation (reference)	*0.42*
Free vector (no donor, answer access)	**1.00**

the graft from one prompt position to all and stitching through a single shared map does not raise full-answer conferral in either Gemma or Qwen, whose answers span several tokens (§A.16).

The most interesting result of this suite comes from the linear task map trained on first-token CE, where we find that, given the recipient outputs the correct first answer token, its full-answer accuracy is roughly equivalent whether it received a grafted vector or not (0.108 when it receives a grafted vector and 0.127 when it does not across 1,010 problems). If computation were transferred through the task map, it should provide some latent full-answer benefit even without explicit training. Thus, the learned map appears to have found little to no consistent computational encoding that the recipient can expand upon downstream, suggesting that the models' computational representations are not sufficiently aligned for useful continuation through this interface.

Additionally, we rule out the idea that a single-layer intervention cannot move full-answer accuracy. Optimizing an unconstrained vector at the recipient graft site per problem, with access to the gold answer, produces the correct answer on 100% of a 200-problem subsample of the Gemma unsolvable bin (0.94 and 0.89 on the same subsample size for Qwen and Llama). The bottleneck is therefore the donor's state at the graft site (Table 1).

6 What Crosses Is Decodable Content

6.1 The stitch writes the answer in, and it is causally responsible

In the Gemma same-family pair, fitting a linear probe to the grafted vector for the first output token yields a decodability of 0.82, which is the fraction of gold first tokens that are accurately predicted by this vector alone. This is within noise of the same probe fitted to the donor graft site L_d (0.78), and well above the recon map's output and the recipient's native state which carry it at 0.58 and 0.53 (the recipient answers every problem in this bin incorrectly, so 0.53 is what a probe trained on labelled data can extract from its state, not a readout the recipient itself has).

While the probe result suggests that the task map's encoded vector may make the answer more pronounced, it does not prove that only an encoded answer transfers through the stitch. Therefore, to quantify how much conferral is explained by the donor's precomputed content, we fit a nine-class digit probe to the donor stitch site, freeze it, and train a writer to build the grafted vector from just

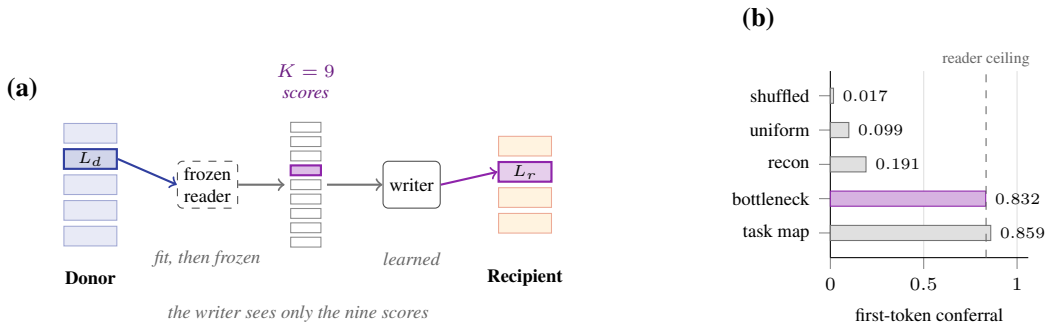

Figure 3: **(a)** The donor state is replaced by the softmax scores from the frozen linear answer reader, ensuring that nothing but the answer can cross into the recipient. **(b)** First-token conferral on the Gemma unsolvable bin ($n = 638$). This setup confers 0.832, 96.9% of a task map trained on matched data (0.859). The probe itself is only 0.834 accurate on this bin (dashed line), so the writer delivers the answer nearly perfectly each time (§A.9).

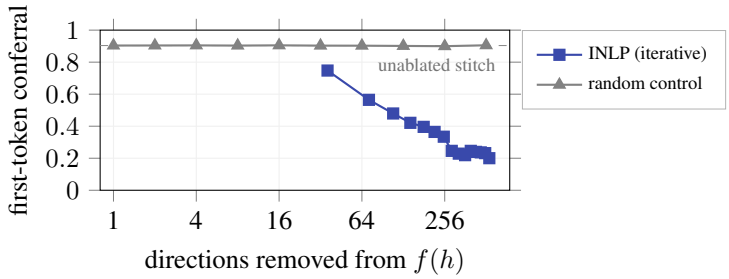

Figure 4: First-token conferral on the Gemma unsolvable bin as directions are removed from the graft. Iterative INLP, whose directions are found by fitting an answer probe, collapses conferral, while removing the same number of random directions changes nothing.

its softmax scores. This setup confers 0.832 first-token, 96.9% of a task map trained on matched data (0.859) (Fig. 3; §A.9).

Additionally, removing this answer presence from the grafted vector by iteratively projecting out the linear probe's directions (Ravfogel et al., 2020) collapses conferral $0.904 \rightarrow 0.200$, while erasing the same number of random directions leaves it at 0.902 (Fig. 4). This erasure holds for other pairs (Qwen $0.79 \rightarrow 0.34$; Llama $0.28 \rightarrow 0.07$). Iteratively removing directions from the donor state collapses conferral in the same way (§A.19).

For Gemma, while grafted vector erasure is significant relative to a random baseline, its results suggest that the answer subspace is still fairly large at $540/2{,}304$ dimensions. This subspace is nonetheless sufficient on its own, as projecting the graft onto those directions and discarding everything else leaves conferral at 0.896. Splitting the map at rank 16, however, establishes a sharper picture, as the top sixteen channels recover 99% of full-rank conferral on their own, while the remaining $2{,}288$ recover 17% of it while carrying the larger share of the map's output energy (57% against 43%; §5). Despite only needing 16 directions to encode the answer into the recipient, due to the lossy nature of this encoding, the task map learns to spread that signal across 540 directions in the grafted vector.

The stitched activations are also not simply readout directions of the recipient's unembedding matrix. The map's top eight left singular directions are only 24.5% contained in the recipient's eight unembedding directions for the answer tokens, measured as the mean squared cosine across principal angles (chance 0.3%). Zeroing that readout component still leaves conferral at 0.662, and keeping only the readout component reaches just 0.196, suggesting transfer also encodes an answer intermediate that is decoded by the later layers.

6.2 THE RECIPIENT EMITS THE DONOR'S ANSWER, USING ITS OWN FEATURES.

The task map's learned transfer of a precomputed answer makes the output almost entirely dependent on the donor's residual stream. When we feed a perturbed problem $P' = (a, b, c + \delta)$ to the recipient and the original problem P to the donor, making sure to fix the prompt's token length for a consistent graft position, the recipient emits the first token of the donor's answer 0.890 of the time, and the perturbed one only 0.088. These sit against unstitched baselines of 0.366 (the recipient's wrong answer coinciding with y_P's digit) and 0.000 (by construction) respectively across five seeds and 306 problems. If we instead feed the donor the perturbed problem, the recipient produces the corresponding answer 0.476 of the time, confirming it is capable of emitting either one (Appendix A.12).

Decoding Gemma's grafted state with a sparse autoencoder (Cunningham et al., 2024) pretrained on the recipient (Lieberum et al., 2024) shows that the task stitch moves features $6.3\times$ more answer-aligned than the average feature on arithmetic, and up to $82\times$ on state-tracking bins (§A.10), while the recon map moves almost nothing (168k against 50 in total feature movement). Even if we restore computation lost in the stitch by adding back donor arithmetic features the recon map drops, restoring content worth 17.6% of the graft's norm, the task map is essentially unchanged for both first-token and full-answer conferral, though the same addition lifts the recon map's own conferral from 0.191 to 0.248 (Appendix A.11).

7 WHAT DOES NOT TRANSFER

The stitch reads a single prefill state, so any computation the donor performs across generated tokens rather than within that one forward pass cannot cross it. We show this for multi-digit carry below, and for grade-school reasoning in §A.20.

When probing the Gemma donor's graft site for each answer digit, we find that the first is decodable at 0.83 while the second and third sit right around the majority baseline at just 0.21 and 0.14 (a nonlinear MLP probe reaches only 0.26 and 0.13). In prefill, the donor has not undergone sufficient serial computation to predict digits beyond the first, which explains this discrepancy. Swept from the graft layer to the final layer of the forward pass that predicts it, the second digit's decodability climbs from 0.18 to 0.46 and 0.39 for Gemma and Qwen, reaching $\approx$ 0.93–1.00 only once it has been emitted as a token. A similar trend emerges when grafting from those same states, as fixing the prompt and first digit and varying only which state we graft raises second-digit conferral for Gemma and Qwen from $0.23/0.15$ at the prefill state to $0.52/0.44$ at a deep decode-state layer, ultimately reaching ≈ 1.0 after emission (Table 17). The stitch does not appear to transfer computation that the recipient can continue, instead transferring the products only after they emerge in the donor. This strengthens the results of §5 and §6, as even when the answer is not sufficiently present at the donor site and additional downstream computation would therefore be required to recover it, we find little evidence that the stitch supplies the recipient with a computational state it can expand upon. Under a task-supervised mapping regime that directly rewards useful transferred information, this continued failure to obtain such a benefit suggests that the models' computational representations are insufficiently aligned to support reliable transfer through this interface.

8 CONFERRAL TRACKS ANSWER PRESENCE

Across tasks and depths, conferral tracks answer presence at the donor's graft site (Fig. 5), suggesting once more that a precomputed answer is transferred through the stitch.

8.1 CONFERRAL TRACKS PRESENCE ACROSS TASKS AND DEPTHS.

Despite requiring multi-step computation, our symbolic binding and state tracking evaluations at depths 1–5 confer the full answer on the unsolvable bin at $0.90/0.83/0.81$ and $0.79/0.63/0.74$ (Gemma / Qwen / Llama). These results exceed a native ~ 0.02 and a shuffle control of ~ 0.10.

Sweeping across depths on our symbolic binding and state tracking tasks (up to 14 and 10 steps of computation respectively) reveals that, although the donor's solve rate falls to $\sim$11–30%, conferral on the problems the donor still solves stays relatively flat across later depths (e.g. Qwen symbolic

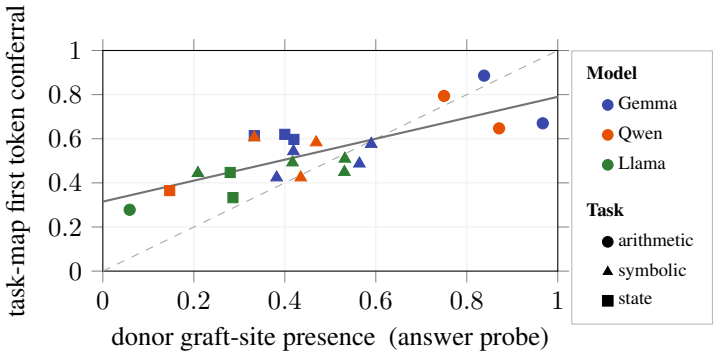

Figure 5: x is the answer's decodability from the donor's graft site (a linear probe), y is first-token conferral on the recipient's unsolvable bin. Each point is one (task $\times$ depth $\times$ model) bin. Llama arithmetic uses the whole-answer probe of §3. Spearman $\rho = 0.67$, $n = 22$. The dashed line is $y = x$, and the solid line the least-squares fit $y = 0.32 + 0.47x$ ($R^2 = 0.58$). The two cross at $x = 0.60$.

≈ 0.6 from depth 7 to 14; Table 10). Transfer therefore factors approximately as

$$\text{transfer rate} \;\approx\; P(\text{donor solves in a single pass}) \times P(\text{extract} \mid \text{present}),$$

where, beyond an initial drop from the shallow bins, task complexity has little further impact on the recipient's ability to confer a precomputed answer from the stitch site. This once again demonstrates the map's reliance on answer transfer. Once the answer is sufficiently encoded at the donor stitch site, variation in task complexity has little additional effect on the recipient's ability to extract it.

Plotting conferral against donor graft-site presence across 22 task, depth, and model bins is consistent with this (Fig. 5, Spearman $\rho = 0.67$, $n = 22$), although the small sample size does not establish this relationship definitively. At low presence, conferral sits above the hypothetical $1:1$ relationship shown by the dashed line. This is because, in these more complex cases, the donor has not yet begun optimizing for answer readout by the stitch site, meaning the answer presence that does exist is not linearly decodable for the probe. Conferral remains high at these points since the remaining layers of the recipient act as a nonlinear decoder able to extract less explicit answer encodings, introducing a bias term to this relationship. At high presence, conferral instead appears to fall below the hypothetical $1:1$ relationship. This is because, in these simpler problems, the donor has already begun explicitly encoding the answer by the stitch site, making the answer presence linearly decodable. This diminishes the benefit from the recipient's nonlinear layers, and the lossy nature of the task map's encoding dominates instead, reducing the slope of this relationship.

We can verify this trend experimentally by sweeping across Gemma and Qwen layers with a linear probe against a nonlinear MLP. On early layers, where the donor has yet to explicitly encode the answer, there is a sizable gap in their performance ($0.141/0.114$ for Gemma/Qwen). On later layers, however, the gap shrinks as much more of the answer becomes linearly decodable ($0.017/0.020$ for Gemma/Qwen; Table 11).

8.2 The relationship does not depend on the task family, vocabulary, or scale.

Stitching across model families from a Qwen2.5 7B or Llama-3.1 8B donor into a Gemma-2 2B recipient maintains the same profile we see within family, with leading-digit and full-answer conferral sitting well above controls ($0.823/0.886$ and $0.137/0.241$ for Qwen/Llama; controls in §A.14). Additionally, consistent with how few directions the map needs, the substance of the transfer is not dependent on the capacity of the recipient. Taking a Gemma-2 27B donor and comparing conferral between a 9B and 2B recipient on the 153-problem intersection where both fail natively and the donor succeeds yields nearly identical first-token conferral (0.719 against 0.693, McNemar $p = 0.42$), while full-answer conferral differs substantially (0.118 against 0.013, $p = 0.0004$). The first token is directly transferred from the donor, but since the full answer has not been computed at the donor stitch site, full-answer conferral depends primarily on the recipient's native capacity.

We also demonstrate that transfer extends to naturalistic tasks by evaluating model stitching on factual recall (TriviaQA, Gemma-2 9B→2B at the same layer pair, $n = 570$). Factual recall is unique in how the reconstructive and task maps confer the full answer (0.342 and 0.323) and first-token (0.326 and 0.351; these results are confounded by inconsistent tokenization §A.8) with no statistically significant difference (McNemar $p = 0.21$ and $p = 0.31$). Both sit comfortably above our controls (§A.8). Within factual recall, the task map recovers its advantage as the number of classes shrinks (§A.8). We take these results to reflect task map encoding differences between naturalistic and algorithmic tasks. Naturalistic tasks have a variety of different answer classes, making it difficult to train a map that consistently encodes each answer into the recipient. However, this distinctiveness allows them to survive a simple reconstructive map intact, leading to the recon conferral we see in our naturalistic evaluations. Conversely, for arithmetic tasks, our task map can differentiate between the far fewer classes, but digits specifically share a lot of latent content (e.g. what separates digits from other representations), leading to a reconstructive encoding that contains latent digit representation rather than information differentiating one digit from another. This tradeoff, however, remains suggestive, and we leave a more in-depth investigation to future work.

9 Limitations

Our method grafts from a single layer at prefill, but later digits are far more present in states read during generation (Table 17). We leave a dynamic grafting setup as potential future work to improve full-answer conferral. We also never evaluate a stitch that writes during the recipient's decoding. While both are viable and interesting experiments, our paper focuses primarily on establishing the relationships and mechanisms of cross-model stitching.

Our reconstructive map is a fairly conservative control. It is fit to reproduce the recipient's native state, so it ends up suppressing donor directions that could have improved conferral. To mitigate this risk, we evaluate a Procrustes map, a reconstructive map which adjusts only the donor vector's direction, for our arithmetic and factual recall arms (§A.17). This map confers largely as we would expect, elevating baselines below the task map, but this gap remains for evaluations that do not have a Procrustes arm (e.g. symbolic binding and state tracking).

Presence is measured with a linear probe, which can only decode what is linearly present at the stitch site. This broadly understates answer presence, especially on more complex tasks where the answer is not linearly decodable by the stitch site (§8). We do not use a nonlinear MLP as answer presence would be confounded with potential computation learned by the stitch.

The scatterplot has a Spearman ρ of 0.67 and a sample size of only 22. While it helps reinforce the relationship we establish in other evaluations, it is not sufficient evidence on its own.

Every pair we test is a decoder-only transformer of similar design, with donors of 7–27B into recipients of 0.5–2B, on English text. Whether the same correspondence holds across architecture families, for frontier models, or in other modalities such as vision is untested.

10 Conclusion

A task-supervised linear stitch is a functional channel that confers the donor's correct first token on problems the recipient cannot solve. While the stitch confers the first token exceptionally well, the full answer does not, even through more powerful maps. This is because these maps transfer the donor's precomputed answer, and the first digit of the answer is far more present at the prefill state we graft than the second or third digits. This result holds throughout our evaluations, including through cross-family stitching and naturalistic factual recall, and we ultimately find no evidence of transferred computation in the sense of §1. In the language of universality, our results suggest these models share where answers live, and are aligned enough that those answers can be moved linearly, but they likely do not share the circuits or computation necessary to compute these answers in a form that is readily transferable through the interfaces we test. Whether any stitch can move computation remains open.

Reproducibility Statement

Our paper describes everything necessary to reproduce our results. Our general stitching and evaluation procedures are given in §3. The model pairs and graft layers are given in Table 2, with the methodology behind choosing those layers in §3. Our various maps and objectives are described in §A.21, along with our prompting templates. Our full conferral tables can be found in Tables 4, 10, 12, and 16, and the compute, seed counts, and interval methods are in §A.22. We will release the evaluation notebooks and result files for all reported experiments in the camera-ready version.

AI Use Statement

We used generative AI tools to assist in constructing initial drafts of several evaluation notebooks, with each generated cell carefully reviewed prior to execution. We also used OpenAI's Deep Research to assist in identifying potentially relevant related work, auditing each result to ensure genuine relevance. Finally, we used Grammarly, which incorporates occasional AI-generated suggestions, to improve grammar and clarity. We take full responsibility for all content presented in this paper.

A Experimental Details and Full Results

A.1 Model pairs, layer derivation, and settings

All models run in bfloat16 with eager attention and greedy decoding (`transformers` 4.46.3). The recipient graft layer L_r for each family is set at the elbow of the activation-patching recovery curve. The counterfactual is operand substitution on a three-shot addition prompt with a clean $a + b$ against a corrupted $a + c$ with $c \neq b$, both sums restricted to a single digit so the first answer token is the whole answer and the two prompts have equal token length. The donor layer L_d is selected based on a combination of centered kernel alignment with L_r, how much accuracy patching that layer restores, and how reliably a linear probe reads the answer off it, which improves toward later layers (Table 2, last column, and §A.4). The primary metric is centered kernel alignment, with the accuracy and probing checks used to ensure a functional donor layer. Taking the CKA argmax alone gives layer 34 for Gemma, which is the layer we use, but layer 4 of 32 for Llama, far below any depth at which its donor has done anything substantial.

Table 3 gives the resulting evaluation bins. Arithmetic uses 3,000 train / 2,000 evaluation problems, sampled uniformly without replacement as $a \cdot b + c$ with $a, b \in [2, 40]$ and $c \in [1, 99]$ for Gemma and Qwen. Llama instead uses $a, b \in [2, 30]$, $c \in [1, 40]$ with answers restricted to $[100, 999]$, so that every answer is exactly three digits and therefore a single token under its tokenizer.

Pair	d_{model} (recip.)	(L_r, L_d)	layers (recip./donor)	recip. recovery @ L_r	donor probe @ L_d
Gemma-2 2B ← 9B	2304	$(20, 34)$	26 / 42	0.704 [0.692, 0.716]	0.838 [0.811, 0.861]
Qwen2.5 0.5B ← 7B	896	$(17, 23)$	24 / 28	0.968 [0.963, 0.972]	0.750 [0.720, 0.778]
Llama-3.2 1B ← 3.1 8B	2048	$(11, 17)$	16 / 32	0.831 [0.809, 0.851]	0.701 [0.649, 0.748]

Table 2: Model pairs and graft-layer derivation. Recipient recovery is the patching-recovery fraction at L_r; donor probe is a linear answer probe at L_d.

A.2 Full arithmetic conferral (unsolvable bin)

Table 4 expands Fig. 2 with confidence intervals, the self-graft control, and full-answer conferral. Native accuracy and self-graft (overwriting the recipient's own state) are zero by construction. The shuffled control and reconstruction map act as conservative floors (with §A.17 serving as a more neutral floor).

A.3 Capacity is not the bottleneck

Table 5 gives full-answer conferral on the Gemma unsolvable bin across various types of maps. Table 6 reports first-token conferral for task maps of various ranks across all three model families.

Table 3: **Arithmetic evaluation bins.** Graft layers are in Table 2, derived once per pair (activation patching + CKA). Conferral is measured on the unsolvable bin, where native accuracy is zero by construction.

Pair	donor-solved n	recipient solves	unsolvable n
Gemma-2 9B → 2B	1689	62%	638
Qwen2.5 7B → 0.5B	1681	66%	568
Llama-3.1 8B → 3.2 1B	642	2.5%	626

Condition	Gemma	Qwen	Llama
Native, first	0.000 [.000,.006]	0.000 [.000,.007]	0.000 [.000,.006]
Self-graft, first	0.000 [.000,.006]	0.000 [.000,.007]	0.000 [.000,.006]
Shuffled, first	0.102 [.081,.128]	0.178 [.149,.211]	0.005 [.002,.014]
Recon map, first	0.191 [.163,.224]	0.134 [.108,.164]	0.019 [.011,.033]
Task map, first	**0.886 [.859,.908]**	**0.794 [.759,.825]**	**0.278 [.244,.314]**
Task map, full	0.127 [.105,.148]	0.038 [.028,.048]	0.291 [.276,.305]
Recon cosine (recon / task)	0.981 / 0.219	0.991 / 0.071	0.977 / 0.261

Table 4: Arithmetic conferral on the unsolvable bin. The task map confers far above every information-preserving baseline while landing off-manifold (low task reconstruction cosine).

The MLP map of Table 1 is a single hidden layer of width 1024 with a GELU nonlinearity, $d_{\text{donor}} \to 1024 \to d_{\text{recipient}}$, trained on the same objective as the linear task map. The oracle map scores slightly below the plain task map on full answers (0.12 against 0.13) despite its privileges. It is important to note, however, that the oracle map is optimized for first-token cross-entropy, meaning this result bounds full-answer capabilities for the first-token objective specifically. The free vector is one unconstrained vector per problem, initialized from the reconstruction map's output so it starts on-manifold and optimized with Adam (lr $= 5 \times 10^{-2}$, 150 steps) against cross-entropy over every answer token, on a 200-problem subsample with the recipient frozen. Note that the free vector's outputs are only partly driven by linearly decodable structure, with digit probes on the optimized vectors recovering $0.64/0.34/0.42$ of the answer's three digits, so the recipient's downstream layers must unpack a partly nonlinear code.

Condition (Gemma)	Full-answer
Native recipient	0.003 [.001,.011]
Linear task map	0.127 [.105,.148]
MLP map	0.168 [.141,.199]
Oracle map (fit on eval)	0.119 [.096,.147]
Linear map, whole-answer obj.	0.225 [.194,.259]
MLP map, whole-answer obj.	0.245 [.213,.280]
Donor (reference)	*0.42*
Free vector	**1.000 [.981,1.000]** (n=200)

Table 5: Map capacity (Gemma). The two whole-answer rows come from a separate matched session ($n = 632$), where the first-token linear task map measures 0.136 and the donor's own completion rate 0.418.

Rank	Gemma	Qwen	Llama
1	0.215	0.356	0.000
2	0.541	0.528	0.006
4	0.708	0.671	0.016
8	0.820	0.646	0.050
16	0.875	0.790	0.139
32	0.895	0.789	0.257
full	0.887	0.796	0.278

Table 6: Low-rank task-map first-token conferral (unsolvable bin). Full ranks: 2304 / 896 / 2048.

A.4 Donor answer-presence rises with depth

A linear first-digit probe over donor layers (Table 7) shows the answer's decodability generally increasing toward later layers, motivating the graft-layer choice and matching the emergence result (§7).

Donor layer	L3	L11	mid	deep		final
Gemma (of 42)	0.327	0.459 (L11)	0.542 (L27)	0.688 (L31)	0.838 (L34)	0.970 (L39)
Qwen (of 28)	0.396	0.503 (L11)	0.479 (L19)	0.750 (L23)	0.952 (L27)	—
Llama (of 32)	0.548	0.673 (L11)	0.701 (L17)	0.897 (L19)	0.910 (L23)	0.941 (L31)

Table 7: Donor first-digit probe accuracy by layer. Bold graft layers are L34 / L23 / L17.

A.5 Arithmetic depth dose–response

While increasing the step count lowers the donor and recipient solve rates, it has little to no effect on the relationship between donor presence and first-token conferral, which tracks across task depth and model families (Table 8). Llama's donor cannot solve the deeper problems in one pass, so there is nothing to transfer.

	steps	donor solves	recip. native	donor presence	task confer (first)
Gemma	1	0.994	0.964	0.904	0.833 [.608,.942]
	4	0.899	0.503	0.911	0.891 [.854,.919]
Qwen	1	0.870	0.977	0.619	0.600 [.313,.832]
	4	0.843	0.749	0.754	0.757 [.688,.816]
Llama	1	0.970	0.693	0.395	0.007 [.001,.037]
	4	0.040	0.094	0.062	0.034 [.006,.172]

Table 8: Arithmetic depth dose–response (steps 1 and 4 shown; 2–3 interpolate). "Donor solves" is over the full evaluation bin. Every other column is restricted to the problems the donor solves, and conferral further to the unsolvable bin within those. "Presence" is the donor first-digit probe. All rows are single-seed. Gemma's depth-1 row is $n = 18$, against the $n = 106$ expanded depth-1 pool quoted in §8, which is why the two report different presence and conferral at the same depth. The numbers are so small due to the simplicity of questions at depth 1. At depth 4 Llama's donor solves only 0.040 of the pool in a single pass, so the unsolvable bin is nearly empty and the row carries little weight.

A.6 The linear probe is a floor, and it matches conferral at the graft layer

While a nonlinear (MLP) probe reads the answer slightly better than a linear probe, first-token conferral stays relatively similar for Gemma and Qwen (Table 9, top; Llama's high class count gives a distinct advantage to nonlinear probes). For Gemma and Qwen, later digits stay near the majority baseline under both probes (bottom), so little about them is recoverable from the prefill state. Llama's higher digit-3 probe reflects its packed-token structure rather than genuine early presence. While second and third digit presence is minimal, it does exist, as a channel carrying only the correct leading digit tops out at 0.116 full-answer (Table 14), while both the MLP map (0.168) and task map with the whole answer objective (0.25) clear it.

At validated L_d	linear probe	MLP probe	task conferral
Gemma (L34)	0.834	0.903	0.902
Qwen (L23)	0.750	0.815	0.794
Llama (L17)	0.701	0.782	0.278
Later digits at L_d (linear / MLP; majority baseline)			
Gemma digit 2 / digit 3	0.214 / 0.144	0.256 / 0.130	(maj. 0.138 / 0.117)
Qwen digit 2 / digit 3	0.180 / 0.125	0.199 / 0.119	(maj. 0.122 / 0.121)
Llama digit 2 / digit 3	0.153 / 0.461	0.153 / 0.511	(maj. 0.121 / 0.131)

Table 9: Linear vs. nonlinear probes. Llama's low arithmetic conferral despite high leading-digit presence is the packed-tokenizer mismatch discussed in §3/§8. The task-conferral column is single-seed. The donor probe is measured over the full evaluation set here; the 0.78 quoted in §1 and §6 is the same probe restricted to the $n = 638$ unsolvable bin, which is the comparison those sentences make.

A.7 NON-ARITHMETIC TASKS BY DEPTH

On the presence–conferral scatter (Fig. 5), the state tracking and symbolic binding bins occupy the shallower part of the relationship, allowing us to generalize the tracking beyond just arithmetic. We additionally extend our symbolic binding and state tracking evaluation up to depths of 14 and 10 respectively, where our prior findings are verified across the suite (Table 10).

Task	pair	task (first)	recon (first)	shuffle (first)	native (full)
Symbolic (depths 5–14)	Gemma	0.507 [.425,.589]	0.271	0.150	0.036
	Qwen	0.557 [.472,.639]	0.198	0.145	0.023
	Llama	0.474 [.410,.538]	0.228	0.289	0.031
State track (depths 5–10)	Gemma	0.608 [.539,.673]	0.146	0.085	0.020
	Qwen	0.301 [.231,.383]	0.059	0.125	0.007
	Llama	0.383 [.305,.468]	0.120	0.150	0.000

Table 10: Extended-depth non-arithmetic conferral (pooled over the deep bins). The main-text §8 headline is measured on the shallower depth 1–5 generation; these deeper pools are lower but still far above the floors, and flat across depth (e.g. Qwen symbolic ≈ 0.6 from depth 7 to 14). All conferral rows are single-seed.

Pair	donor layer	linear probe	MLP probe	gap
Gemma (of 42)	L14	0.462	0.603	0.141
	L21	0.475	0.586	0.111
	L30	0.603	0.738	0.135
	L34	0.834	0.903	0.069
	L37	0.935	0.952	0.017
Qwen (of 28)	L9	0.520	0.634	0.114
	L14	0.532	0.648	0.116
	L20	0.508	0.620	0.112
	L23	0.750	0.815	0.065
	L25	0.913	0.933	0.020
Llama (of 32)	L10	0.673	0.698	0.025
	L16	0.685	0.766	0.081
	L17	0.701	0.782	0.081
	L23	0.910	0.910	0.000
	L28	0.947	0.956	0.009

Table 11: Linear and nonlinear first-digit probes on the donor state, swept over donor layers. Graft layers are in bold. Majority baselines are $0.321/0.331/0.349$, and the run is single seed.

A.8 NATURALISTIC FACTUAL RECALL

Table 12 gives conferral on the factual-recall bin, and Table 13 with Fig. 6 the sweep over answer-class count.

A.8.1 DATASETS

Facts are drawn from TriviaQA (Joshi et al., 2017) (`rc.nocontext`, train split), keeping $40{,}000$ questions whose answer is at most three words and de-duplicated on the prompt. Prompts use three exemplars that are not drawn from TriviaQA. A $60/40$ split gives $24{,}000$ training items. Of the $16{,}000$ on the evaluation side we keep the first $3{,}000$ and discard the rest due to computational expenses. Because de-duplication is on the prompt, no evaluation question appears in training. The unsolvable bin is built on the decoded full-answer for both models, exactly as in the cross-family arms, because of the variety of first-token fragments that can be used to construct the same full-answer result, which occasionally leaves full-answer conferral above first-token. The donor solves 0.659 and the recipient 0.502 of the evaluation set, giving $n = 570$ problems the donor answers and the recipient does not. A generation counts as correct if it normalizes to the gold answer, or if it

begins with the gold answer followed by a word boundary, so *Japan is an island nation* counts for *Japan* but *Japanese* does not.

A.8.2 What counts as a class

The task map is trained with cross-entropy on the gold first answer token, so we define answer classes as unique first tokens. The $24{,}000$ training items contain $14{,}369$ distinct full answers but only $7{,}252$ distinct first-token classes (3.3 items per class; $3{,}600$ classes and 5.7 items per class after singleton classes are dropped from the cross-entropy loop). The bin's 570 items span 459 distinct first-token classes. All class counts below are first-token classes.

A.8.3 Map

The task map is a rank-32 residual on the reconstruction map, $W = W_{\text{recon}} + AB$ with B initialized to zero, trained with weight decay at lr $= 10^{-4}$. Full-rank training at 10^{-3} was tried first and overfits due to the number of classes ($7{,}252$ classes; training conferral reaches 0.99 within one epoch while held-out conferral falls).

Condition	first-token	full-answer
Native recipient	0.065 [.047,.088]	0.000 [.000,.007]
Self-graft	0.061 [.044,.084]	—
Shuffled donor, task map	0.004 [.001,.013]	—
Shuffled donor, recon map	0.004 [.001,.013]	—
Reconstruction map	0.326 [.289,.366]	0.342 [.304,.382]
Task map	**0.351 [.344,.357]**	**0.323 [.286,.362]**

Table 12: Factual-recall conferral on the unsolvable bin ($n = 570$, Gemma-2 9B→2B, $L_{34} \to L_{20}$). Native full-answer is 0 by construction. The task-map first-token row is reported over five seeds (per-seed 0.342–0.356). Every other row, and the task-map full-answer row, is single-seed with a Wilson interval. Paired McNemar on seed 0 gives $p = 0.31$ (first-token, 26 vs 35 discordant) and $p = 0.21$ (full-answer, 38 vs 27). The number of unique classes makes it difficult for the task map to learn a consistent encoding for each token, which is why the task map does not exceed the reconstruction map on this task. Restricting to the 410 bin items never used for checkpoint selection leaves both unchanged (0.341 and 0.349).

A.8.4 Answer inventory

Our sweep across the number of classes K reveals that, as the number of classes begins to fall, the task map begins to exceed the reconstruction map (Table 13, Fig. 6). We evaluate on a single seed across two different selection criteria: once with top-K by training frequency, and twice with K drawn at random, leading to three runs total. Both maintain the same number of training items across the sweep so only the number of classes varies, and the reconstruction map is re-scored on each condition's own evaluation subset, making $\Delta = \text{task} - \text{recon}$ the value of interest. A shuffled-donor arm is run per condition to rule out a restricted-output prior at small K; it stays at or below 0.072 throughout.

At $K = 50$ the two rules differ tenfold in items per class, 79.3 under the frequency rule against 7.7 under the random rule, since the commonest classes carry far more training items than an arbitrary draw does. This disparity leaves Δ unchanged ($+0.160$ against $+0.172$), showing that only the number of classes K matters.

A.9 The answer bottleneck

Table 14 expands the bottleneck test of §6, where the donor layer is replaced by a frozen linear answer probe fit on held-out problems whose $K = 9$ softmax scores are the only thing a writer sees when producing the graft. The reader and writer are each trained on disjoint halves of the original $1{,}500$ problem set used on the task map. We train a separate task map on the writer's $1{,}500$ problem set to ensure a fair comparison. Evaluation is the same $n = 638$ Gemma unsolvable bin used throughout.

Selection	classes	items/class	eval n	recon	task	Δ	McNemar p
Class-matched, top-K by frequency (budget 3,963)							
	50	79.3	125	0.448	0.608	+0.160	0.004
	200	19.8	306	0.360	0.415	+0.056	0.068
	337	11.8	486	0.346	0.362	+0.017	0.509
Class-matched, random draw (budget 387)							
	50	7.7	58	0.448	0.621	+0.172	0.002
	46	8.4	54	0.315	0.519	+0.204	0.003
	113	3.4	246	0.317	0.346	+0.028	0.167
	105	3.7	251	0.315	0.351	+0.036	0.122
	157	2.5	486	0.346	0.350	+0.004	0.888
	157	2.5	486	0.346	0.352	+0.006	0.775

Table 13: First-token conferral against the number of answer classes the task map is trained on. Classes is the number the map was actually trained on, which falls below the requested count because only some answer classes in the unsolvable bin appear more than once in training, and, for large K on the random draw, since we must maintain a smaller number of training items throughout the sweep, the class count may exceed the training-item budget, forcing us to remove those items. The random-draw rows are two draws each at requested $K = 50$, 200 and 800. How classes are chosen is the only decision separating the two blocks; the differing budgets, items per class and evaluation subsets are all consequences of that choice.

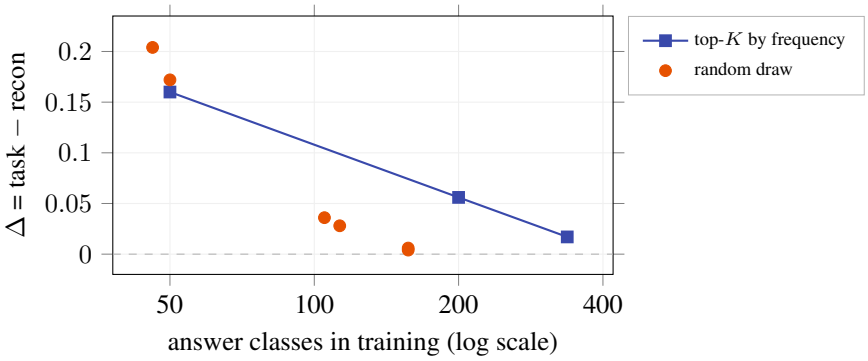

Figure 6: The task map only sustains an advantage at lower values of K. Training-item budget is fixed within each rule and the reconstruction map is re-scored on each condition's own evaluation subset, so Δ is comparable within each rule. The random-rule points sit at a tenth of the frequency rule's budget (387 against 3,963 items) and reproduce the same trend.

Condition	First-token	Full-answer
Reader accuracy on the bin (ceiling)	0.834 [.803,.861]	—
Uniform scores (bias only)	0.099 [.078,.124]	—
Shuffled donor	0.017 [.010,.031]	—
Predicted scores	**0.832 [.801,.859]**	**0.100 [.079,.126]**
Gold one-hot	1.000 [.994,1.000]	0.116 [.093,.143]
Task map, matched data	0.859 [.830,.884]	—
Task map, full data	0.886 [.859,.908]	0.129 [.105,.157]
Reconstruction map	0.191 [.163,.224]	—

Table 14: The answer bottleneck on the Gemma unsolvable bin ($n = 638$). A writer seeing only nine answer scores recovers 96.9% of the matched-data task map. Additionally, the predicted scores track almost exactly with the reader's accuracy, and a writer handed a probe with the one-hot gold labels is able to output that correct label at 1.0, meaning, when the answer is decodable, the task map can reliably encode it into the recipient such that it outputs the correct answer. Paired McNemar against the full-data task map leaves a residual advantage for the task map (62 against 28 discordant, $p = 0.0004$), so the bottleneck reproduces nearly all of conferral but not quite all of it.

For the shuffled row, not only does own-answer conferral fall to 0.017, but the recipient instead emits the partner's answer on 0.835 [.805,.862] of the bin, within noise of the actual evaluation's 0.832. Additionally, when only providing the first token, full-answer conferral falls to 0.100 against the task map's 0.129, suggesting once again that there isn't much of the full-answer to recover at the donor stitch site.

A.10 Answer-aligned features the stitch turns on

We decode the recipient's state with a sparse autoencoder ($16{,}384$ features from `gemma-scope-2b-pt-res-canonical`, `layer_20/width_16k/canonical`) and take the features whose mean activation rises most under the task stitch relative to the recipient's native run. We score each one by its *answer alignment* (η^2, the share of a feature's activation variance accounted for by the answer's leading digit, so a feature scores high when it fires at different levels across digits and near zero when it does not), to ask whether they are primarily answer-carrying features. This is measured on the recipient's own ungrafted states, meaning we isolate nodes the task map would move in the recipient if it were trying to encode an answer. Added features are the 30 with the largest rise in mean activation under the task stitch relative to native. The multipliers below are the mean η^2 of those features divided by the mean η^2 over all 16k features. Arithmetic added features are $6.3\times$ more answer-aligned than the average feature; state tracking $39\times$ overall, rising to $82\times$ at depth 4.

- Arithmetic, top added: 159, 14637, 11555, 13399, 2865, 7435, 7089, 2835, 379, 7506.
- Symbolic (deep bins), top added: 8231, 14023, 9998, 4140, 15111, 7019, 9089, 9185.
- State tracking (deep bins), top added: 14023, 8231, 16193, 12748, 6822, 2115, 11164.
- Cross-task recurring (candidate shared answer/output machinery): 2865, 629, 10356.

A.11 Restoring discarded donor features

We additionally want to make sure that a lack of computation is not simply a confound of the task map discarding the most relevant arithmetic features. To test this, we identify the donor's arithmetic features using a sparse autoencoder and reconstruct their contribution in the donor model's dimensions. We then project the reconstructed arithmetic features away from the arithmetic features maintained through the recon map and add them to the map's output after stitching into the recipient's dimensions. After adding those features back, despite a probe confirming that first-token output is still decodable (0.73), conferral for the task map is unchanged, at both first token ($0.886 \to 0.882$) and full-answer ($0.129 \to 0.129$).

A.12 Scoring the shuffle and perturbation baselines

The shuffled control feeds the task map an unrelated problem's donor state. It attains an accuracy of 0.109 on the problem the recipient reads (this section rebuilds the bin in a separate five-seed session, which is why it reports 634 pairs and 0.109 against the 638 and 0.102 of Table 4). Scored instead by what the recipient actually emits, its first token is the leading digit of the donor's answer 0.931 of the time (0.930 as a five-seed mean), on the 564 of 634 pairs whose two answers do not share a first digit. Each output is additionally scored against a third, unrelated problem's answer drawn from the same pool over 20 draws to get a better idea of this set's first-token distribution. That matches 0.202 of the time (SD 0.0163), putting the observed rate 44.8 standard deviations above chance. The lack of full-answer transfer exists here too. The recipient reproduces the unrelated problem's full-answer only 0.005 of the time, and is correct for the problem it actually reads on 0.014 of first tokens and 0.007 of full answers.

In the perturbation test of §6, every δ is chosen so that y_P and $y_{P'}$ begin with different digits. Problems admitting no such δ within range are dropped.

A.13 Geometry of the graft

Table 15 measures each grafted vector against the recipient states it replaces on the unsolvable bin. The z-distance standardizes by the recipient's per-coordinate mean and standard deviation and reports the root mean square of those per-coordinate scores, $\|(v-\mu)\oslash\sigma\|_2/\sqrt{d}$, so a native state scores ≈ 1.0 by construction. The task map is far off-manifold for Gemma and Qwen but close to on-manifold for Llama. Extending training six epochs moves the task cosine from 0.197 to 0.158 (Gemma) and 0.819 to 0.774 (Llama), so the map is converging on optima that land outside the recipient's natural manifold. Its lack of ties to the original manifold likely stems from optimizing for answer encoding rather than computation. The 0.197 here and the 0.219 in Table 4 are the same quantity in two separate runs of the same experiment, the follow-up run of this appendix and the main evaluation run respectively.

	recon map			task map			lead confer.	
Donor	cos	$\|\cdot\|$	z	cos	$\|\cdot\|$	z	recon	task
Gemma-2 9B	0.978	0.98	0.95	0.158	6.95	13.41	0.584	0.935
Qwen2.5 7B	0.973	0.97	0.91	0.254	4.31	8.49	0.544	0.816
Llama-3.1 8B	0.933	0.89	0.62	0.774	1.12	1.51	0.572	0.881

Table 15: Grafted activation geometry compared against the recipient's native activations, all into Gemma-2 2B at L_r=20. $\|\cdot\|$ is the written vector's norm divided by the norm of the native state it replaces. Task cosines are from the extended-training run described above. $n = 604/439/236$.

A.14 Controls for the generalization arms

Table 16 reports our generalization results (two cross-family, two across recipient scale, and two instruction-tuned) each with its own shuffle and reconstruction controls. Graft layers (L_r, L_d) are $(20, 23)$ for both cross-family arms, $(34, 37)$ and $(20, 37)$ for the two 27B recipients, and $(20, 34)$ for both instruction-tuned arms, which are not paired arms. The cross-family unsolvable bin is based on full-answer correctness, so there is native leading digit accuracy. The 27B rows are the full unsolvable bin rather than the difficulty-matched intersection used in §8, and the donor solves 0.894 of the evaluation set ($n = 1789$).

A.15 Later digits at the donor site

Table 17 has three blocks. The first probes the donor's graft-site state for each answer digit. The second probes the second digit at the state that will predict it, and the state after it has been emitted, showing how the additional serial computation improves the decodability of later digits. The third grafts from each of those points, varying only the donor state, meaning we read from different donor activations but write into the same recipient in prefill. We leave a stitch that writes into the recipient while it decodes to future work.

Arm	native	shuffle	recon	task (first)	task (full)
Qwen2.5 7B → Gemma-2 2B	0.542	0.182	0.544	0.823	0.137
Llama-3.1 8B → Gemma-2 2B	0.453	0.119	0.572	0.886	0.241
Gemma-2 27B → 9B (full bin)	0.000	0.088	0.360	0.782	0.115
Gemma-2 27B → 2B (full bin)	0.000	0.090	0.217	0.824	0.053
Gemma-2 9B → 2B (base → IT)	0.000	0.131	—	0.850	0.153
Gemma-2 9B → 2B (IT → IT)	0.000	0.150	—	0.811	0.137

Table 16: Detailed results of our generalization evaluations (two cross-family, two across recipient scale, and two instruction-tuned)

Donor graft-site probe	digit 1	digit 2	digit 3
linear	0.83	0.21	0.14
MLP	—	0.26	0.13
Second digit	Gemma / Qwen / Llama		
at the state that predicts it	0.46 / 0.39 / 0.49		
once emitted as a token	0.93–1.00		
Second-digit conferral by graft source	Gemma / Qwen		
prefill state (our stitch)	0.23 / 0.15		
deep decode-state layer	0.52 / 0.44		
after emission	≈1.0		
native / shuffle	0.00 / ≈0.10		

Table 17: Later-digit availability and transfer.

A.16 Grafting every prompt position

Table 18 grafts at every prompt position across a given layer instead of just the final one. We individually feed the roughly fifty vectors (corresponding to the 50 tokens on a given prompt) into a single shared matrix so as to not confound with a higher parameter count. We only run this on the three same-family pairs due to grafting by position requiring a shared tokenizer. Both conditions are trained and scored in the same session against the same bin and seeds, under a shorter schedule than the main results (two seeds and two epochs, against five and six). Thus, we compare these results against a $k = 1$ baseline retrained with those seeds.

Pair	(L_r, L_d)	one position	every position
Gemma-2 9B → 2B	(20, 34)	0.206	0.191
Qwen2.5 7B → 0.5B	(17, 23)	0.070	0.039
Llama-3.1 8B → 3.2 1B	(11, 17)	0.222	0.256

Table 18: Conferral under the whole-answer objective. Gemma and Qwen are full-answer; Llama is first-token, which spans its whole answer (§3). Two seeds.

Despite the widened channel, full-answer conferral stays flat for Gemma ($0.206 \rightarrow 0.191$) and falls for Qwen ($0.070 \rightarrow 0.039$) against the single-layer map, once again showing no sign of computational transfer. A sweep on Gemma writing at the last $1, 2, 4, 8, 16$ and all positions under the first-token objective holds full-answer conferral between 0.090 and 0.112 throughout, while first-token conferral rises from 0.812 to 0.875. Our controls hold too as replacing every position with a different problem's donor run (length-matched) drops first-token conferral to 0.153.

Conversely, Llama, which tokenizes entire answers rather than single digits, actually does improve full-answer conferral with extra positions ($0.222 \rightarrow 0.256$). This result, along with the improvement in Gemma's first-token conferral, shows that additional layers can draw out a bit more of what's already present, but can't compute what isn't there to begin with (like the full-answer for Gemma and Qwen).

We also fit and graft task maps at three recipient-donor pairs at once, $(18, 31)$, $(20, 34)$ and $(22, 37)$, which confers 0.870 $[.842, .894]$ first-token on the Gemma unsolvable bin, against 0.886 $[.859, .908]$ for the single site. The additional layers added virtually nothing to conferral.

A.17 An unsupervised alignment baseline

Table 19 compares a Procrustes map against the reconstruction and task maps, and Table 20 tracks it across answer-class count. The reconstruction map is built to discard whatever the donor knew and the recipient did not, making it often underestimate a true direction-preserving floor. To solve this, we fit an orthogonal Procrustes map with A and B the centred donor and recipient states on the training set, $R = UV^\top$ from the singular value decomposition of $A^\top B$, applied with a single scalar scale. R has orthonormal columns, so it applies no shrinkage to its retained subspace. It is not lossless, since mapping $d_{\text{donor}} \to d_{\text{recipient}}$ discards a subspace of dimension $d_{\text{donor}} - d_{\text{recipient}}$. It is fit on activations alone and never sees an answer.

Pair	dims discarded	reconstruction	Procrustes	task map
Gemma-2 9B → 2B	36%	0.190	0.441	0.881
Llama-3.1 8B → 3.2 1B	50%	0.019	0.026	0.217
Qwen2.5 7B → 0.5B	75%	0.134	0.465	0.785

Table 19: First-token conferral on the arithmetic unsolvable bin. All three rows are at the layer pairs of Table 2. All three columns use the short schedule of §A.16.

The map recovers half of the first-token channel in pairs with a functioning task map: 0.441 against Gemma's 0.881 and 0.465 against Qwen's 0.785, roughly three times what the reconstruction map delivers in each. How much of the donor's dimension is discarded does not govern this, since Qwen discards twice the subspace Gemma does and scores slightly higher. Llama is low in all three columns because its own task map is starved by answer-class count (§A.18).

On factual recall, the Procrustes map confers 0.333 first-token and 0.368 full-answer against a task map with 0.351 and 0.323 (and a reconstructive map with 0.324 and 0.338). Additionally, consistent with our understanding of how the task map confers, Procrustes gains nothing from having fewer answer classes, as it cannot specifically encode answers into the recipient like the task map can. Its margin over the reconstruction map is smallest at $K = 50$ ($+0.038$) and shows no trend across the sweep, where the task map's margin is $+0.213$ at $K = 50$ and near zero above it (Table 13).

K answer classes	examples per class	eval n	reconstruction	Procrustes
50	83.7	106	0.387	0.425
200	20.9	181	0.348	0.442
800	5.4	281	0.320	0.356
3200	2.2	404	0.262	0.319

Table 20: First-token conferral for the two label-free maps as the number of answer classes varies, with the item budget fixed at $4{,}183$ and each condition scored on its own evaluation subset. Unlike Table 13, the reconstruction baseline is refit per condition, so its values are not comparable to that table's.

A.18 Training-set size and answer-class count

The task map is trained with cross-entropy on the gold first answer token, so its output classes are distinct first-token targets. For Gemma and Qwen, those are the nine leading digits. For Llama, whose tokenizer packs a three-digit number into one token, it is the set of distinct answers the problem range admits, which is 841.

On Llama, training the same $k = 1$ map on $18{,}000$ problems instead of $3{,}000$, scored on the same unsolvable bin in the same session, raises conferral from 0.217 to 0.586 (McNemar $p < 10^{-40}$, 264 against 33 discordant). The identical change on Gemma moves conferral from 0.881 to 0.913, a difference of 0.032, since Gemma's nine classes are already saturated at $3{,}000$ problems. Llama's

low conferral in Table 4 is therefore an artifact of Llama's examples per answer class. The same regime appears on factual recall, where the task map loses its advantage over the reconstruction map at several hundred answer classes and recovers it below fifty (Table 13), and where the label-free maps of §A.17 show no such dependence.

A.19 Answer-subspace erasure

§6 performs INLP erasure on the grafted vector. On Gemma the task map falls to $0.174\,[.146, .206]$ against $0.892\,[.866, .914]$ for an equal number of random directions at the converged rank of 540 (Fig. 4). The free vector behaves the same way, falling to $0.225\,[.173, .288]$ against $0.890\,[.839, .926]$.

Table 21 instead performs INLP erasure from the donor state before it is sent through the task map. We iteratively fit a first-digit probe on the donor's graft-site states, project its directions out, and push the ablated donor state through the unchanged map.

Pair	unablated	answer directions erased	random directions erased
Gemma-2 9B → 2B	0.881	0.350	0.861
Qwen2.5 7B → 0.5B	0.785	0.246	0.782
Llama-3.1 8B → 3.2 1B	0.217	0.002	0.241

Table 21: First-token conferral with 540 directions removed from the donor state at L_d, against the same number of random directions. The map is frozen throughout and only its input changes. Short schedule, as in §A.16.

Conferral collapses in all three pairs while matched-rank random erasure leaves it intact. Because the map is never refit, this cannot be explained by any property of the map's output geometry.

A.20 GSM8K

GSM8K word problems require several steps. A linear probe on the donor's graft-site activation recovers the leading digit of its final answer at only 0.14, below the 0.28 majority baseline. Stitching does not improve accuracy on the full $1{,}319$-problem test set when grafting at the single site, across a band of layers, or through an oracle map refit on the eval problems (0.265, 0.249 and 0.262 against a 0.246 baseline). These are absolute accuracies on the full test set, not conferral in the sense of §3, which is defined only on an unsolvable bin.

Two features of the setup differ from the other tasks. GSM8K answers are emitted token by token after a chain of thought, so no single answer token sits at the graft position for a cross-entropy task map. The stitch here uses the reconstruction map, fit by ridge regression from the donor's state at L_d to the recipient's at L_r, fit over every prompt position. And because scoring is on the full test set rather than an unsolvable bin, the recipient's own solutions are included and the baseline is 0.246 rather than zero.

A.21 Objectives and prompts

The reconstruction map is fit by ridge regression ($\lambda = 10^3$) from the donor's residual state at L_d to the recipient's own residual state at L_r, both taken at the last prompt token, over the same $3{,}000$ arithmetic training problems the task map uses. It sees no answer labels. Its target is the state the recipient would have had, which is why it is a conservative rather than a neutral control (§9, §A.17).

The KL objective is trained on the donor's output logits. The teacher distribution is the donor's own next-token distribution at the last prompt token, cached once over the training problems, and the student is the recipient's next-token distribution with the graft in place. We minimize the forward $\mathrm{KL}(\text{teacher} \,\|\, \text{student})$ at temperature 1, optionally plus λ times the mean-squared error between the graft and the recipient's own state at L_r. $\lambda = 0$ is pure distillation and the sweep runs $\lambda \in \{0, 0.1, 1, 10\}$. The MSE term is the same quantity the reconstruction map minimises, so the sweep interpolates between distilling the donor and reproducing the recipient.

Sweeping λ, first-token conferral falls from 0.95 to 0.28 as reconstruction cosine rises from 0.24 to 0.95. The $\lambda = 0$ setting never sees a gold answer, only the donor's own next-token distribution, yet it outperforms the task map's 0.886. This is because the channel is trained on donor output content, and since conferral scores agreement with the donor, the KL objective is able to optimize for a fundamentally easier task (matching its own output versus task correctness), while still maintaining all of the conferral.

A.21.1 THE SYNTHETIC-TASK AND FACTUAL-RECALL PROMPTS

Arithmetic uses a six-line few-shot prefix followed by the problem and a trailing =:

```
2 + 5 = 7
6 * 3 = 18
4 * 7 + 2 = 30
9 * 8 = 72
3 * 4 + 5 = 17
40 * 20 = 800
17 * 23 + 45 =
```

Symbolic binding and state tracking each use two worked exemplars in the same format as the query, shown here with one exemplar each for space:

```
Resolve the final label.
Definitions:
dax is red.  wug is dax.
Question:  What is wug?
Answer:  red

Track the final color.
Initial state:
box has red.  tray has blue.  shelf has green.
Actions:
Swap box and tray.
Question:  What color is in tray?
Answer:  red
```

Factual recall uses three question-answer exemplars followed by the query, with answers capped at a small number of words:

```
Question:  In which country is the city of Osaka?
Answer:  Japan
```

A.21.2 THE GSM8K PROMPT

GSM8K is evaluated with one fixed worked exemplar followed by the test question, and answers are read from the #### marker, falling back to "answer is" and then to the last number in the continuation:

```
Below are math problems with detailed step-by-step
solutions.
Problem:  Natalia sold clips to 48 of her friends in April,
and then she sold half as many clips in May.  How many clips
did Natalia sell altogether in April and May?
Solution:  Let's think step-by-step.
1.  Clips sold in April:  48
2.  Clips sold in May:  48 / 2 = 24
3.  Total clips:  48 + 24 = 72
#### 72
Problem:  {question}
Solution:  Let's think step-by-step.
```

A.22 Compute and statistical reporting

Each pair runs on a single 48 GB GPU (RunPod A6000/A40/L40S) with donors in bf16, and layer pairs are re-derived per model. Task maps are trained with cross-entropy on the first answer token. The primary arithmetic map pools five seeds, while the symbolic, state-tracking, dose-response, and probe follow-up tables are single-seed. Intervals are Wilson (proportions), bootstrap (means), and across-seed t-intervals for the pooled task map.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016. arXiv:1610.01644.

András Balogh and Márk Jelasity. How not to stitch representations to measure similarity: Task loss matching versus direct matching. In *AAAI Conference on Artificial Intelligence*, 2025. arXiv:2412.11299.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2106.07682.

Nora Belrose, Zach Furman, Logan Smith, et al. Eliciting latent predictions from transformers with the tuned lens, 2023. arXiv:2303.08112.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.06609.

Jiaming Cheng, Subhransu Das, and Rajiv Ramnath. When does latent communication pay? a causal audit of relayed kv caches in multi-agent llms. *arXiv preprint*, 2026. arXiv:2608.04893.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2302.03025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems, 2021. arXiv:2110.14168.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.08600.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.03215.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2401.06102.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023. arXiv:2310.15916.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *International Conference on Machine Learning (ICML)*, 2024. PMLR 235; arXiv:2405.07987.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. arXiv:1705.03551.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition, 2025. arXiv:2502.00873.

Su-Hyeon Kim, Jiwan Mun, and Yo-Sub Han. One adapter pair per model: A universal activation interface for language models, 2026. arXiv:2608.09521.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019. arXiv:1905.00414.

Michael Lan, Philip H. S. Torr, Austin Meek, Ashkan Khakzar, David Krueger, and Fazl Barez. Quantifying feature space universality across large language models via sparse autoencoders, 2024. arXiv:2410.06981.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. arXiv:1411.5908.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, et al. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. arXiv:2408.05147.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2202.05262.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2209.15430.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.21272.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in/.

Narmeen Oozeer, Dhruv Nathawani, Nirmalendu Prakash, Michael Lan, Abir Harrasse, and Amirali Abdullah. Activation space interventions can be transferred between large language models, 2025. arXiv:2503.04429.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *Conference on Computational Natural Language Learning (CoNLL)*, 2023. arXiv:2311.04897.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. StateBridge: Training-free hidden-state alignment for latent communication in llm multi-agent systems. In *Conference on Language Modeling (COLM)*, 2026. arXiv:2608.13317.

Fernando Cardenas Piepereit. Architecture-dependent causal transfer of activation states across large language models, 2026. arXiv:2608.16347.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *International Conference on Machine Learning (ICML)*, 2025. PMLR 267:51094–51116; arXiv:2501.14082.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. arXiv:2004.07667.

Damian Smith, Harvey Mannering, and Antonia Marcu. Functional alignment can mislead: Examining model stitching. In *International Conference on Machine Learning (ICML)*, 2025. PMLR 267:55972–55998.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.15213.

Peiyan Zhang and Jason Xin. A negative result on cross-model activation transfer in a pythia multi-hop setting, 2026. arXiv:2606.03280.