Jet Expansions

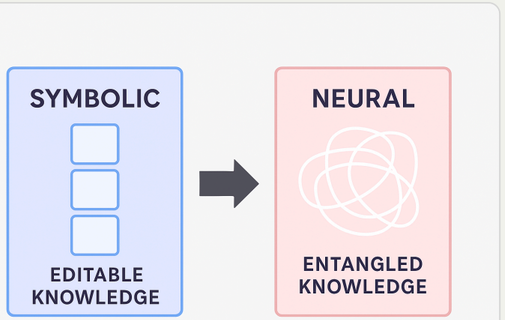
Restructuring LLM Computation for Inspection

Yihong Chen[1], Xiangxiang Xu[2], Pontus Stenetorp[3]
Sebastian Riedel[3], Luca Franceschi[4]

ICLR 2026

[1] OATML, University of Oxford, UK | [2] University of Rochester, USA
[3] AI Centre, University College London, UK | [4] Independent Researcher, Berlin, Germany

Behavior Editing Is Hard in LLMs



	Symbolic system	**LLM**
Remove a behavior	Delete the responsible rule	Diffuse finetuning updates
Add a new behavior	Insert a new rule	Diffuse finetuning updates
Control handle	Explicit and local	Implicit and distributed
Underlying unit	Addressable rule / fact	Entangled representation

In symbolic systems, behavior can be changed by editing explicit units. In LLMs, the relevant computation is distributed across entangled pathways, so model-level control becomes indirect, opaque, and hard to verify.

Motivation: Why Model Inspection?

For engineering, we often treat LLMs like Lego bricks: stack layers, swap modules, move on.

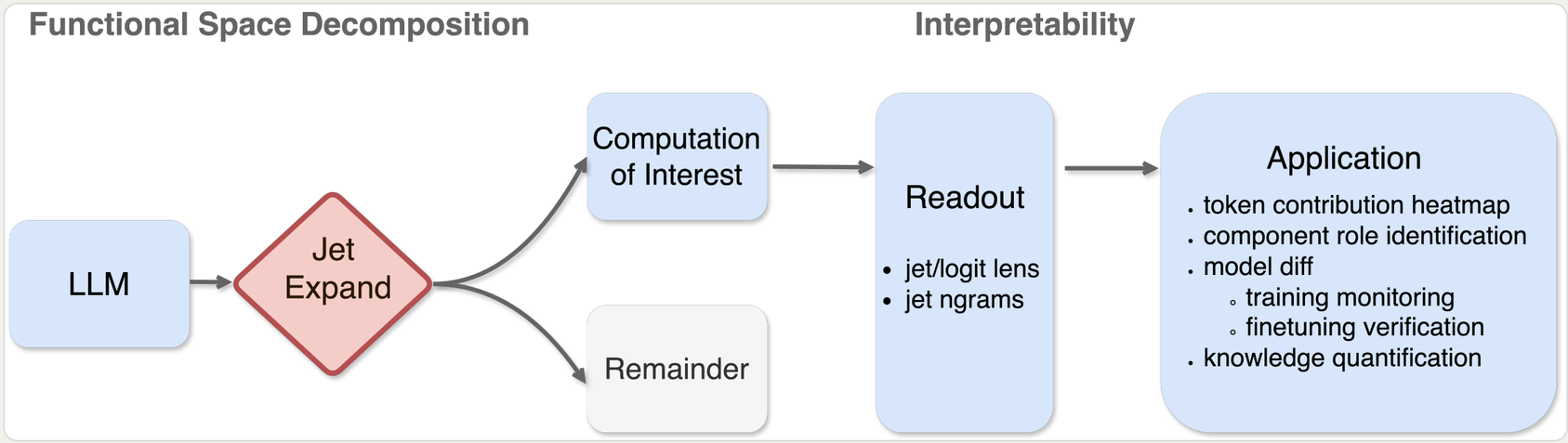
For interpretability, that breaks down. A layer is a **building brick**, not a **knowledge brick**.

So what do most interpretability methods do? They take one part we care about and leave the rest as remainder.

- one decoding view: logit lens
- one family of paths: path analysis
- one local center: approximation-based views

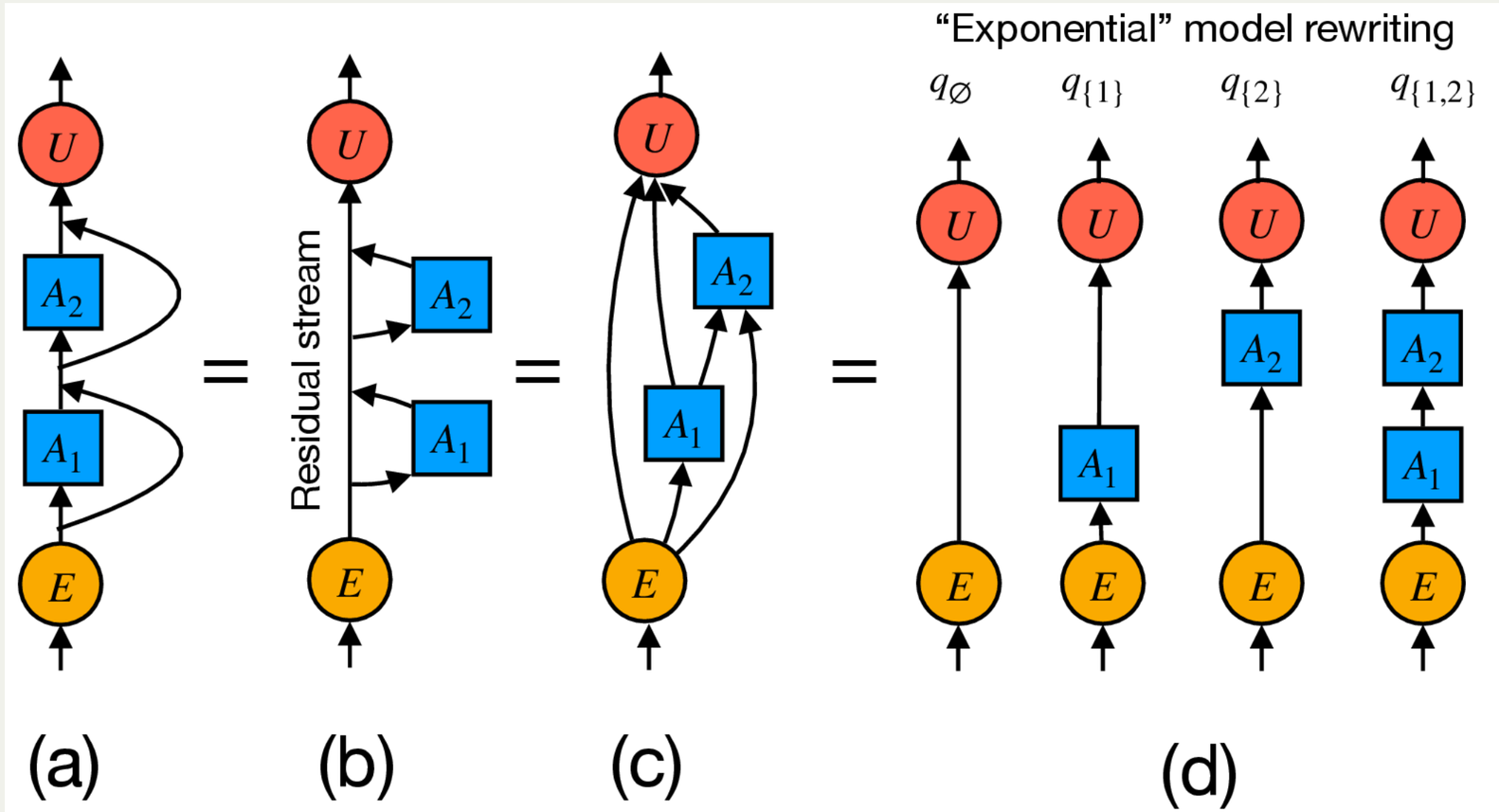
> **Jet expansions make this systematic:** cut out the computation of interest, keep the remainder explicit, and do this for many kinds of interpretability readouts in one framework.

Jet Expansions: Interest Part + Remainder



This is the main idea of the paper: given an LLM, jet expansion cuts the computation into an **explicit part we care about** and a **complementary remainder**. Once that split is explicit, we can read out lenses, n-grams, model diff, training dynamics, and knowledge mass from the selected pathways.

Linear Residuals: Exact Path Decomposition



If $\gamma_\ell(x) = A_\ell x$, then $f = U\big(\sum_{S\subseteq[L]} \prod_{\ell\in S} A_\ell\big)E = \sum_{S\subseteq[L]} f_S$. Each subset S is one skip/non-skip path, yielding an explicit 2^L-term decomposition; carving is as easy as taking paths of interest. But this clean separability breaks with nonlinearities.

Non-linear Case: Local Polynomial Expansion

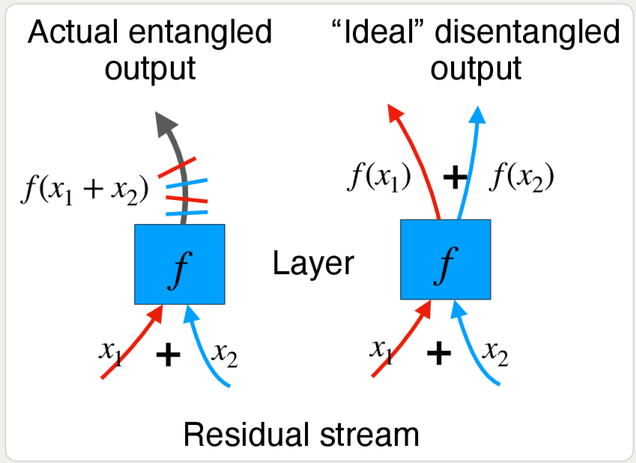
For $f \in C^{k+1}$ around basepoint x_0:

$$f(x) = f(x_0) + \sum_{j=1}^{k} \frac{1}{j!} D^j f(x_0)\,(x - x_0)^{\otimes j} + O(\|x - x_0\|^{k+1})$$

Jets: define the k-jet operator $J_k f(x_0)$ as the truncated polynomial component.

Interpretation: each derivative tensor isolates interactions of specific orders.

Non-linear Case: Local Disentanglement of Jets (Sketch)



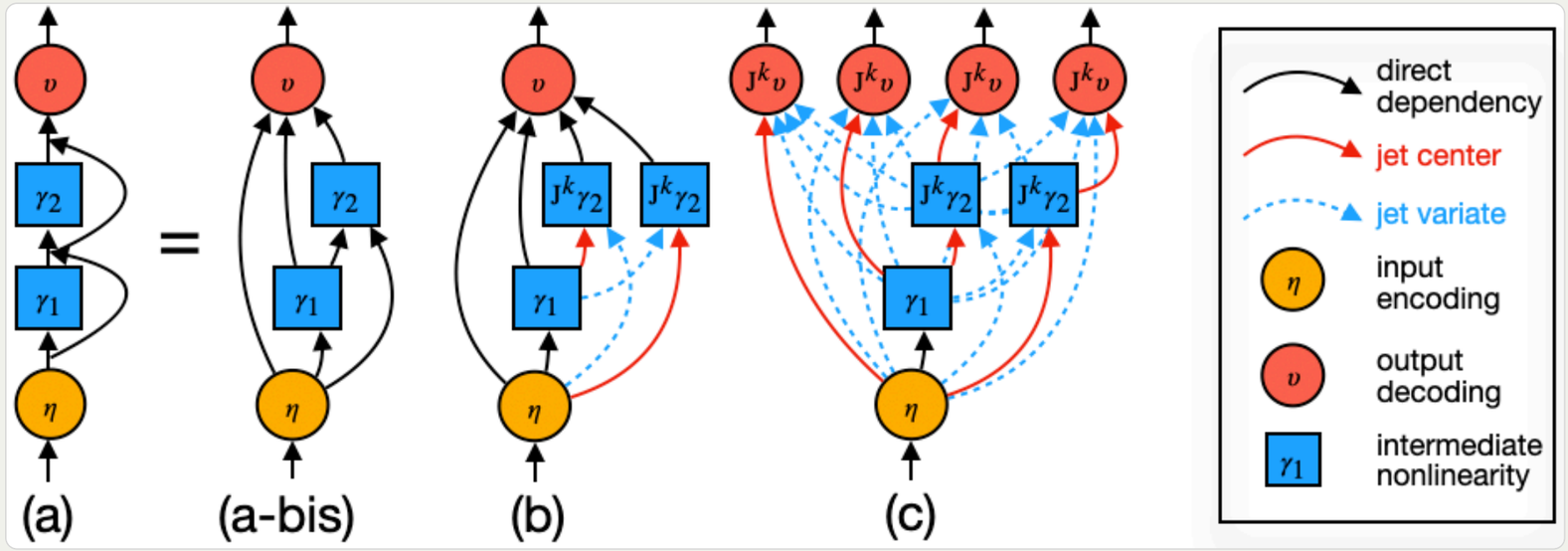
Setup. Let $\bar{x} = \sum_{i=1}^{N} x_i$.

Claim. For sufficiently small interaction radius r,

$$J_k f(\bar{x}) = \sum_{i=1}^{N} w_i \, J_k f(x_i) + O(r^{k+1})$$

Why this matters. Jets are approximately a convex combination of component jets, with controlled error. This gives a principled way to separate nested residual streams passing through one nonlinear block.

Non-linear Case: Recursive Jet Disentanglement



Apply local jet disentanglement at one block, pass the separated components to the next block, and repeat, yields a structured sum of component streams instead of one opaque nonlinear mixture

Pathway Case 1: Model Diff

Do carved pathways reveal transferred code knowledge?

Rank	LLAMA2-7B	CodeLLAMA-7B	CodeLLAMA-Python-7B
0	(_more, _than)	(_like, wise)	(_like, wise)
50	(_Now, here)	(_just, ification)	(_Like, wise)
100	(_system, atically)	(_in, _case)	(_all, udes)
150	(_all, erg)	(_get, ters)	(_no, isy)
200	(_on, ions)	(któber, s)	(output, ted)
300	(_other, world)	(_all, ud)	(Object, ive)
350	(_Just, ified)	(gebiet, s)	(_as, cii)
400	(_trust, ees)	(_Protest, s)	(_can, nab)
450	(_at, he)	(_deploy, ment)	(.transport, ation)
500	(_book, mark)	(Class, room)	(Tag, ging)
550	(_from, 而)	(_access, ory)	(_personal, ized)
600	(_WHEN, ever)	(_In, variant)	(_excess, ive)
650	(_where, about)	(_I, _am)	(_Add, itional)
700	(ag, ged)	(add, itionally)	(_**, kwargs)

Comparing Llama to CodeLlama, pathway bigram shifts surface code-specific patterns directly from the model, giving a pathway-level view of transferred knowledge.

Pathway Case 2: Model Evolve

Do carved pathways track learning during pretraining?

Rank	0K	100K	200K	300K	400K	555K
0	immortal	's	at least	&	&	&
1	ICUirling	at least	's	at least	its own	its own
2	ords architect	its own	&	its own	their own	their own
3	yaml Adam	okerly	your own	your own	at least	his own
4	231 next	VENT thanks	its own	their own	your own	make sure
5	clonal 条	iums	iums	more than	his own	your own
6	Charg@{	you're	you're	can't	2nd	2nd
7	avoir careless	Everything v	2nd	his own	more than	at least
8	HOLD worsening	erna already	you guys	2nd	make sure	more than
9	Horse dismant	'my	more than	make sure	can't	iums

On OLMo checkpoints, early stages show noisy pairs such as `yaml Adam`, while later stages surface coherent phrases such as `its own` and `make sure`.

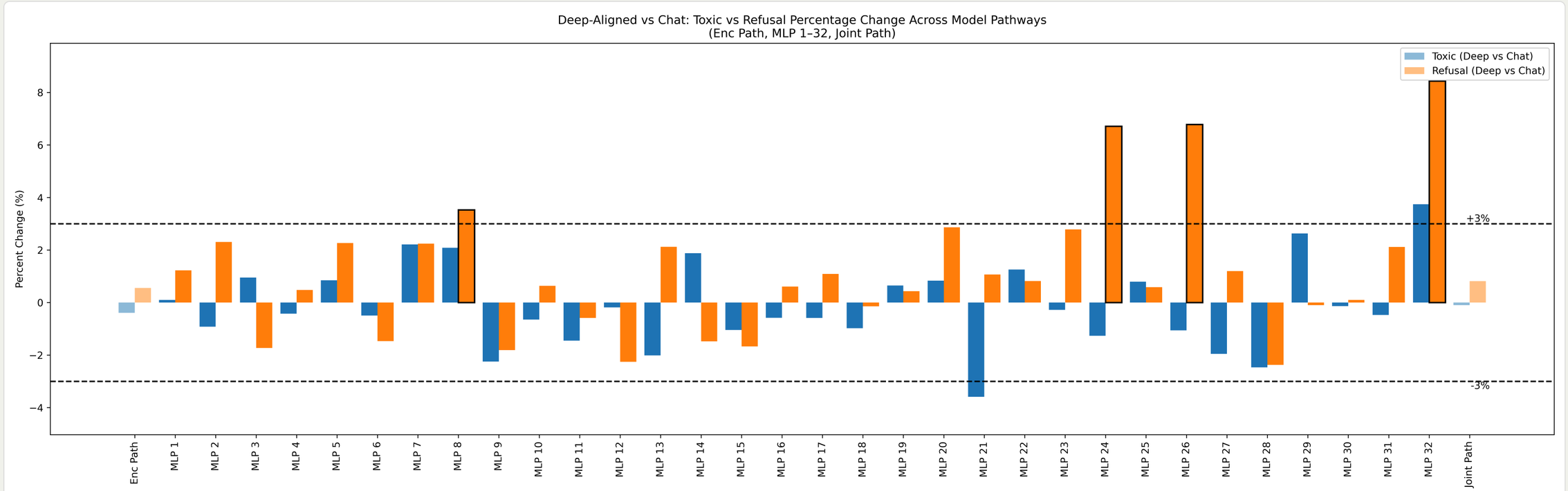
Pathway Case 3: Knowledge Quantification

Can carved pathways quantify toxic knowledge directly?

Model	ToxiGen Score Hartvigsen et al. (2022)	Jet Bi-grams Mass of "toxic" bi-grams	RTP Challenging Prompts			
			No	Very mild	Medium	Hard
Llama-2-7B	21.25	0.102	38%	49%	64%	88%
Llama-2-7B-chat	0.0	0.093	23%	35%	64%	84%

Jet n-gram mass gives a data-free index of toxic knowledge, which can be compared against ToxiGen-style behavior scores and RTP challenge prompts.

Alignment: Deep Removal or Masking?



Reading. Refusal-related mass increases more clearly than toxic mass disappears.

Takeaway. On these models, alignment often looks more like masking harmful continuations than deeply removing the underlying knowledge.

Why jets matter. This kind of distinction is visible from pathway-based knowledge mass, not only from surface behavior benchmarks.

Takeaways

1) Why a new interface?
LLM knowledge is entangled and does not live in addressable units, so model inspection needs more than prompt-based probing.

2) What jets do.
Jet expansions carve computation into explicit pathways plus remainder, turning interpretability into a functional decomposition problem.

3) Why it matters.
Those pathways support model diff, model evolve, and knowledge quantification directly in model space.

4) Safety takeaway.
This lets us ask whether alignment changed underlying knowledge or only masked outputs.