SPARSE AUTOENCODERS REVEAL THAT MOE EXPERTS EXHIBIT HUMAN-INTERPRETABLE FINE-GRAINED SPECIALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) models route each token to a few of many experts, yet what an individual expert is responsible for remains unsettled. Existing analyses count how often an expert is selected on domain-labeled corpora, an instrument that fixes the taxonomy in advance, stays coarse, and records selection rather than contribution. We instead study specialization in feature space by training a sparse autoencoder (SAE) on the output of an MoE layer. Since this output is a router-weighted sum of expert outputs and the SAE encoder is affine, the pre-activation of every latent decomposes exactly over the experts. Aggregated over a corpus, this gives each expert a dominant set of latents whose activation it principally drives. On layers 4, 30 and 46 of Qwen3-30B-A3B, contribution is far more concentrated than gating, most latents have only one or two dominators, and the overlap between dominant sets concentrates on a few expert pairs. Auto-interpreting these sets reveals a division of labor below coarse domains, with distinct experts for Java and front-end web code, or for geometry and for arithmetic and polynomials, while the coarse domains themselves take very unequal numbers of experts.

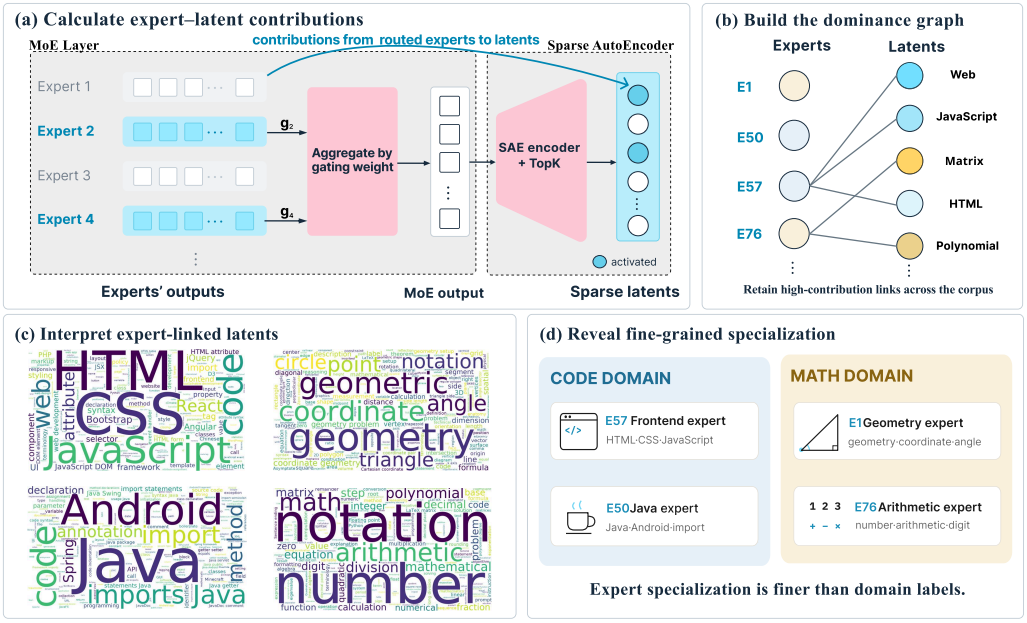


Figure 1: An MoE layer outputs a router-weighted sum of its active experts (left), and since the SAE encoder is affine, each latent's activation decomposes exactly over these experts (middle). Thresholding the contribution fractions averaged over a corpus gives the dominance graph (right), and auto-interpreting the latents an expert dominates describes its role (bottom), such as a Java expert (E50) and a front-end web expert (E57) at layer 30 of Qwen3-30B-A3B.

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.

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures underlie much of the recent progress in large language models. By routing each token to a small subset of experts, an MoE model scales its parameter count, and with it the knowledge and capacity it can store, while keeping per-token inference cost close to that of a far smaller dense model. Central to this design is the router, yet the mechanism it implements has received comparatively little scrutiny. How a trained model allocates tokens across experts, and hence what any individual expert comes to be responsible for, remains poorly understood.

The existing evidence is mixed. Some work reports that experts are domain-specialized, activating preferentially on data drawn from particular domains (Liu et al., 2024; Muennighoff et al., 2025). Another concludes that routing is unrelated to domains as humans conceive of them, and is instead driven by surface properties such as token identity (Jiang et al., 2024; Xue et al., 2024). What they do share is an instrument, namely counting how often each expert fires on corpora carrying human-assigned labels, such as GitHub for code or arXiv for academic. This instrument has four limitations: (a) The taxonomy is imposed by the analyst rather than recovered from the model; such a study can only ask whether the model's division of labor agrees with a partition chosen in advance, instead of the organization the model actually learned. (b) The labels available at corpus scale are coarse, typically a handful of categories such as code and math. This is far coarser than the division of labor one would expect among the tens to hundreds of experts in a single layer. (c) The label attaches to a corpus while the routing decision being measured is made per token. For example, a math problem statement may contain a great deal of story narrative framing, and counting every token of it toward "math" blurs precisely the distinctions the analysis is meant to detect. (d) Activation frequency records which experts were selected, without regard to the router coefficients that weight them. An expert can be selected on a token and still account for little of what the layer computes there.

We propose to study expert specialization in the feature space of the model layer. We train a Sparse AutoEncoder (SAE; Huben et al., 2024) on the output of an MoE layer. SAE latents approximate the features the model itself uses, and are discovered from its activations rather than stipulated in advance. Moreover, they also admit a natural decomposition into the contributions of individual experts. The output of an MoE layer is a router-weighted sum of the outputs of the active experts; because the SAE encoder is affine in that sum, each latent's pre-activation decomposes exactly over the experts that produced it. This yields, for every (expert, latent, token) triple, the share of the latent's activation contributed by that expert, which we aggregate over a corpus into an expert-level contribution profile. Figure 1 gives an overview, from the decomposition of a latent's activation over experts to the account of an expert's role that it supports.

Our approach differs from prior analyses along two axes. In the unit of analysis, we replace analyst-defined domains with latents the SAE learns unsupervised, so the resulting account of specialization is not constrained by a taxonomy chosen in advance, and it is finer-grained than any label set that can be assigned at corpus scale. In resolution, we replace corpus-level selection frequency with a per-token, per-expert contribution score, which both avoids attributing every token of a document to a single domain and measures functional contribution rather than mere co-occurrence.

We instantiate this analysis on layers 4, 30, and 46 of Qwen3-30B-A3B (Yang et al., 2025), chosen to cover shallow, mid-late, and deep positions in the network. Our first observation is that contribution is highly concentrated. Most latents receive the bulk of their activation from a small number of experts. Building on this, we say that an expert *dominates* a latent when it is a principal contributor to that latent's activation, and we collect, for each expert, the set of latents it dominates. Interpreting the latents in an expert's dominant set therefore describes what that expert does in the model's own vocabulary rather than in categories chosen in advance. We obtain such interpretations by auto-interpreting each latent from activating contexts and logit effects (Section K).

Most pairs of dominant sets share almost nothing, the overlap that exists concentrates on a few pairs of experts, and about half of each expert's set is its own. Reading these sets yields two further findings. First, experts specialize at a granularity well below that of conventional domain labels; for instance, we identify experts whose dominant set consists primarily of latents on front-end web code rather than on code in general. Second, summarizing the sets into coarse domains shows that the allocation of experts across domains is strongly unequal. Together these show that the experts of this model do divide labor, but along feature-level distinctions rather than the domain-level partitions under which specialization has previously been sought.

2 RELATED WORK

Expert specialization in MoE models. Prior analyses report either domain-specialized experts (Liu et al., 2024; Muennighoff et al., 2025) or routing driven by surface properties such as token identity (Jiang et al., 2024; Xue et al., 2024), and both rest on selection counts over domain-labeled corpora, whose limitations we discuss in Section 1. A recent work leaves the domain level behind. Motivated by k-sparse probing, which finds the neurons inside an expert less polysemantic than those of a comparable dense network, it labels hundreds of whole experts automatically and concludes that they are fine-grained task experts rather than domain specialists or token-level processors (Herbst et al., 2026). Its specialization score is computed against clusters of the model's own output embedding, which keeps the categories model-derived, but the number of clusters is a hand-set grid of a few values, so the score reports which of the chosen resolutions fits an expert best rather than the level at which the model has drawn its division of labor. Our units are learned by the SAE without supervision, so an expert's dominant set is read off instead of being matched against a partition fixed in advance, and the level at which that set is coherent is itself one of our results.

SAEs, interpretability and tracing. Our unit of analysis comes from dictionary learning. Since a model holds more features than it has dimensions, placing them in superposition (Elhage et al., 2022), its neurons are polysemantic, and an SAE instead learns an overcomplete sparse dictionary whose directions are considerably more monosemantic (Huben et al., 2024), a construction since scaled to frontier models and to millions of latents (Templeton et al., 2024; Gao et al., 2025). We adopt the top-k variant of Gao et al. (2025) in Section 3.2 and read latents by auto-interpretation from activating contexts and logit effects (Section K), following Paulo et al. (2024). Latents also serve to trace computation, in sparse feature circuits (Marks et al., 2025) and attribution graphs (Ameisen et al., 2025), which take latents as nodes and attributions between them as edges. Our attribution runs along a different axis, because those works trace a computation across layers and components whereas we decompose the activation of one latent within a single layer over the experts that produced it. The decomposition of the pre-activation is also exact rather than approximate, since Equation (5) follows algebraically from the linearity of the layer output in the expert outputs.

3 METHOD

3.1 THE MOE BLOCK LINEARLY COMBINES EXPERT OUTPUTS INTO THE LAYER OUTPUT

Consider an MoE LLM with hidden dimension d, where one MoE block contains N experts. Given a token sequence $x = (x_1, \ldots, x_T)$, let $x_{1:t} = (x_1, \ldots, x_t)$ be its prefix and $h(t) \in \mathbb{R}^d$ be the corresponding input to the MoE block. The router scores this input, takes the top-k experts and renormalizes their weights. The router assigns a non-negative weight to every expert, the weight of the j-th expert being $g_j(t) \in [0, 1)$. Only k of the $g_j(t)$ are positive, and satisfy $\sum_{j=1}^{N} g_j(t) = 1$. We call an expert with $g_j(t) \neq 0$ an activated expert. Denoting by $\mathcal{A}(t)$ the set of ids of the activated experts and by $u_j(t) \in \mathbb{R}^d$ the output of the j-th expert, the output of this layer is

$$y(t) = \sum\nolimits_{j \in \mathcal{A}(t)} g_j(t)\, u_j(t) \in \mathbb{R}^d. \tag{1}$$

3.2 ENCODING THE MOE OUTPUT INTO A LATENT SPACE WITH AN SAE

We train an SAE whose objective is to reconstruct y, with latent dimension $m \gg d$. Its encoding and decoding are

$$f(t) = \text{TopK-ReLU}_{\kappa}\big(W_{\text{enc}}\, y(t) + b_{\text{enc}}\big), \tag{2}$$

$$\hat{y}(t) = W_{\text{dec}}\, f(t) + b_{\text{dec}} \approx y(t), \tag{3}$$

where $W_{\text{enc}} \in \mathbb{R}^{m \times d}$, $W_{\text{dec}} \in \mathbb{R}^{d \times m}$, and the columns of W_{dec} have unit norm. The i-th component $f_i(t)$ of $f(t)$ is the activation magnitude of latent i on this prefix, and at most κ components are non-zero. We denote the activated set by $\mathcal{F}(t) = \{\, i \mid f_i(t) > 0 \,\}$.

3.3 THE CONTRIBUTION OF AN EXPERT TO A LATENT

We observe that the activated part of a latent is an affine transformation of y. Let $W_{\text{enc},(i)} \in \mathbb{R}^d$ denote the i-th row of the encoder weight and $b_{\text{enc},(i)}$ the i-th entry of the encoder bias. Then, for

an activated latent i,

$$f_i(t) = W_{\mathrm{enc},(i)} \cdot y(t) + b_{\mathrm{enc},(i)}. \tag{4}$$

Substituting Equation (1) into Equation (4) gives

$$f_i(t) = \sum\nolimits_{j \in \mathcal{A}(t)} g_j(t)\, W_{\mathrm{enc},(i)} \cdot u_j(t) + b_{\mathrm{enc},(i)}. \tag{5}$$

Equation (5) decomposes the activation of a latent exactly into the encoder bias plus the sum of the contributions of several experts. It follows that, for an expert j to which $h(t)$ is routed and an activated latent i, the positive contribution of j to i is

$$c(j, i, x_{1:t}) = \mathrm{ReLU}\big(g_j(t)\, W_{\mathrm{enc},(i)} \cdot u_j(t)\big). \tag{6}$$

A negative contribution strong enough to suppress a latent leaves it inactive and is never observed, so any statistics over negative contributions would be systematically biased. We therefore keep only the positive part in Equation (6) and use it exclusively in all subsequent analyses.

3.4 Computing the contribution fraction

On a single prefix, the relative magnitude of an expert's contribution to a latent reflects whether this expert is important for that latent. Normalizing c over the set of activated experts, we obtain

$$C(j, i, x_{1:t}) = \frac{c(j, i, x_{1:t})}{\sum_{j' \in \mathcal{A}(t)} c(j', i, x_{1:t})} \tag{7}$$

for $j \in \mathcal{A}(t)$ and $i \in \mathcal{F}(t)$, and $C(j, i, x_{1:t}) = 0$ otherwise. We are interested in which latents an expert is generally better at contributing to, and therefore take the expectation of C over the entire dataset X, obtaining

$$\mathcal{C}(j, i) = \mathbb{E}_{(x,t) \sim \mathcal{U}\{(x,t)\,|\,x \in X,\ 1 \le t \le |x|\}} \left[C(j, i, x_{1:t}) \right]. \tag{8}$$

Finally, in order to assess whether an expert is generally important for a latent, we normalize $\mathcal{C}$ over the set of experts,

$$s(j, i) = \frac{\mathcal{C}(j, i)}{\sum_{j'=1}^{N} \mathcal{C}(j', i)}. \tag{9}$$

We call Equation (9) the contribution fraction of j to i.

3.5 Dominance graph

Given a threshold τ, $s(j, i) \ge \tau$ indicates that j generally makes a substantial contribution to i; in this case we say that j dominates i. We define the dominant set of an expert j to be the set consisting of all latents dominated by j, denoted $\mathcal{S}(j) = \{\, i \mid i \text{ is dominated by } j \,\}$.

We construct a bipartite graph $G(U, V, E)$, whose left vertex set U is the expert set and whose right vertex set V is the latent set of the SAE. Its edge set is $E = \{\, (j, i) \mid j \in U,\ i \in \mathcal{S}(j) \,\}$. We call this bipartite graph the dominance graph.

4 Qwen3-30B-A3B has specialized experts

We apply Section 3 to layers 4, 30, and 46 of Qwen3-30B-A3B, which we select to represent the model's shallow, mid-late, and deep-layer behaviors. The model has 48 layers in total, where each MoE layer of this model contains $N = 128$ experts, with $k = 8$ experts activated per token and no shared experts. We train one individual SAE with latent size of 2^{20} for each of the three layers. Statistics are computed on a validation set of 1.5B tokens that shares the same distribution as the SAE training set and has no data overlap with it. Every latent of each SAE activates at least once on this set, so $s(j, i)$ is defined for all latents.

4.1 Contribution is more concentrated than gating

The contribution defined in Equation (6) carries $g_j(t)$ as a factor, and the gating weights are themselves unevenly distributed. The router takes the top-k experts and renormalizes their weights, so higher-ranked experts naturally receive larger $g_j(t)$. Before adopting this statistic we therefore have

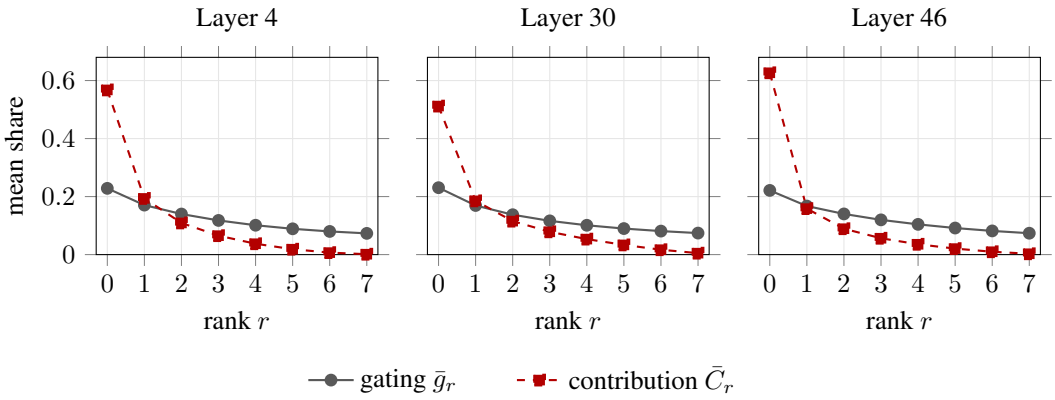


Figure 2: Rank profiles of the gating weight and of the contribution. $\bar{C}_r$ is markedly steeper than $\bar{g}_r$.

to rule out a trivial explanation, namely that the unevenness of the contribution across the activated experts is merely a direct projection of the unevenness of the gating, and has nothing to do with whether an expert has any preference over latents.

We compare the profiles that the two quantities form once aggregated by rank. For each prefix $x_{1:t}$ we sort the gating weights of the 8 activated experts in descending order, take the value in position r ($r = 0, \ldots, 7$), and average it over the entire validation set; we denote the result by $\bar{g}_r$. Analogously, for each (prefix, activated latent) pair $(x_{1:t}, i)$ with $i \in \mathcal{F}(t)$, we sort the 8 values $C(j, i, x_{1:t})$ in descending order, take the value in position r, and average it over all such pairs; we denote the result by $\bar{C}_r$. Note that position r need not correspond to the same expert from one case to the next, because what we aggregate is the rank. The two profiles are directly comparable, because the renormalization performed by the router together with the definition in Equation (7) gives $\sum_{r=0}^{k-1} \bar{g}_r = \sum_{r=0}^{k-1} \bar{C}_r = 1$. The two profiles thus distribute one and the same total.

Figure 2 reports the results for the three layers, and in all three $\bar{C}_r$ is markedly steeper than $\bar{g}_r$. Taking layer 30 as an example, the first three gating values are 0.231, 0.169 and 0.137, decaying gently, whereas the first three contribution values are 0.512, 0.185 and 0.115, the first of them alone accounting for more than half of the total. The two curves cross between $r = 1$ and $r = 2$, beyond which the contribution stays below the gating throughout; by $r = 7$ we have $\bar{C}_7 = 0.005$ while $\bar{g}_7 = 0.074$. Layers 4 and 46 exhibit the same shape.

The ReLU in Equation (6) could also steepen $\bar{C}_r$, since a truncated expert receives $c = 0$ and the remaining experts share the whole unit among themselves. The truncation is mild, however. Of the 8 activated experts of a (prefix, activated latent) pair, on average only 2.09, 1.29 and 1.64 are truncated at layers 4, 30 and 46, so most activated experts keep a positive contribution and the truncation is not the main source of the steepness.

The concentration of the contribution is therefore not an artifact of the top-k routing structure. While the gating profile decays gently, the contribution is highly concentrated on a small number of experts.

4.2 The dominance graph reveals the structure of latent dominance

We build the dominance graph of Section 3.5 with $\tau = 0.2$. This threshold sits far above $1/N \approx 0.008$, the value an expert would receive if a latent's activation were sourced evenly from all $N = 128$ experts, so an edge records a contribution that is significant rather than merely nonzero.

We first look at the latent-side degree, that is, the number of experts dominating a latent, whose full distribution is given in Section C. In all three layers over half of the 2^{20} latents have exactly one dominator, namely 57.4% at layer 4, 62.1% at layer 30 and 56.7% at layer 46. Including the latents with two dominators accounts for 89.9%, 85.1% and 92.7% respectively, whereas degree three or above never exceeds 3.7%, and the ceiling of five permitted by τ is reached in no layer at all. A latent with no dominator is one whose activation is sourced comparatively evenly, no single expert clearing τ; such latents remain a minority, at 7.0%, 13.6% and 3.7%. Latent dominance is therefore concentrated. The activation of a latent is typically attributable to one or two experts rather than spread over the eight that the router selects or the universal set of experts.

Table 1: Pairwise Jaccard similarity over the 8128 expert pairs of each layer. The intersection in the top 10% of pairs is the share of the total intersection carried by the 813 pairs with the largest J. The lower block describes the pairs with $J > 0.05$, a single partner means exactly one other expert above this threshold; a connected group is a connected component of the graph formed by these pairs.

	Layer 4	Layer 30	Layer 46
Mean J	1.73×10^{-3}	1.57×10^{-3}	2.12×10^{-3}
Median J	1.06×10^{-4}	1.01×10^{-4}	1.37×10^{-4}
Pairs sharing no latent	37.8%	40.5%	37.5%
Intersection in top 10% of pairs	83.8%	78.5%	81.0%
Pairs with $J > 0.05$	25 (0.31%)	20 (0.25%)	46 (0.57%)
Experts involved (out of 128)	35	34	60
with a single partner	22	28	36
Largest connected group of experts	10	4	8
Largest J	0.138	0.098	0.317

On the expert side, every one of the 128 experts dominates a nonempty set of latents in every layer, and the ten highest-degree experts hold at most 30.1% of all edges. Dominance is thus spread across the whole layer instead of being absorbed by a few hub experts (Section C).

4.3 The overlap of dominant sets concentrates on a few expert pairs

Section 4.2 shows that a latent usually has one or two dominators. This alone does not make the dominant sets disjoint, since the same few experts could still co-dominate many latents. We therefore measure the overlap directly by the Jaccard similarity of every pair of dominant sets, $J(j, j') = \frac{|\mathcal{S}(j) \cap \mathcal{S}(j')|}{|\mathcal{S}(j) \cup \mathcal{S}(j')|}$, over all $\binom{128}{2} = 8128$ unordered expert pairs of a layer.

Table 1 summarizes the distribution of J, and the full matrices, which look alike across the three layers, are shown in Section D. The median of J stays near 10^{-4} in all three layers, and between 37.5% and 40.5% of the pairs share no latent at all.

What the degrees leave open is how this total is spread over the pairs, and the spread is strongly skewed. The mean of J is an order of magnitude above the median, and the tenth of the pairs with the largest J carries 83.8%, 78.5% and 81.0% of the total intersection at the three layers. Most pairs thus share almost nothing, and the overlap that exists sits in a right tail, where at most 0.57% of the pairs exceed $J = 0.05$ and the largest J reaches 0.138, 0.098 and 0.317.

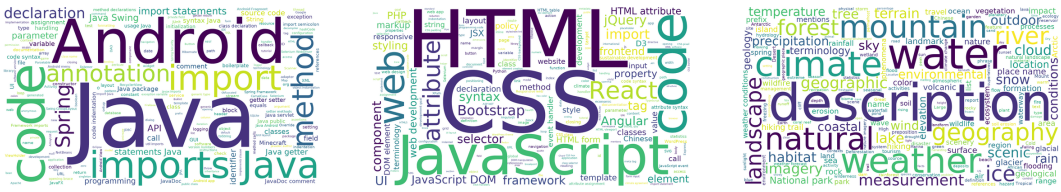
The tail is organized into pairs rather than groups. Of the experts that have a partner above $J = 0.05$, most have exactly one, the graph linking these pairs splits into connected components of at most 10 experts, and 26, 33 and 30 pairs of experts are each other's nearest neighbour by J. The paired sets are not nested, since a pair above $J = 0.05$ shares a median of 16% to 18% of the smaller of its two sets. On average between 46% and 59% of an expert's dominant set is dominated by that expert alone. Section 5.3 returns to the pairs in the tail once the experts are labeled.

5 Label experts with latent explanations

Section 4 establishes that every expert holds a dominant set of latents, that about half of this set is dominated by that expert alone, and that substantial overlap occurs only between a few pairs of experts. The set attached to an expert is therefore a description of that expert, so interpreting the latents it contains tells us which features the expert is chiefly responsible for. Section 4.2 and Section 4.3 further show that the structure of dominance and its trends are the same at layers 4, 30 and 46, and we therefore report only layer 30 in the rest of this section.

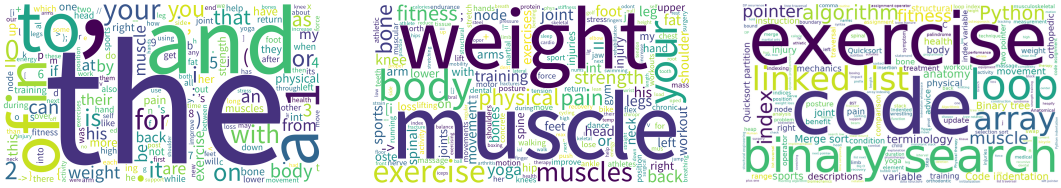
5.1 Dominant-set explanations carry more information than tokens

For every expert we sample 2000 latents uniformly from its dominant set, taking the whole set when it holds fewer than 2000 latents. Each sampled latent is interpreted by GPT-5.5 from activating contexts and logit effects, following the pipeline described in Section K. Reading the resulting explanations expert by expert, we find that for most experts they concentrate on a few themes. To make this concentration visible at a glance, we render the explanations of each expert as a word cloud, in which the size of a word is its frequency among the explanations of that expert's dominant set. Figure 3 shows the explanation word clouds of experts 50, 57 and 93 of layer 30 (a Java expert,



(a) E50: Java and Android (b) E57: frontend web (c) E93: geography and weather

Figure 3: Explanation word clouds of three experts of layer 30. Word size is the frequency among the explanations of the latents sampled from the expert's dominant set.



(a) routed tokens (b) routed tokens, MI reweighted (c) latent explanations

Figure 4: Word clouds of expert 72 of layer 30, built from (a) routed tokens weighted by gating mass, (b) the same tokens reweighted by their contribution to mutual information without the top five function words, and (c) latent explanations. The algorithm theme of (c) is barely readable in (a) and (b).

a front-end web expert and an expert on natural geography and weather), and the word clouds of the remaining experts are collected in Section E.

We compare this construction with a word cloud drawn directly from the tokens routed to an expert. Ours differs in two respects. The dominant set keeps only the latents the expert principally drives, which leaves less unrelated material in the picture, and an explanation is written from the context around the activating positions, which carries more than the token alone. Figure 4(a) gives the token word cloud of expert 72 of layer 30, in which a word is weighted by the gating mass the expert accumulates on it over the corpus. The words it puts forward are *the*, *and*, *of* and other function words. Such function tokens do carry varied and rich information (Zhang et al., 2025), but that information resides in the features activated at their positions and cannot be read off the tokens themselves, so at the token level they contribute nothing but noise.

We remove the five most frequent function words[1] and reweight each remaining token by its contribution to the mutual information between the expert and the token. Figure 4(b) is the result. It is more informative than Figure 4(a), yet part of what the expert does is still lost. Figure 4(c) is the explanation word cloud of the same expert, which shows that expert 72 is associated with two themes at once, exercise and health, and algorithms and data structures. The second theme is barely readable in Figure 4(b), because the latents that carry it activate only at particular positions within an algorithm. Latent F97783, for instance, activates only at the <= operator of a binary search, and it is its explanation, "Binary search loop condition", that ties it to binary search. On a token word cloud this latent can contribute no more than the symbol <=, whose origin and meaning are left open.

5.2 Experts divide labor below the level of coarse domains

Existing analyses of expert specialization ask whether experts align with domains such as code. Our explanation word clouds show that the model does not stop at this granularity. Inside a single coarse domain we find experts that hold nearly disjoint parts of it, and the part each one holds is a recognizable subfield rather than an arbitrary slice.

Consider the code domain, for instance. Figure 3(a) and (b) show the explanation word clouds of experts 50 and 57, which both work on code. Expert 50 is a Java expert. The words that take the largest font are *Java*, *Android* and *imports*, and the latents behind them are such things as "Java import statements", "JDBC generated keys" and "ButterKnife Android imports". Expert 57 is a front-end web expert, carried by *HTML*, *CSS* and *JavaScript* together with the framework names

[1] *the*, *of*, *to*, *and*, *a*

Table 2: Share of latent explanations judged by GPT-5.5 to belong to each theme at layer 30, among the latents sampled from the expert's dominant set (Share) and among 2000 latents drawn uniformly from the pooled samples of the other 127 experts (Random).

Expert	50	57	1	76
Theme	Java development	Front-end development	Geometry	Arithmetic and polynomial
Share	86.7%	78.6%	78.1%	59.1%
Random	1.9%	3.5%	2.0%	6.7%

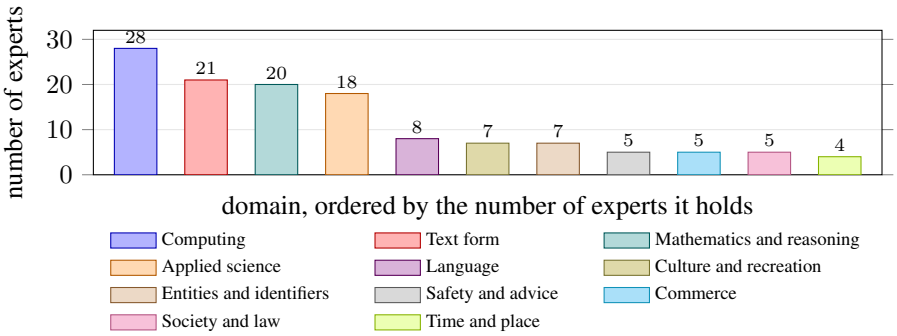


Figure 5: Number of experts in each of the eleven domains of layer 30. The domains and their names are produced by GPT-5.5 from the expert labels alone. Section I lists the experts of every domain.

React, *Bootstrap* and *jQuery*, and by latents about DOM events, CSS selectors and responsive layout.The same holds inside mathematics. Section E gives experts 1 and 76. Expert 1 is a geometry expert, dominated by *geometry*, *coordinate*, *triangle* and *angle*. Expert 76 instead concentrates on arithmetic and polynomial computation, with *polynomial*, *matrix*, *division* and *decimal* in the largest font, over latents such as "Polynomial long division", and "arbitrary-precision integers".

To put a number on the separation, we use GPT-5.5 to judge whether each latent explanation of an expert belongs to the corresponding theme, with the prompt reproduced in Section F. Table 2 reports the share of each expert's explanations falling in the corresponding theme among the latents sampled from its dominant set. For example, 86.7% of the latent explanations in expert 50's dominant set relate to Java development. Against a random sample of 2000 latents drawn from the dominant sets of the other 127 experts, the share of the same theme falls to between 1.9% and 6.7%.

5.3 Specialization is unevenly allocated across domains

Section 5.2 takes the experts one at a time and reads off the theme that each dominant set is chiefly about. Here we change the unit of measurement and ask how the layer distributes its experts over coarse domains. We do not fix the domains beforehand. Instead, we recover them by labeling each expert from its dominant-set word cloud and grouping those labels.

The procedure has two steps, carried out by GPT-5.5. We first name each expert. We hand the model the words of one expert in the wordcloud together with their weights and ask for a short label naming what that expert is responsible for, with the prompt in Section G. For instance, expert 50 receives "Java and Android programming" and expert 72 receives "coding algorithms and musculoskeletal fitness", which recovers from the word cloud alone the two themes that Section 5.1 reads off it. We then group the experts. A single call receives all 128 labels at once and returns a partition of experts together with a name for each part, with the prompt in Section H. We call these parts domains.

Eleven domains come out of the 128 experts of layer 30, and Figure 5 gives the number of experts each of them holds. Section I lists every expert with its label and its domain.

The domains are internally diverse. Besides the Java and front-end experts of Section 5.2, computing holds experts on competitive programming (9, 40 and 118), system administration (51) and Python API documentation (113), each a different part of working with code.

The pairs of experts whose dominant sets overlap most (Section 4.3) fall inside domains. Of the 20 pairs with the largest J at layer 30, 19 join two experts of the same domain, whereas only 12.8% of all expert pairs do. The two experts of such a pair cover neighbouring ground, as with experts 83 and 105 on punctuation, 77 and 107 on mojibake, or 57 and 69 on JavaScript web programming. The

one exception, experts 11 and 95, links LaTeX algebra in mathematics and reasoning with chemical and algebraic notation in applied science. Each label is written from the whole dominant set, so the shared latents enter both labels of a pair. They make up a median of only 12.3% of the dominant sets of these experts, however, so most of what each label is written from is not shared.

The allocation across domains is strongly unequal. Computing takes 28 of the 128 experts and mathematics and reasoning takes 20, so these two domains alone hold 37.5% of the layer. Most of the labels in computing name code, the remainder covering such things as cybersecurity and software platforms, and mathematics and reasoning also takes in statistics, parts of the physical sciences and problem-solving reasoning. At the other end the model spends very little on areas that a human taxonomy would list in their own right. Art occurs in one label out of 128, that of expert 54, where it sits beside animals inside culture and recreation.

Three of eleven domains (text form, language, and entities and identifiers) are not subject areas, and together they take 36 experts, or 28.1% of the layer. The largest of them, text form with 21 experts, is the second largest domain of the layer and covers document structure, punctuation, delimiters and encoding artifacts. All three are formed from the labels rather than requested. A taxonomy of subject matter alone would have to spread these 36 experts over all of its categories, and would recover none of these domains as a unit.

One plausible source of the imbalance among the topical domains is the composition of the pretraining data, in which code and mathematics take a share well above the one they occupy in general text. Llama 3, for instance, reports a final data mix of roughly 25% mathematical and reasoning tokens and 17% code tokens against 50% of general knowledge (Grattafiori et al., 2024).[2] On this reading the allocation of experts across domains reflects the proportions of the corresponding domains in the training data. The model draws finer distinctions inside a domain that occupies more of the data, and spends more of its experts on holding them.

6 Conclusion

We have proposed to study expert specialization in the feature space of an MoE layer instead of in a taxonomy of domains fixed in advance. A layer output is a router-weighted sum of the outputs of the activated experts and the SAE encoder is affine in that output, so the pre-activation of every latent decomposes exactly over the experts that produced it. Tracing latents back through this decomposition yields, for each expert, the dominant set of latents whose activation it principally drives, which describes the expert in the model's own vocabulary rather than in categories we supply.

We apply this analysis to layers 4, 30 and 46 of Qwen3-30B-A3B, and our findings are the following.

- The contribution profile is markedly steeper than the gating profile that produces it, and the activation of a latent is typically attributable to few experts rather than to all those that the router selects.
- Read from the expert side, most pairs of dominant sets share almost nothing, the overlap that exists concentrates on a few pairs of experts, which at layer 30 almost always belong to the same domain, and about half of each expert's set is its own.
- Interpreting the latents of these sets shows that the experts are specialized, and that the specialization sits below the granularity of conventional domain labels.
- Summarizing each expert's dominant set into a short label and grouping those labels into coarse domains shows that the allocation of experts across domains is uneven.

The analysis needs only the expert outputs and an SAE on the layer output, so it applies to any MoE layer whose output is a weighted sum of its experts, and can compare how routing objectives and load balancing losses shape the division of labor. The dominant sets also yield a testable hypothesis for expert pruning, merging and feature-level steering, namely that removing or editing an expert chiefly affects the latents it dominates. We discuss the limitations of this work in Section A.

[2]The Qwen3 technical report does not disclose its data mixture, so we cannot check this on the model we study. Raising the proportion of code, mathematics and reasoning data is nonetheless common practice in current pretraining pipelines.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 6(16318-16352):1, 2025.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *Advances in Neural Information Processing Systems*, 38:82318–82355, 2026.

Tom Conerly, Hoagy Cunningham, Adly Templeton, Jack Lindsey, Basil Hosmer, and Adam Jermyn. Dictionary learning optimization techniques. Transformer Circuits Thread, Circuits Updates — January 2025, January 2025. URL https://transformer-circuits.pub/2025/january-update/index.html#DL. Accessed: 2026-07-08.

Boyi Deng, Xu Wang, Yaoning Wang, Yu Wan, Yubo Ma, Baosong Yang, Haoran Wei, Jialong Tang, Huan Lin, Ruize Gao, et al. Qwen-scope: Turning sparse features into development tools for large language models. *arXiv preprint arXiv:2605.11887*, 2026.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jeremy Herbst, Stefan Wermter, and Jae Hee Lee. The expert strikes back: Interpreting mixture-of-experts language models at expert level, 2026. URL https://arxiv.org/abs/2604.02178.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Samuel Marks, Can Rager, Eric Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, volume 2025, pp. 23888–23923, 2025.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shaohua Zhang, Yuan Lin, and Hang Li. Memory retrieval and consolidation in large language models through function tokens. *arXiv preprint arXiv:2510.08203*, 2025.

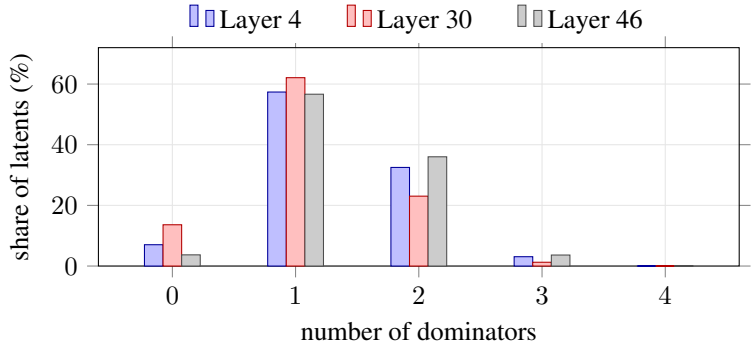


Figure 6: Latent-side degree distribution of the dominance graph, the share of the 2^{20} latents that are dominated by exactly $0, \ldots, 4$ experts. In every layer more than half of the latents have a single dominator. Degree 4 accounts for at most 0.014% and is not visible at this scale.

A Limitations and future work

- Our analysis inherits the imperfections of the SAE. The reconstruction of the layer output carries an error, and the latents are not a clean basis, since one feature can be split across several latents and can be absorbed into another (Chanin et al., 2026). A better decomposition of an MLP into features would sharpen the picture of an MoE layer that we draw here.
- The pretraining mixture of Qwen3 is not disclosed, and our statistics are computed on a corpus of our own choosing. What we measure is therefore routing on our data rather than on the data the model was trained on, and the reading of Section 5.3 that relates the allocation of experts to the proportions of domains in the training data remains a conjecture.
- We report a single model. Extending the analysis to more and to larger MoE models would show which of these observations are general, and the same instrument can be used to compare how different load balancing losses shape the division of labor among experts.

B Use of large language models

In this work, we used generative AI tools for implementing code, analyzing data, interpreting experimental results, and polishing the writing. Separately, GPT-5.5 is a component of our method, used to explain latents, judge theme membership, label experts and group them into domains, with every prompt reproduced in Sections F to H and K. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

C Degree distributions of the dominance graph

Figure 6 gives the latent-side degree distribution of the dominance graph of Section 4.2, the number of experts dominating a latent, from which the figures quoted in Section 4.2 are read off.

Figure 7 gives the expert-side degree of the dominance graph of Section 4.2, the number of latents an expert dominates, with the experts sorted by degree in ascending order. As in Figure 2 the horizontal axis is a rank rather than an expert identity. Every one of the 128 experts dominates a nonempty set of latents in every layer, the smallest degree observed being 362 at layer 4, 224 at layer 30 and 83 at layer 46. The distribution is right-skewed, its mean degree exceeding its median in all three layers (10786 against 8020, 9170 against 7395, and 11439 against 7835). But the skew is moderate. The ten highest-degree experts hold 24.6%, 20.1% and 30.1% of all edges, against the 7.8% they would hold under a uniform degree. Dominance is thus spread across the whole layer instead of being absorbed by a few hub experts.

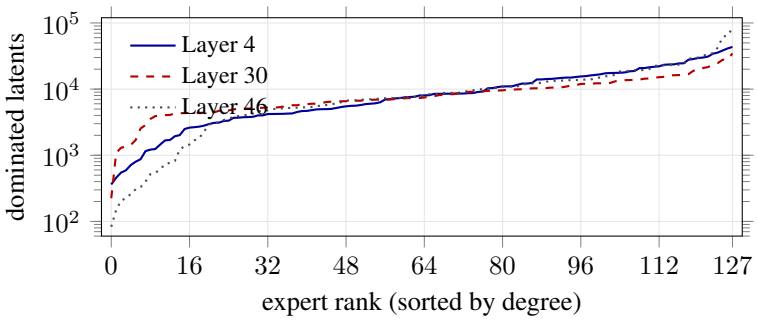


Figure 7: Expert-side degree of the dominance graph, the number of latents each expert dominates, with the 128 experts of each layer sorted by degree in ascending order. The horizontal axis is a rank rather than an expert identity. No expert has degree zero in any layer.

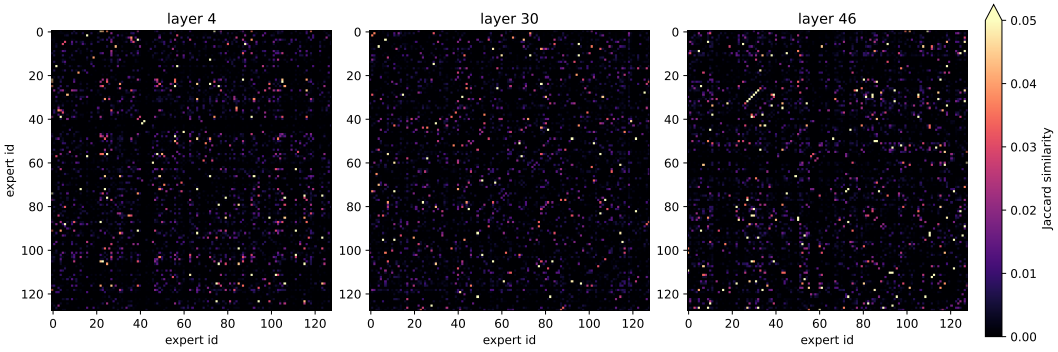


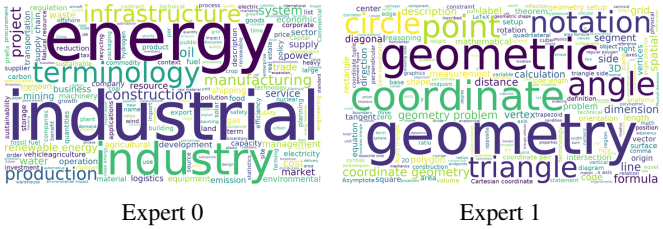
Figure 8: Pairwise Jaccard similarity of the expert dominant sets at $\tau = 0.2$, with the experts in their original index order on both axes. The three layers share one colour scale, which saturates at 0.05, and the diagonal is masked.

D Pairwise overlap of dominant sets

Figure 8 gives the full matrices of the pairwise Jaccard similarity summarized in Table 1. The three layers look alike. Each matrix is a dark field with a few isolated bright cells. No block of mutually overlapping experts appears, and no row is bright along its whole length, as an expert overlapping with all others would produce.

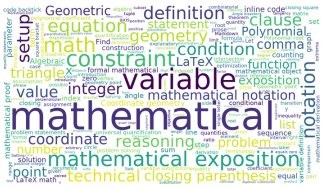
E Word clouds

Figure 9 to Figure 16 give the explanation word clouds of all 128 experts of layer 30, each built as in Figure 3.

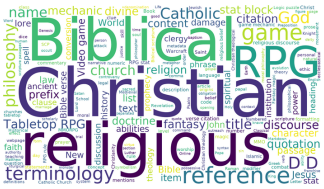


Expert 0 Expert 1

Figure 9: Explanation word clouds of experts 0 and 1 of layer 30. Each cloud is built from the explanations of the latents sampled from that expert's dominant set, and the size of a word is its frequency among those explanations.



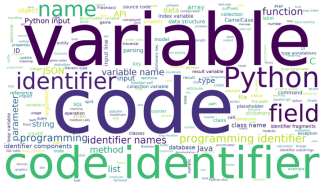
Expert 2



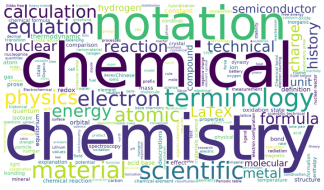
Expert 3



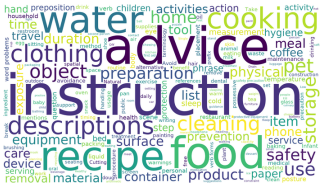
Expert 4



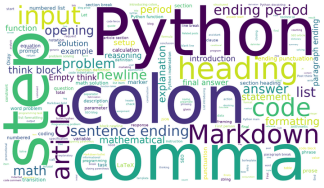
Expert 5



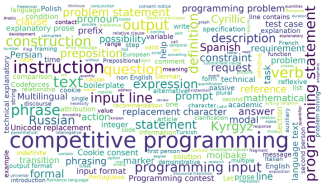
Expert 6



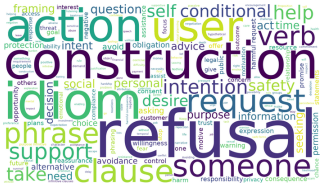
Expert 7



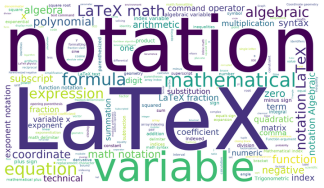
Expert 8



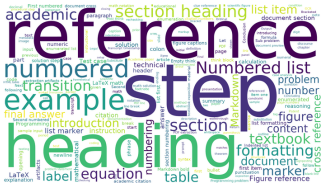
Expert 9



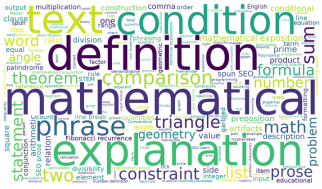
Expert 10



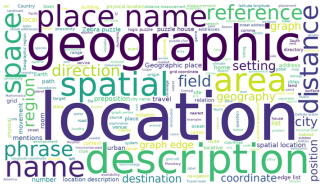
Expert 11



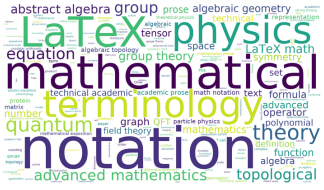
Expert 12



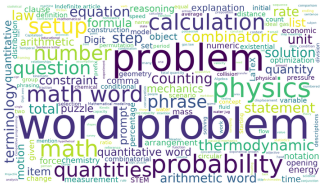
Expert 13



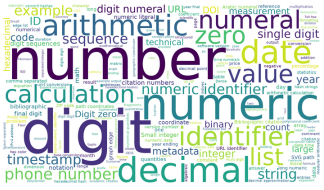
Expert 14



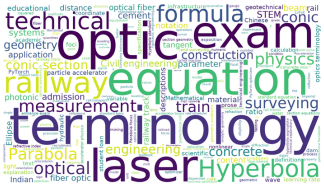
Expert 15



Expert 16



Expert 17

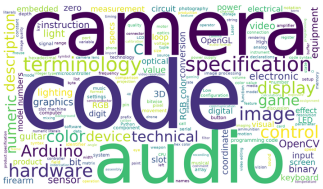


Expert 18

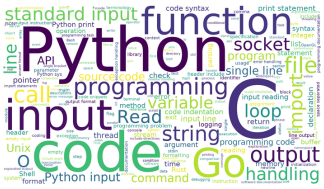


Expert 19

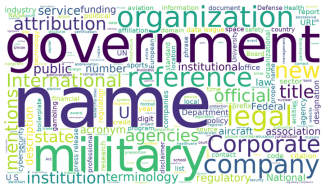
Figure 10: Explanation word clouds of experts 2 to 19 of layer 30, continued from Figure 9.



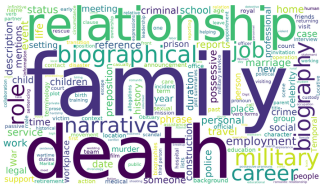
Expert 20



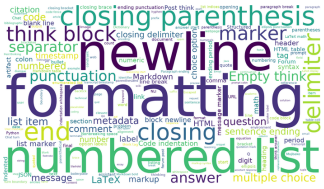
Expert 21



Expert 22



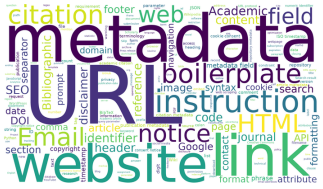
Expert 23



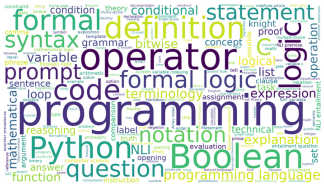
Expert 24



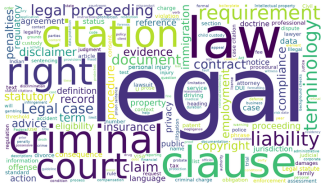
Expert 25



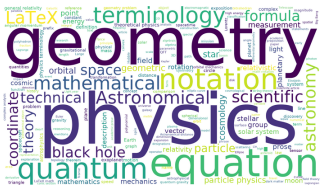
Expert 26



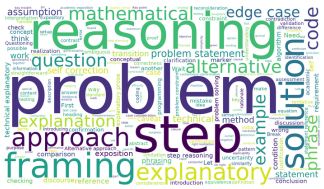
Expert 27



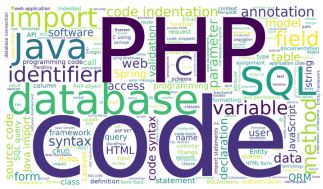
Expert 28



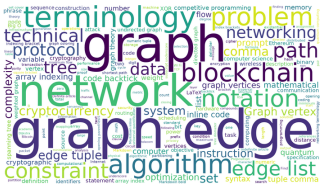
Expert 29



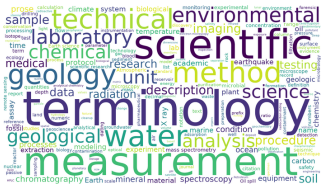
Expert 30



Expert 31



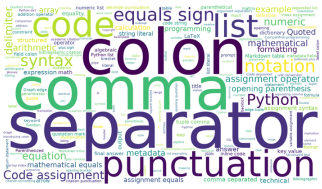
Expert 32



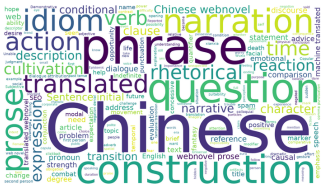
Expert 33



Expert 34



Expert 35



Expert 36



Expert 37

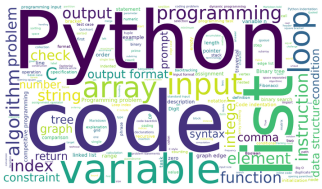
Figure 11: Explanation word clouds of experts 20 to 37 of layer 30, continued from Figure 9.



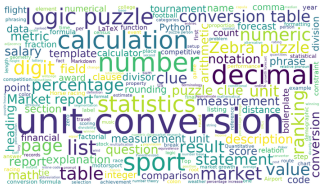
Expert 38



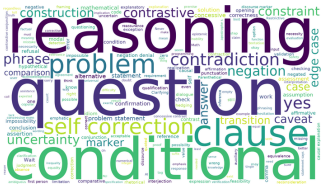
Expert 39



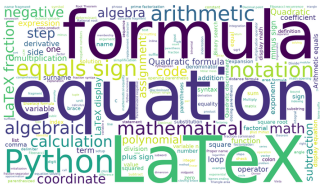
Expert 40



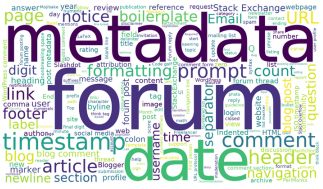
Expert 41



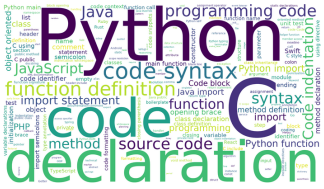
Expert 42



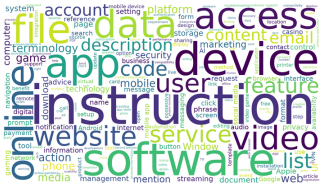
Expert 43



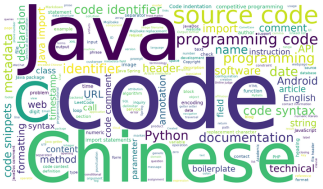
Expert 44



Expert 45



Expert 46



Expert 47



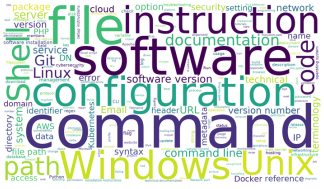
Expert 48



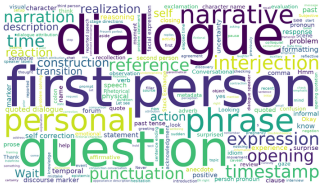
Expert 49



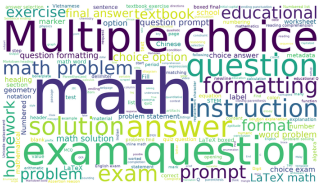
Expert 50



Expert 51



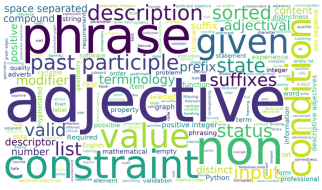
Expert 52



Expert 53

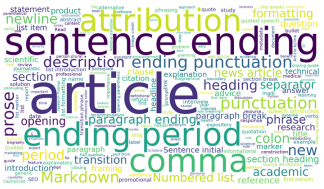


Expert 54

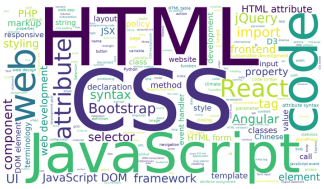


Expert 55

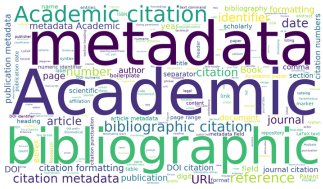
Figure 12: Explanation word clouds of experts 38 to 55 of layer 30, continued from Figure 9.



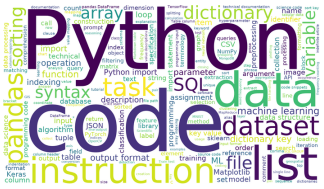
Expert 56



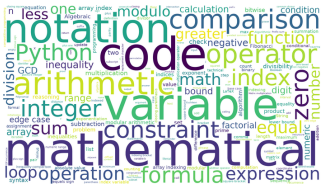
Expert 57



Expert 58



Expert 59



Expert 60



Expert 61



Expert 62



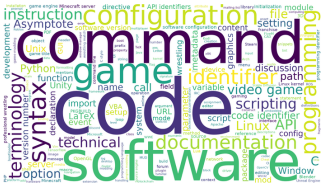
Expert 63



Expert 64



Expert 65



Expert 66



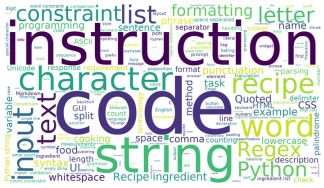
Expert 67



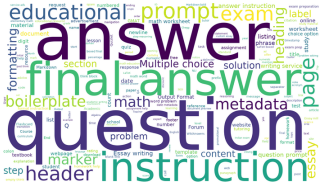
Expert 68



Expert 69



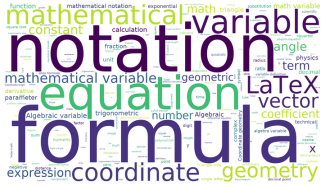
Expert 70



Expert 71

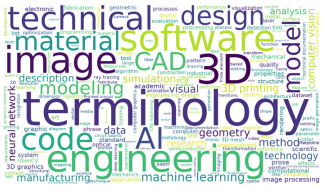


Expert 72



Expert 73

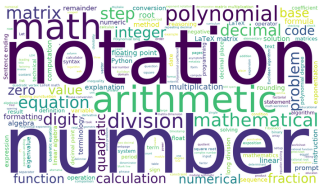
Figure 13: Explanation word clouds of experts 56 to 73 of layer 30, continued from Figure 9.



Expert 74



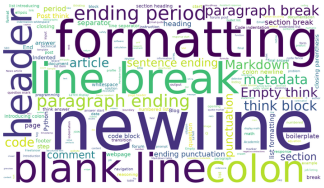
Expert 75



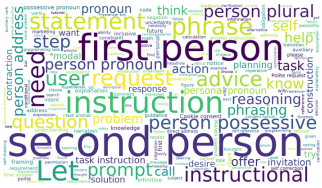
Expert 76



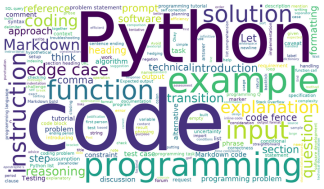
Expert 77



Expert 78



Expert 79



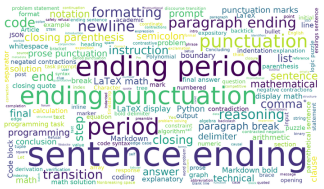
Expert 80



Expert 81



Expert 82



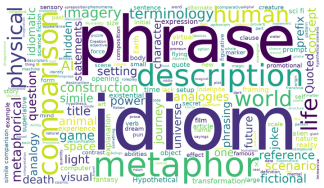
Expert 83



Expert 84



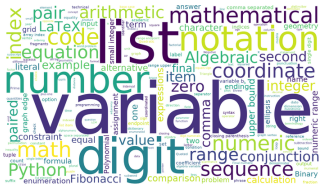
Expert 85



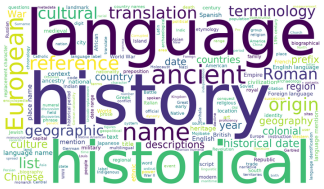
Expert 86



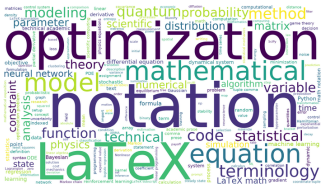
Expert 87



Expert 88



Expert 89



Expert 90

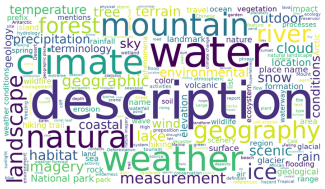


Expert 91

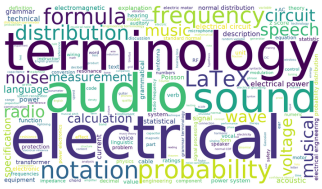
Figure 14: Explanation word clouds of experts 74 to 91 of layer 30, continued from Figure 9.



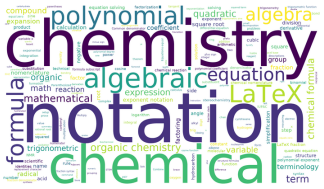
Expert 92



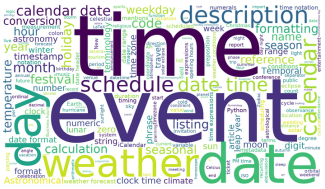
Expert 93



Expert 94



Expert 95



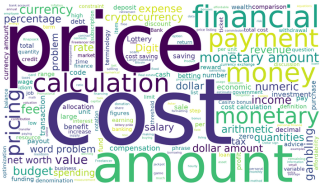
Expert 96



Expert 97



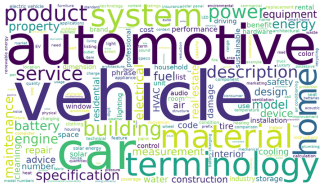
Expert 98



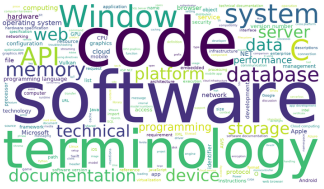
Expert 99



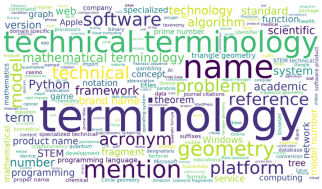
Expert 100



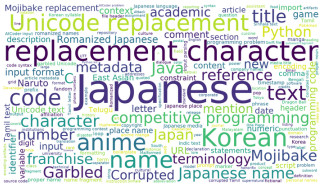
Expert 101



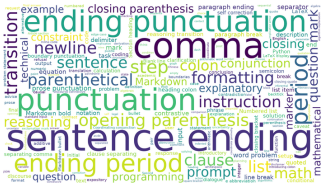
Expert 102



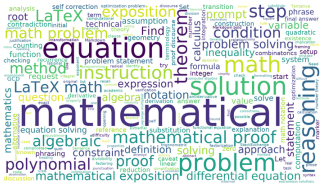
Expert 103



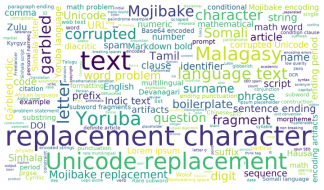
Expert 104



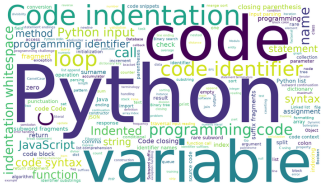
Expert 105



Expert 106



Expert 107

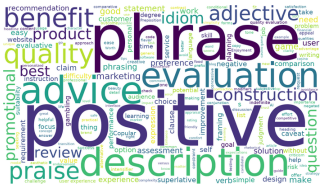


Expert 108

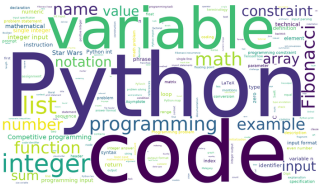


Expert 109

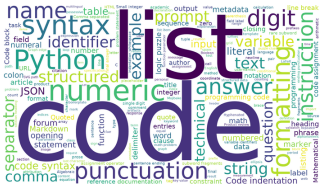
Figure 15: Explanation word clouds of experts 92 to 109 of layer 30, continued from Figure 9.



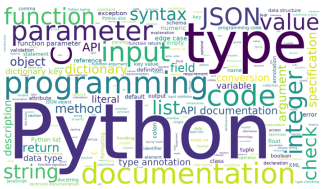
Expert 110



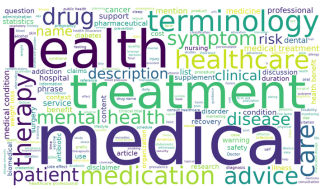
Expert 111



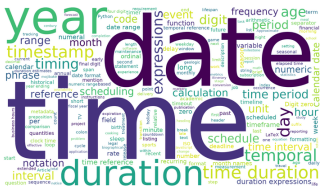
Expert 112



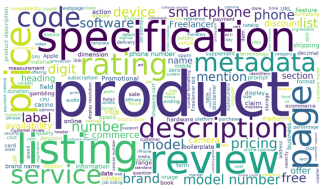
Expert 113



Expert 114



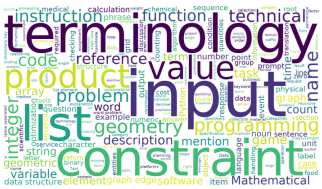
Expert 115



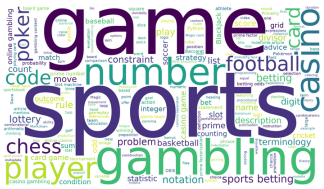
Expert 116



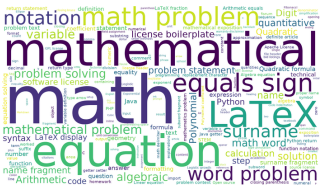
Expert 117



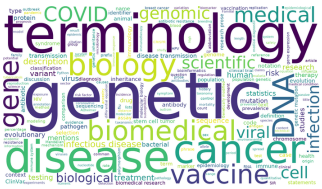
Expert 118



Expert 119



Expert 120



Expert 121



Expert 122



Expert 123



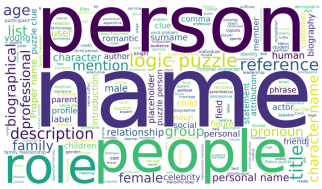
Expert 124



Expert 125



Expert 126



Expert 127

Figure 16: Explanation word clouds of experts 110 to 127 of layer 30, continued from Figure 9.

F PROMPT TO JUDGE WHETHER A LATENT BELONGS TO A THEME

The theme name is substituted for {theme} in the prompt below, and the latent explanation is appended after the final line. The model is asked to answer only yes or no. The same prompt is used for the share row of Table 2 and for the random baseline, which draws 2000 latents from the dominant sets of every expert other than the one under study.

```
You are an expert in mechanistic interpretability, specializing in deciding whether a
     Sparse Autoencoder (SAE) latent of a large language model belongs to a given theme.

## Background

We study expert specialization in Qwen3-30B-A3B. Every MoE layer holds 128 experts. For
     each expert we recover the set of SAE latents whose activation that expert
     principally drives, and we interpret latents sampled from that set into a short
     natural language explanation, a noun phrase such as "Sailing knot names", "Origami
     folding steps" or "Falconry equipment terms".

## Task

Decide whether the latent explanation below belongs to the theme named after this
     paragraph. Belonging means the explanation is about that theme, not merely adjacent
     to it or sharing a generic word with it. For example, under the theme "Beekeeping",
     "Hive frame inspection" and "Queen bee marking" belong, while "Insect wing anatomy"
     and "Pollinator garden plants" do not, even though they are also about insects and
     flowers.

The theme is: {theme}

## Output format

Your response should contain only yes or no, in lowercase, with no other content.

The feature explanation is:
```

G PROMPT TO LABEL AN EXPERT FROM ITS EXPLANATION WORD CLOUD

The word cloud of an expert is passed to the model as one line per entry, of the form word: weight, appended to the prompt below in decreasing order of weight.

```
You are an expert in mechanistic interpretability, specializing in describing what an
     individual expert of a Mixture-of-Experts (MoE) language model is responsible for.

## Background

We study expert specialization in Qwen3-30B-A3B. Every MoE layer holds 128 experts, and a
     router sends each token to a few of them. To find out what an expert is responsible
     for we do not look at the tokens routed to it, because a token is selected by the
     router but the information the expert adds lives in the features of the layer output.

A Sparse Autoencoder (SAE) with 2^20 latents is trained to reconstruct the output of one
     MoE layer, so each of its latents is one interpretable feature of what that layer
     computes. The activated part of a latent is an affine function of the layer output,
     so the activation of a latent decomposes exactly into a sum of per-expert
     contributions. Averaging the normalized contribution over a large corpus gives, for
     every expert and latent pair, the fraction of that latent's activation the expert is
     responsible for. When this fraction is large we say the expert dominates the latent,
     and the set of all latents an expert dominates is its dominant set. Dominant sets
     are nearly disjoint across experts, so the dominant set of an expert is a
     description of that expert alone.

Latents sampled from each dominant set were separately interpreted by a language model
     from activating contexts and logit effects. The result is a short natural language
     explanation, a noun phrase such as "Sailing knot names", "Origami folding steps" or
     "Falconry equipment terms".

## Input

You are given the word cloud of one expert, built from the explanations of latents
     sampled from its dominant set. Each entry is a word, or a frequent two word
```

```
     collocation, together with a weight. The weight is the frequency of that word among
     the explanations divided by the frequency of the most common word, so the heaviest
     entry always has weight 1.000. Common English stopwords have been removed, plural
     forms have been merged into their singulars, and at most 200 entries are listed in
     decreasing weight.

This word cloud is the only evidence you have about the expert. You are seeing the
     vocabulary of the explanations rather than the explanations themselves, so it tells
     you which concepts recur and how often, but not how they combined inside any single
     latent.

## Task

Name what this expert is responsible for, using at most nine words.

- Weigh words by their weights and spend your attention on the heaviest ones. A long tail
     of light words is normal and does not have to be covered.
- Look for what sets this expert apart from an arbitrary other expert. Words such as
     code, text, term, terminology, description, name, reference and value are frequent
     in many experts and are weak evidence on their own. Prefer the words that would be
     surprising in a random expert.
- Say what the expert is about instead of concatenating its heaviest words. For an expert
     whose cloud is led by hive, honey, beeswax and queen, "hive, honey, beeswax, queen"
     is a bad answer and "beekeeping" is a good one.
- You are not choosing from a fixed list of topics. An expert may specialize along an
     axis that is not a subject area at all. Name that axis when that is what the cloud
     shows.
- A few experts carry several unrelated themes at once. When the heavy words clearly
     split into separate groups, name each group and join them with "and", for instance
     "beekeeping and origami". The nine word limit binds here, so keep every theme to one
     or two words, and when the groups do not all fit, keep the heaviest ones and drop
     the rest. Do not force one label onto such an expert, and do not invent a second
     theme when the cloud only supports one.
- Be as specific as the evidence allows and no more specific than that. If the cloud is
     about geometry, say geometry rather than mathematics. If it spreads over several
     unrelated parts of mathematics, say mathematics.

## Output format

Your response should contain only the label with no other content. Write a noun phrase of
     at most nine words, with no trailing punctuation and no quotation marks. Keep the
     capitalization of proper nouns such as Kyoto, UNESCO and Darwin, and leave every
     other word in lowercase.

The word cloud of this expert is:
```

H PROMPT TO GROUP THE EXPERTS INTO DOMAINS

The 128 labels are appended to the prompt below as one line per expert, of the form `E{expert id}: {label}`. The only quantities we set are the bounds on the number of groups, between 6 and 13, and the returned number of groups, eleven, falls strictly inside them. The prompt fixes the level of grouping but names no domain that the partition should contain.

```
You are an expert in mechanistic interpretability, working on the division of labor among
     the experts of a Mixture-of-Experts (MoE) language model.

## Background

We study expert specialization in Qwen3-30B-A3B. One MoE layer holds 128 experts, and a
     router sends each token to a few of them. For every expert we recovered the set of
     Sparse Autoencoder latents whose activation that expert is chiefly responsible for,
     we interpreted latents sampled from that set into short natural language
     explanations, and we summarized those explanations into one short label naming what
     the expert is responsible for. Those 128 labels are the input below, one per expert.

The labels were written independently of one another, so their wording is not
     standardized. Two experts can hold closely related responsibilities while sharing no
     word, and two labels can share a generic word such as terminology, description or
     reference while describing responsibilities that have nothing to do with each other.
     Read for what an expert does rather than for surface overlap.

This grouping asks a coarse question. Group at the coarsest level the labels still
     support, and leave the internal structure of a group alone.
```

```
## Task

Partition the 128 experts into between 6 and 13 groups, aiming for about 10.

- Every expert belongs to exactly one group and all 128 experts are placed. Nothing may
    be dropped and nothing may appear twice.
- A group is good when its name is true of every expert in it. Among the names that stay
    true, take the most general one rather than the most specific, and prefer one large
    group under a general name over several small groups under specific ones.
- There is no upper bound on the size of a group, and group sizes do not need to be
    balanced. Do not split a group into subfields merely to even out the sizes.
- Do not cut a small coherent domain apart either, and do not manufacture groups merely
    to reach the group count.
- A group may be organized around any responsibility its members share.
- Many labels name two or three themes at once. Place such an expert by the theme that
    carries most of its label, preferring a theme that already has a home among the
    other experts. Do not open a group for a single odd pairing.
- Keep one axis in one place. Two experts responsible for the same kind of thing belong
    in the same group even when their labels attach that thing to different subject
    matter, so do not leave near duplicates of a single theme sitting in two different
    groups.
- Avoid a catch-all group. A group is a catch-all whenever some of its members lack the
    property its name states, and giving that group a specific sounding name does not
    repair it. Check every group by walking its members one at a time and asking whether
    the name is true of that member, then move out the ones for which it is not. If a
    few experts genuinely fit nowhere they may share one residual group, which must stay
    small and must be named for something its members actually have in common.
- Name each group as generally and as briefly as you can, one or two words wherever the
    group allows it. Two closely related things may be joined with and. What is not
    acceptable is a name that enumerates three or more subject areas, of the shape "A,
    B, and C", because such a name states no organizing principle at all. When the only
    name you can find for a candidate group is a list of that shape, either find the
    more general word that covers the list, or split the group along the very boundaries
    the list names.

## Output format

Respond with JSON and nothing else. No prose before or after it and no markdown code
    fence. The JSON is a list of objects ordered from the largest group to the smallest,
    each object having exactly two fields:

- "group", the name of the group
- "experts", the ids of the experts in that group, as integers in ascending order

The shape of the response is

[{"group": "<name of the largest group>", "experts": [0, 7, 19]}, {"group": "<name of the
    next group>", "experts": [3, 12]}]

The 128 expert labels are:
```

I EXPERT LABELS AND DOMAINS

Table 3 gives the label of every expert of layer 30 together with the domain it was grouped into, the experts of one domain being listed together. The domains appear in decreasing order of size, and the experts within a domain in increasing order of identity.

Table 3: The 128 experts of layer 30, their labels and the domains they were grouped into. A label is written by GPT-5.5 from the explanation word cloud of that expert alone, and the domains are recovered from the labels in a single further call. Both prompts are given in Section G and Section H.

Domain	Expert	Label
Computing	9	competitive programming statements and formal instruction phrasing
	19	conditional programming logic and edge case handling
	20	embedded multimedia hardware programming and specifications
	21	C and Python programming input/output
	31	database-backed web application code
	32	graph algorithms and blockchain protocols
	40	competitive programming and algorithmic code

Domain	Expert	Label
	45	source code syntax and declarations
	47	Chinese Java source code and technical documentation
	50	Java and Android programming
	51	command-line software configuration and system administration
	57	HTML CSS and JavaScript frontend development
	59	Python data processing and machine learning code
	60	integer arithmetic and algorithmic code notation
	66	software configuration command syntax and game modding
	68	software development packages and documentation
	69	JavaScript web and Swift iOS programming
	70	programming string manipulation and recipes
	72	coding algorithms and musculoskeletal fitness
	81	code indentation and block syntax formatting
	100	cybersecurity and safety risk management
	102	software engineering and computing platforms
	108	Python code syntax and indentation
	111	Python competitive programming and Star Wars
	112	code syntax and structured list formatting
	113	Python programming types and API documentation
	117	government safety and event handler programming
	118	competitive programming problem specifications
Text form	8	Markdown answer formatting and Python/math problem solving
	12	academic document structure and formatting
	24	structured text formatting and delimiters
	26	webpage metadata and academic citation boilerplate
	34	definitions and reference page formatting
	35	punctuation and separators in code and notation
	44	web forum metadata and boilerplate
	56	article prose punctuation and formatting
	58	academic citation metadata and bibliography formatting
	71	educational Q&A and exam page formatting
	77	non-English proper names and Unicode mojibake
	78	newlines and structured text formatting
	80	Python programming solutions and Markdown formatting
	83	sentence endings and punctuation formatting
	98	web and source code boilerplate notices
	104	Japanese text, Unicode mojibake, and competitive programming
	105	punctuation and text formatting
	107	Unicode replacement characters and mojibake
	122	numeric tables and OCR formatting artifacts
	124	Chinese web boilerplate and contact metadata
	126	comma separated lists and numeric sequences
Mathematics and reasoning	1	coordinate and Euclidean geometry
	2	formal mathematical notation and problem setup
	11	LaTeX math notation and algebraic expressions
	13	mathematical definitions and corrupted SEO prose
	15	LaTeX notation for advanced mathematics and physics
	16	quantitative STEM word problems
	27	Boolean operators in programming and formal logic
	29	mathematical physics, geometry, and astronomy
	30	problem-solving reasoning and solution exposition
	41	unit conversions, sports statistics, and logic puzzles
	42	conditional reasoning and self-correction
	43	mathematical equations, LaTeX, and Python syntax
	53	multiple-choice math exam questions and solutions
	67	statistical analysis and research methods
	73	mathematical formulas and notation
	76	mathematical notation, arithmetic, and algebra
	88	mathematical and programming variables, indices, and lists
	90	mathematical optimization and modeling notation
	106	mathematical problem solving and proofs
	120	mathematical equations and problem solving
Applied science	0	industrial energy and infrastructure sectors
	6	chemistry terminology and notation

Domain	Expert	Label
	18	optics, conic sections, and civil engineering
	33	scientific measurement and geoscience laboratory methods
	49	biological and biomedical terminology
	61	medical symptoms and mind-body wellness
	62	mechanical engineering specifications and botanical morphology
	74	3D CAD engineering and computer vision
	82	biomedical research terminology and identifiers
	84	food chemistry and nutrition
	92	medical anatomy and physiology terminology
	94	electrical/audio engineering, probability notation, and grammar terminology
	95	chemical and algebraic notation
	101	automotive and residential building systems
	103	STEM and software technical nomenclature
	114	medical treatment and healthcare advice
	121	biomedical genetics and disease terminology
	123	scientific causal reasoning about quantitative change
Language	36	Chinese webnovel translationese and discourse constructions
	38	polite greetings, thanks, and sign-offs
	52	first-person dialogue and narrative discourse markers
	55	adjectival morphology and technical constraints
	75	definitions and NLI tasks
	79	first and second person instructional dialogue
	85	temporal and comparative framing of growth and change
	86	idioms and figurative comparisons
Culture and recreation	3	Christian theology and role-playing game mechanics
	54	animals and classical arts
	63	entertainment media metadata and reviews
	65	biblical references, fiction franchises and SVG path coordinates
	89	history, languages and cultural geography
	109	physical movement in games, vehicles, and combat
	119	sports, games, gambling, and number theory
Entities and identifiers	5	programming identifier and variable names
	17	numeric strings and identifiers
	22	official government, military, and corporate entity names
	23	biographical life events and relationships
	87	proper names titles identifiers and their fragments
	91	uppercase code identifiers constants and comments
	127	personal names, roles, and logic puzzle characters
Safety and advice	4	food recipes and ingredient descriptions
	7	everyday household, cooking, and safety advice
	10	refusal constructions and safety request framing
	46	digital device and software usage instructions
	110	positive evaluative phrasing in advice and marketing
Commerce	37	financial accounting, tax, and investment calculations
	39	financial market news and weather forecasts
	64	business software marketing and project management
	99	monetary amounts and pricing calculations
	116	e-commerce product listings and specifications
Society and law	25	academic education and job requirements
	28	legal documents and proceedings
	48	education and management theory terminology
	97	political economy and social policy discourse
	125	legal documents, citations, and American history
Time and place	14	spatial and geographic locations
	93	physical geography and weather
	96	calendar events, date-time expressions, and weather
	115	date, time, and duration expressions

J SAE TRAINING DETAILS

We trained our SAE with the latent size of $2^{20} = 1,048,576$ on the layer 4,30,46 MoE layer output stream of Qwen3-30B-A3B (Yang et al., 2025) independently. Activations were collected from a mixture of pre-

Table 4: The FVU of our SAE on each layer.

Layer	FVU
4	12.6%
30	25.1%
46	7.6%

training and post-training data taken from the Olmo 3 (Olmo et al., 2025) training corpus: $5/8$ from pre-training data, $1/4$ from a reasoning chain-of-thought dataset, $1/8$ from an instruction-following supervised fine-tuning dataset.

In order to increase the diversity of the activation sources within a single batch, we dumped all the activations before training. For each example in the dataset, we ran a forward pass of the LLM to obtain the MoE activations and then sampled only 512 of them from that example into the pool, discarding the remaining activations. For each training-batch, we drew a batch from activations we dumped uniformly at random without replacement.

Our hyperparameters are set following the common practice. We used a square-root warmup followed by a linear decay. During the first 100 steps, the learning rate is warmed up from $0.1\eta_{\max}$ to the peak value $\eta_{\max}$ following a square-root schedule, i.e., $\eta(t) = \eta_0 + (\eta_{\max} - \eta_0)\sqrt{t/100}$ with $\eta_0 = 0.1\eta_{\max}$. The learning rate then linearly decays from $\eta_{\max}$ to η_0 during the remainder of training. We set $\eta_{\max} = 8 \times 10^{-5}$.

We trained the SAE with the AdamW optimizer (Loshchilov & Hutter, 2017) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and a weight decay of 0.01. We used a batch size of 8192 and trained each SAE on about 3.2B tokens.

Our initialization setup generally follows Conerly et al. (2025) and Ameisen et al. (2025). Let $n = 2048$ be the hidden size of LLM, and $m = 1,048,576$ be the latent size of SAE. We normalize the activations to $\sqrt{n}$, i.e., $\mathbb{E}\left[\|X\|\right] = \sqrt{n}$, where X denotes the activations of the given layer.

After normalization, we filter out activations whose norm exceeds 20 times the batch-wise mean, as such outliers would destabilize training.

We initialized every entry of the encoder matrix from $\mathcal{U}(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}})$ and every entry of W_{dec} from $\mathcal{U}(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$.

Our SAE is trained with activation function TopK. For the number of activations κ retained by the activation function, we set $\kappa = 1000$ initially, and let κ linearly decay to 100 during the first $10{,}000$ steps, after which it is held constant. We used an auxiliary loss following Gao et al. (2025) and Deng et al. (2026) with the coefficient of $1/32$. The auxiliary loss always uses the same value of κ as our main activation function. After training we normalize each column of W_{dec} to unit norm and rescale the corresponding row of W_{enc} by the reciprocal factor, which yields an equivalent SAE. We likewise fold the activation normalization factor into the weights, so the exported SAE applies directly to the raw layer output as in Equations (2) and (3).

We use fraction of variance unexplained (FVU) to evaluate our SAE. The FVU values of each layer are shown in Table 4. Consistent with Gao et al. (2025), we also observe that the intermediate layers of LLMs are more difficult to reconstruct using SAE, compared with both shallow and deep layers.

K Pipeline of auto-interpretability

Following common practice, we use an LLM to produce a natural-language explanation for each latent. Unlike prior work that relies primarily on activated contexts, we provide the explainer with three sources of evidence: (i) activated contexts, (ii) logit effects, and (iii) layer information.

Activated contexts. We run inference on the 1.5B-token validation set of Section 4, which does not overlap with the SAE training data. For each latent, we collect all activated contexts within a 64-token window and select the top 5 by activation strength. We additionally sample 5 contexts uniformly at random from the remaining activations. We found that top-activating contexts are often highly similar to one another, which biases the explainer toward the literal surface pattern shared by those examples. Adding randomly sampled contexts increases diversity and encourages the explainer to summarize the latent's more general function rather than its most salient literal trigger. All ten contexts, along with their token-level activation values, are provided to the explainer.

Logit effects. To quantify a latent's direct effect on the output distribution, we apply the logit lens, projecting the latent's decoder vector through the LM head and extracting the top-10 boosted and top-10 suppressed tokens

by logit value. Boosted tokens indicate what the latent promotes, while suppressed tokens indicate what it suppresses.

Layer information. We also tell the explainer the total number of layers, the layer at which the latent lives, and its relative depth, together with a soft prior on how features are typically distributed across depth: shallow layers tend to contain *input features* that respond to specific tokens, local syntax, or surface patterns; middle layers tend to contain *abstract features* that capture discourse patterns, reasoning states, or semantic categories; and deep layers tend to contain *output features* that directly shape next-token prediction. This prior guides how the explainer weighs the evidence but is explicitly framed as a tiebreaker rather than a constraint. For input features, activated contexts are usually sufficient; for output features, the logit effects are more informative, since their activated contexts are often diffuse; for abstract features in the middle layers, neither source alone is typically conclusive, and the explainer is instructed to combine both. For output features, we adopt a fixed naming convention (`Say X` for boosted tokens and `Don't Say X` for suppressed tokens) so that output-oriented explanations are easily distinguishable from concept-level ones.

Output format. In addition to a concise explanation phrase, we ask the explainer to report a confidence rating on a three-level scale (`High`, `Medium`, `Low`) reflecting how well the provided evidence supports its explanation. The explainer is also allowed to output `Uninterpretable` when the activated contexts and logit effects do not exhibit a consistent, describable pattern, rather than being forced to produce a speculative explanation.

The full system prompt and user prompt are given in Section K.1.

K.1 PROMPTS

We provide the full system prompt and user prompt template below. In the user prompt, fields in double braces are filled in per latent.

```
You are an expert in mechanistic interpretability, specializing in interpreting features
     extracted by Sparse Autoencoders (SAEs) from a large language model.

## Task

Given one SAE feature, infer the concept, behavior, pattern, or output tendency it
     represents. Produce:
1. A concise interpretation phrase (2-6 words preferred; clarity > brevity).
2. A confidence rating.

---

## Input

You may receive up to three sections. Any may be missing.

### 1. Context Activations
Text examples that strongly activate the feature, with token-level activation highlights.

[Context 1]
[Text]
"The cat sat on the mat in Paris"
[/Text]
Top Activations: [mat|0.92] [Paris|0.87]
[/Context 1]

How to read them:
- Global pass: identify topic, domain, task, language, register, formatting, or discourse
     situation shared across examples.
- Local pass: examine the highest-activating tokens -- do they share a semantic field,
     syntactic role, position, or discourse function?
- Cross-example pass: isolate the single common thread that explains every activation.
     Reject hypotheses contradicted by any example.

### 2. Logit Effect
The feature's output influence at its own layer, presented as a direct logit-lens
     projection of the decoder vector: a list of boosted / suppressed tokens.

Use it to infer what the feature pushes the model to say or avoid saying. Logit effects
     can be noisy (especially in earlier layers), so weight context activations more
     heavily when they are clearer.

### 3. Layer Information
Soft prior based on layer depth:
```

```
- Shallow -> surface features: lexical items, local syntax, formatting, named entities,
    languages, domain terminology.
- Middle -> abstract / behavioral features: discourse patterns, reasoning states, task
    structure, style, intent, semantic categories.
- Deep -> output-oriented features: answer formats, completion templates, specific
    phrases, refusal / compliance, "say X" behaviors.

The layer prior is a tiebreaker, not a constraint. Override it whenever activations or
    logit effects give stronger evidence.

---

## Reasoning Procedure

1. Activations -- what topic / pattern, which tokens, what common thread?
2. Logit effect -- what is boosted, what is suppressed, is there a clear output push?
3. Layer prior -- what type of feature does the depth suggest?
4. Synthesis:
   - Activations + logit effect agree -> combine into one label.
   - They diverge -> take the clearer signal; do not force a unified story.
   - Both incoherent -> output `Uninterpretable`.
5. Phrase -- specific, 2-6 words, no filler.
6. Confidence -- see scale below.

---

## Output Format

Output only the two tags below. No reasoning, preamble, or commentary.

<explanation>concise interpretation phrase</explanation>
<confidence>High | Medium | Low</confidence>

Confidence scale:
- High -- activations and logit effect are coherent, repeated, and specific.
- Medium -- plausible interpretation; some evidence is partial, noisy, or only one
    channel is clear.
- Low -- weak, sparse, or contradictory evidence; label is tentative.

If evidence is too noisy, contradictory, or insufficient, output exactly:

<explanation>Uninterpretable</explanation>
<confidence>Low</confidence>

---

## Label Conventions

### Surface features (shallow)
Name the concept or pattern directly.
- `European capitals`
- `medical terminology`
- `Python indentation`
- `Chinese punctuation`
- `HTML attribute syntax`

### Behavioral features (middle)
Name the behavior, capability, discourse function, or abstract state.
- `formal register`
- `negation logic`
- `Chinese language response`
- `step-by-step reasoning`
- `instruction-following mode`
- `uncertainty expression`

### Output-oriented features (deep)
- Boosted output -> prefix with `Say`: `Say however`, `Say comma`, `Say closing
    parenthesis`, `Say final answer`.
- Suppressed output -> prefix with `Don't Say`: `Don't Say period`, `Don't Say apology`,
    `Don't Say the`.

Do not substitute `Produce`, `Express`, `Output`, `Generate`, etc. for `Say`.

---

## Specificity Rules

Be precise. Replace vague labels with concrete ones:
```

```
| Avoid                | Prefer                          |
|----------------------|---------------------------------|
| `locations`          | `European capitals`             |
| `science-related`    | `scientific terminology`        |
| `punctuation tokens` | `sentence-ending punctuation`   |
| `negative words`     | `negation / denial`             |
| `medical stuff`      | `medical symptom description`   |

Banned patterns: `xxx-related`, `xxx tokens`, `xxx words`, `xxx stuff`, `xxx things`, and
     meta-descriptions of the feature itself (e.g. `activates on nouns`, `high activation
     feature`).

---

## Honesty Rule

Do not fabricate coherence. When evidence does not support a specific interpretation,
     output the `Uninterpretable` template above. Forced labels are worse than honest
     abstention.
```

```
## Context Activations
Two complementary sets of examples are provided: the strongest (top) activations and a
     random sample. Read both -- the random set guards against over-fitting to the most
     extreme cases.

### Top Activating Contexts
The 5 examples where this feature fires most strongly.

[Top Context 1]
[Text]
{{top_context_1.text}}
[/Text]
Top Activations: {{top_context_1.top_activations}}
[/Top Context 1]

... (Top Context 2-5 in the same format)

### Random Contexts
5 examples sampled at random from this feature's activating set (a less-biased view of
     where it fires).

[Random Context 1]
[Text]
{{random_context_1.text}}
[/Text]
Top Activations: {{random_context_1.top_activations}}
[/Random Context 1]

... (Random Context 2-5 in the same format)

## Logit Effect
Source: decoder-vector projection. This feature lives at layer {{layer_idx}} (single
     decoder; SAE-style).
Decoder L2 norm = {{decoder_l2_norm}}

Top 10 boosted logits:
Rank 1: [{{token}}], logit={{logit}}
... (Rank 2-10)

Top 10 suppressed logits:
Rank 1: [{{token}}], logit={{logit}}
... (Rank 2-10)

## Model Information
Model: {{model_name}} -- {{num_layers}} transformer layers (0-indexed).
Feature extractor: Sparse Autoencoder (SAE), trained at layer {{layer_idx}}. Each feature
     has a single decoder vector at its own layer.

## Layer Information
The model has {{num_layers}} layers in total (0-indexed); the current feature's layer
     index is {{layer_idx}} (relative depth {{layer_idx / num_layers}}). Use the relative
     position as a soft prior, but override it with the logit-effect signal whenever the
     two disagree.

Analyze the context activations, logit effect, and model / layer information, then
     respond in the exact output format specified in the system prompt above.
```