Neuron Populations Exhibit Divergent Selectivity with Scale

Amil Dravid[1] **Yasaman Bahri**[1] **Alexei A. Efros**[1] **Yossi Gandelsman**[2]

[1]UC Berkeley [2]Reve

Abstract

We investigate whether neuron populations within neural networks evolve predictably with scale, extending scaling laws beyond macroscopic observables such as loss. To probe this question, we study *Rosetta Neurons*, a previously characterized class of neurons whose activation patterns are similar across independently trained models (Dravid et al., 2023). In separate analyses of language models up to 30B parameters and vision models up to 5B parameters, we observe that the population of Rosetta Neurons follows a sublinear power law in model size, growing in absolute number but occupying a shrinking fraction of the total neuron count. We further observe a *Neuron Polarization Effect*: Rosetta Neurons become more selective with scale, consistent with increasing monosemanticity, separating from a growing non-Rosetta population that remains less selective. An analytical model balancing feature utility against limited neuron capacity explains the sublinear power-law scaling and this polarization effect. Finally, we find that Rosetta Neurons become more domain-specialized with scale and illustrate their selectivity through a targeted data-filtering case study for continued pretraining. Our results point to a scaling law for interpretable, shared neuron-level structure, linking model size to systematic changes in neuron universality, selectivity, and specialization.[1]

1 Introduction

A central question in both deep learning and neuroscience is how neurons encode structure in the world. In biological systems, this question has motivated longstanding debates about whether representations are localized in single units (Hubel & Wiesel, 1962) or distributed across populations (Haxby et al., 2001), as well as studies of how such structure might recur across subjects (Hasson et al., 2004). An analogous question arises for artificial neural networks. Despite a growing body of work probing neurons, such as those encoding sentiment in language models (Radford et al., 2017) or object segments in vision models (Bau et al., 2017), most neurons in today's large-scale models are not easily interpretable (Bills et al., 2023).

One reason for this difficulty is that neuron-level representations may vary in how cleanly they isolate individual features. Some neurons appear relatively monosemantic, responding selectively to coherent semantic concepts, while *superposition* predicts that others may be polysemantic, encoding multiple unrelated features when models represent more features than available dimensions (Elhage et al., 2022). This makes monosemanticity and polysemanticity natural questions at the neural-population level: as models scale, which neurons become selective, which remain in superposition, and how do these properties relate to shared internal function across independently trained models?

These questions become especially salient in the scaling regime, where models change not only in performance but also in representational capacity. Neural scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Hestness et al., 2017) have revealed striking regularities in external model behavior,

[1]Project Page: https://avdravid.github.io/rosetta-neuron-scaling
YB, AE, and YG jointly advised this work.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

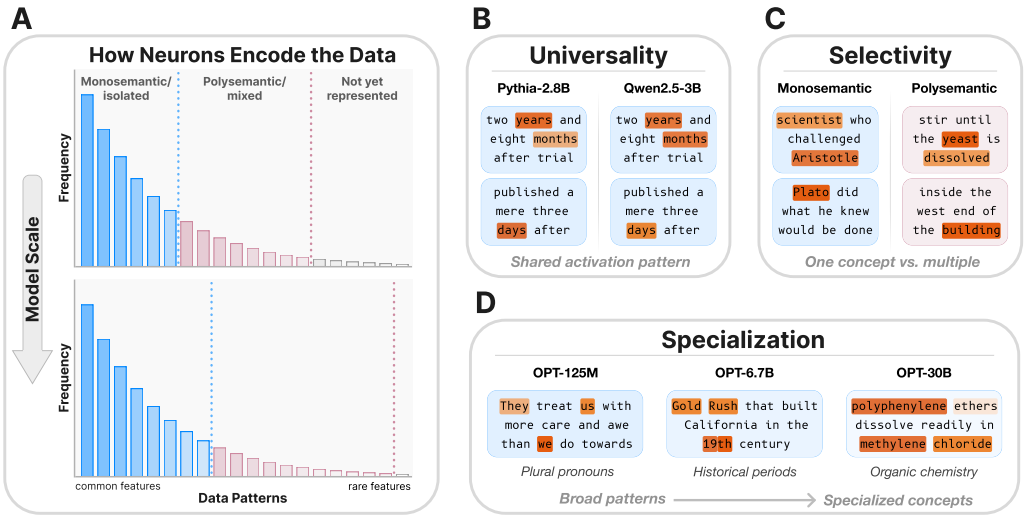


Figure 1: **Neuron populations across scale.** To study how neuron populations scale, we use Rosetta Neurons: units that recur across different models. (A) Features compete for representation in a finite set of neurons, leaving them isolated, mixed, or unrepresented at a given scale. This picture guides our analysis of universality, selectivity, and specialization. In panels B–D, each column shows top-activating contexts from a single neuron. (B) Universality: how does the recurring Rosetta Neuron population scale? (C) Selectivity: do recurring neurons become increasingly monosemantic relative to polysemantic neurons? (D) Specialization: which features of the data distribution are encoded by Rosetta Neurons at different scales? Across language and vision models, Rosetta Neurons grow in number but shrink as a fraction of all neurons, while becoming more selective and specialized.

with quantities such as loss following precise power-law relationships with compute, data, and model size. Yet these laws say little about how representations are organized inside the model. We therefore apply a similar scaling analysis at the neuron level, asking whether cross-model recurring neuron populations evolve predictably with scale: how their size changes, whether they become more selective and monosemantic, and how their features specialize with respect to the data distribution.

To make these questions measurable, we adopt the notion of *Rosetta Neurons* from Dravid et al. (2023): neurons whose activation patterns recur across independently trained models. We use this recurring population to study three neuron-level properties across scale: universality, selectivity, and specialization (Figure 1). Prior work suggests that cross-model recurrence can reflect stable features of the data distribution rather than idiosyncrasies of any one model, but leaves open whether such recurring neurons follow systematic laws across scale and model families in both language and vision.

In this paper, we treat Rosetta Neurons as a scaling observable, identifying recurring cross-model units in language and vision models spanning 80M to 30B parameters across standard pretraining objectives (Section 3). In Section 4, we apply a scaling-law analysis to this population, showing that its size grows predictably with model size as a sublinear power law in neuron count: larger models contain more shared units, yet these units occupy a shrinking fraction of the total neuron population.

We then develop a phenomenological model linking this sublinear scaling to limited neuron capacity (Section 4.3). As networks scale, more features become monosemantically represented in individual neurons and thus more likely to recur across models. However, this set grows more slowly than the broader pool of new features represented in network-specific patterns of superposition (Figure 1A). This theory predicts a *Neuron Polarization Effect*: Rosetta Neurons become increasingly selective with scale, separating from the polysemantic non-Rosetta population. We validate this prediction across language and vision models (Section 5) and further show that Rosetta Neurons become increasingly specialized, shifting toward domains such as code and mathematics.

Finally, we demonstrate this specialization functionally in Section 5.3: Rosetta Neuron activations can filter data from a code domain with near-oracle accuracy, yielding continued-pretraining performance that matches training on ground-truth domain data. Together, our analysis provides a scalable way to identify shared, interpretable, and predictable structure within models, revealing a neuron-level population whose size, selectivity, and specialization evolve systematically with scale.

2 Related Work

Neural scaling laws. A large body of work has shown that neural network performance follows predictable scaling behavior with respect to model scale, dataset size, and training compute, often well described by power laws of the form $\mathcal{L}(x) = \mathcal{L}_\infty + Ax^{-\alpha}$ (Hestness et al., 2017; Kaplan et al., 2020). Subsequent work refined these observations by identifying compute-optimal training regimes (Hoffmann et al., 2022) and demonstrating similar scaling behavior across modalities and architectures (Zhai et al., 2022). A number of works have sought to explain these laws from various perspectives, including (but not limited to) the alignment of tasks to models, the structure of the data distribution, the intrinsic dimensionality of the data manifold, the geometry of learned features, and the learning dynamics of rare tasks under capacity constraints (Bordelon et al., 2020; Michaud et al., 2023; Bahri et al., 2024; Liu et al., 2025; Cagnetta et al., 2026; Huang et al., 2026). We extend this perspective to internal representations, using Rosetta Neurons to study whether a form of shared neuron-level structure also scales predictably.

Structured representations and superposition. Understanding how neural networks develop structured representations has been a central question in deep learning. Interpretability work suggests that learned representations can exhibit approximately linear organization (Mikolov et al., 2013; Arora et al., 2018; Park et al., 2024). Mechanistic studies show that limited-capacity models can represent features in superposition, with toy-model analyses framing this as a capacity-allocation tradeoff between ignoring, isolating, and superposing features (Elhage et al., 2022; Scherlis et al., 2022). Motivated by this structured but superposed organization, sparse decomposition approaches seek to recover interpretable features that are more selective and monosemantic (Bricken et al., 2023b). We build on these ideas at the neuron level, showing that scaling polarizes neurons into a more selective, interpretable Rosetta population against a polysemantic background.

Representational similarity and universality. A complementary line of work studies the extent to which learned representations are shared across models and training runs. Various similarity measures have been proposed to quantify alignment between neural representations (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019), while alternative approaches such as model stitching probe functional equivalence (Bansal et al., 2021; Lenc & Vedaldi, 2015). These ideas build on earlier notions of representational similarity from neuroscience (Kriegeskorte et al., 2008a; Edelman, 1998), where shared geometric structure is used to compare representations across systems (Kriegeskorte et al., 2008b; Haxby et al., 2001). Together, these perspectives have supported growing evidence for convergent structure in learned representations (Sorscher et al., 2022; Huh et al., 2024; Li et al., 2016). At the neuron level, Dravid et al. (2023) identified shared "Rosetta Neurons" across diverse vision models, while Gurnee et al. (2024) studied shared neurons in small-scale GPT-2 models trained from different random seeds. Beyond these settings, we study Rosetta Neurons within both language and vision models spanning heterogeneous architectures and datasets, and characterize their scaling laws.

3 Identifying Rosetta Neurons

We describe here how we identify Rosetta Neurons – common neurons with similar responses across models. We outline how to measure pairwise similarity and then filter these similarities into reliable shared neuron pairs. Further details are provided in Appendix B.

3.1 Quantifying Pairwise Neuron Similarity

We compare MLP neurons (i.e., post-nonlinearity, single-channel activations) between two models. Let model A and model B be two Transformers. We run the same dataset of inputs $\mathcal{X} = \{x_i\}_{i=1}^n$ through both models, where each x_i is either an image or a sequence of text. The Transformer processes each input jointly to produce token-wise activations, which are then aligned across models. In vision models, activations are aligned to a common spatial grid, while in language models they are aligned to a shared sequence of text positions. Details of this alignment procedure are given in Appendix B.4.

Consider a neuron $u = (\ell_A, c_A)$ in model A and a neuron $v = (\ell_B, c_B)$ in model B. Let $m_t^u(x_i)$ and $m_t^v(x_i)$ denote their activations at aligned token t for input x_i. We compare neurons u and v by

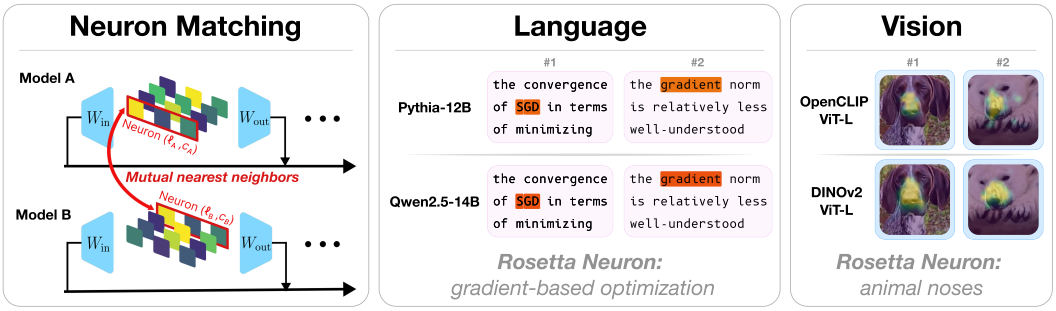


Figure 2: **Identifying Rosetta Neurons.** We compare MLP neuron activations across independently trained models on the same inputs and identify mutual nearest-neighbor pairs under Pearson correlation. The language and vision examples show individual matched neuron pairs firing on the same high-activating inputs, revealing similar activation patterns and coherent shared concepts.

computing the Pearson correlation between these activations on all aligned tokens across the dataset:

$$\text{sim}(u, v) = \frac{1}{N} \sum_{i,t} \frac{m_t^u(x_i) - \mu_u}{\sigma_u} \cdot \frac{m_t^v(x_i) - \mu_v}{\sigma_v}, \tag{1}$$

where μ_u and σ_u^2 are the empirical mean and variance for neuron u over the dataset of N total aligned tokens, and μ_v and σ_v^2 are defined analogously for neuron v. We compute this similarity for all neuron pairs between models A and B.

3.2 Filtering Pairwise Neuron Correspondences

To detect reliable matches between neurons, we retain only those pairs that are nearest neighbors of one another under our similarity metric. Specifically, let $\mathcal{N}(A)$ denote the set of MLP neurons in model A. For neurons $u \in \mathcal{N}(A)$ and $v \in \mathcal{N}(B)$, we include the match (u, v) in the set of matched pairs $\mathcal{R}(A, B)$ iff $v \in \text{NN}_k(u; B)$ and $u \in \text{NN}_k(v; A)$, where $\text{NN}_k(u; B)$ denotes the top-k neurons in model B most similar to u, and analogously for $\text{NN}_k(v; A)$. Unless otherwise stated, all experiments use the default setting $k = 1$; we ablate this choice in Appendix C.2. We interpret the pairs of neurons in $\mathcal{R}(A, B)$ as common units across models, and refer to them as *Rosetta Neurons*. Figure 2 visualizes how these neurons are identified and example matches.

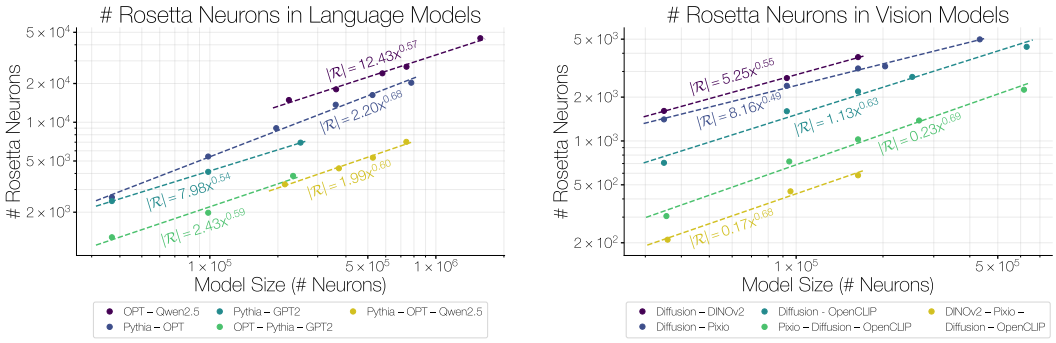
4 Scaling Laws for Rosetta Neurons

In this section, we study how the number of Rosetta Neurons scales with model size in both language and vision models. We first describe the experimental setting, including the data and model families. We then analyze the resulting scaling trends in both domains and compare them against a null baseline. We conclude with a phenomenological model to explain the observed scaling behavior.

4.1 Experimental Setup

Data. In the language setting, each pair of models used for matching is evaluated on a shared set of approximately 10 million tokens formed by sampling sequences i.i.d. from The Pile (Gao et al., 2020). For vision models, we follow Dravid et al. (2023) and match a generative model with a discriminative model. We sample 50,000 class-balanced images from a diffusion model by conditioning on ImageNet-1k labels (Deng et al., 2009), and then pass these images through the discriminative model. Ablations on the data are in Appendix G.

Model Families. In the language domain, we consider the Pythia, GPT-2, OPT, and Qwen-2.5 model families, spanning roughly 100 million to 30 billion parameters (Radford et al., 2019; Biderman et al., 2023; Zhang et al., 2022; Yang et al., 2024). For vision, we use discriminative models from the OpenCLIP, DINOv2, and Pixio families, spanning scales from approximately 80 million to 5 billion parameters (Cherti et al., 2023; Radford et al., 2021; Oquab et al., 2024; Yang et al., 2025).



(a) Rosetta Neuron scaling in language models.

(b) Rosetta Neuron scaling in vision models.

Figure 3: **Scaling laws for Rosetta Neurons in language and vision models.** We plot the number of discovered Rosetta Neurons for various model families at different scales. Dashed lines show power-law fits in log-log space. Across all family comparisons, the fitted exponents are sublinear, and the corresponding fits achieve R^2 values around 0.99. Further details are provided in Appendix C.

For the generative model, we leverage one-step diffusion models built on the Diffusion Transformer architecture (Peebles & Xie, 2023). We provide further details on the model families in Appendix C.

Forming the scaling curves. Given two model families, we form a point on the scaling curve by selecting one model from each family at approximately matched scale. We apply the matching procedure from Section 3 and record the number of discovered Rosetta Neurons. Repeating this across increasingly larger models yields a scaling curve for that family pair. When two or more curves share a common model, we collapse them into a single trajectory by intersecting the Rosetta Neurons identified within the shared model across the comparisons. This produces a stricter notion of universality by keeping neurons shared across more than two model families, allowing us to test whether the same scaling trend holds under a more selective definition of Rosetta Neurons.

Power-law functional form. We model the number of discovered Rosetta Neurons as a power law in model size, $|\mathcal{R}| = cx^{\alpha}$, where $|\mathcal{R}|$ denotes the number of discovered neuron correspondences. Because the matched models are selected to be at approximately the same scale, we use their average total neuron count x to represent the scale of the matched models. In this parameterization, the exponent α is the primary quantity of interest, since it governs how rapidly the number of shared neurons grows with scale, while the constant c depends more strongly on the particular model families and experimental setting. We estimate the fitted curve using ordinary least squares in log-log space.

4.2 Rosetta Neurons Follow Power-Law Scaling

We establish that the Rosetta Neuron population grows as a sublinear power law in both language and vision models. We then compare against two null baselines, untrained networks and input-permuted activations, and find neither exhibits the same scaling trend.

Power-law scaling in language models. Across the language model families we study, the number of Rosetta Neurons increases predictably with model size as shown in Figure 3a. The resulting curves are well fit by power laws, with R^2 values around 0.99. Notably, the fitted exponents lie in a narrow and consistently sublinear range, between approximately $0.5 - 0.7$. This indicates that the number of shared neurons grows predictably with model size, but slower than the rate at which models are scaling. These scaling laws span model sizes from roughly 100 million to 30 billion parameters, corresponding to about 40 thousand to 2 million neurons. The same qualitative behavior holds for the collapsed curves (Pythia-OPT-Qwen2.5, OPT-Pythia-GPT2), suggesting that the scaling trends remain even under a stricter notion of universality.

Power-law scaling in vision models. We observe that a similar scaling trend for Rosetta Neurons emerges in vision models trained with distinct objectives. The scaling curves are well described by power laws, as shown in Figure 3b, with R^2 values around 0.99. The fitted exponents fall in a sublinear range between approximately $0.5 - 0.7$, indicating a stable Rosetta Neuron scaling regime across model families. These scaling laws span model sizes from roughly 80 million to 5 billion parameters, or 40 thousand to 600 thousand neurons. Importantly, this trend holds across models trained with contrastive image-text supervision, self-distillation, masked autoencoding, and diffusion

or flow-based generative modeling. Collapsing curves across shared models reveals a similar scaling pattern, suggesting that the observed scaling law for Rosetta Neurons is robust across multiple models.

Power-law scaling is absent in untrained networks. To test whether matching alone could induce the observed scaling, we apply the same pipeline to untrained networks initialized with their architecture-specific random schemes and report results over three seeds in Figure 4. In both language and vision, this yields few Rosetta Neuron matches and no systematic trend with model size. Thus, the increasing Rosetta count cannot be explained simply by a larger selection pool or by the matching procedure itself. A complementary trained-network null in Appendix C.3 shows that the trend is also absent when input alignment is corrupted. Section C.4 further disentangles model size from the number of training tokens seen.

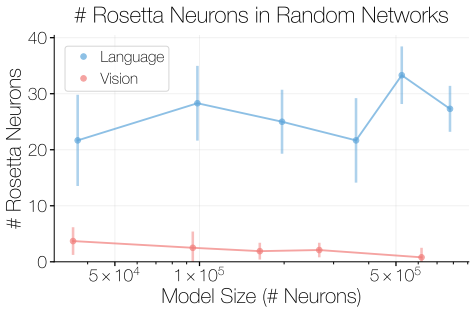


Figure 4: **Rosetta Neuron counts in untrained models lack systematic scaling.**

4.3 An Analytical Model of Rosetta Neuron Scaling

The empirical scaling law in Figure 3 raises a question: why do Rosetta Neurons grow as a sublinear power law, occupying a shrinking fraction of the total neuron population? We adopt a capacity-allocation view of superposition (Elhage et al., 2022; Scherlis et al., 2022): many features compete for representation in a limited number of neuron coordinates. High-importance features, which contribute more to reducing downstream loss, are worth representing more cleanly, while lower-importance features remain mixed in superposition. This gives a natural model of Rosetta Neurons: neurons mostly explained by a single shared feature are reproducible across independently trained networks, whereas neurons that mix many features can do so in network-specific ways and are harder to match. The analytical model below formalizes this intuition; derivations and simulations are in Appendix D.

Feature-isolation setup. We consider the setting where a network with N neuron coordinates remains in a superposition regime as it scales (Liu et al., 2025): it represents $A(N) > N$ latent features with $A(N) = \Omega(N)$, so not every feature can receive a dedicated coordinate. This setting could arise through several non-exclusive mechanisms, which we do not attempt to distinguish here. It could come from effectively infinite data relative to finite model size (Elhage et al., 2022), or from regularization pressures (Liu et al., 2025; Bricken et al., 2023a). We assume a power-law feature-importance spectrum, motivated by prior scaling-law theories in which learnable structure is organized into ranked modes (Michaud et al., 2023; Bordelon et al., 2020; Bahri et al., 2024). Specifically, features have importance $w_r \propto r^{-\beta}$ with $\beta > 1$, where smaller r denotes a more important feature and $\beta > 1$ ensures finite total importance. Intuitively, importance measures a feature's value for reducing loss, reflecting factors such as frequency, predictiveness, and task relevance.

In a superposition regime, a single neuron may contain signal from many features, and a single feature may be distributed across many neurons. We summarize how cleanly feature r is captured by any individual neuron with an isolation score $s_r \geq 0$. Informally, s_r compares how much of the activation variance in the neuron most aligned with feature r comes from feature r, versus from other features mixed into that neuron. Thus, s_r is large when one neuron is dominated by feature r, and small when every neuron containing feature r is strongly mixed with other features. We formalize this quantity in Appendix D, where s_r is derived as a signal-to-interference ratio.

We now use this isolation score to relate superposition to Rosetta Neuron matching across models. Assuming matched networks trained on the same data distribution share a common feature spectrum, Rosetta Neurons can be analyzed through whether a feature is sufficiently isolated in a representative single model. Intuitively, high-isolation features produce reproducible matches because the same shared feature dominates the neuron, whereas low-isolation features are obscured by model-specific mixtures of other tail features. We formalize this intuition in Appendix D.6 by relating s_r to expected cross-model activation correlation under model-specific interference. We call a feature Rosetta-detectable if $s_r \geq \tau$: its shared signal is strong enough relative to model-specific interference to produce a reproducible single-neuron match. We take $\tau > 1$, which ensures in the signal-to-interference model that each detectable feature has a distinct dedicated neuron.

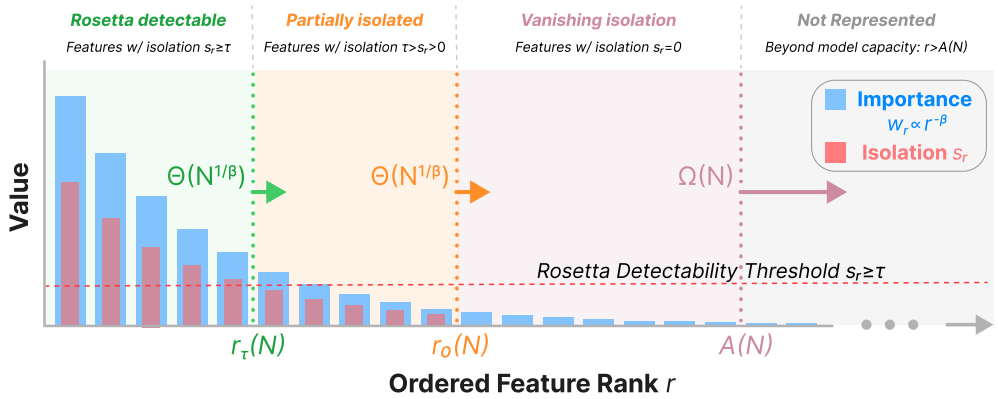


Figure 5: **Feature-isolation frontiers.** Features are ordered by decreasing importance $w_r \propto r^{-\beta}$. The optimal allocation partitions the spectrum into Rosetta-detectable features with $s_r \geq \tau$, partially isolated features with $0 < s_r < \tau$, strongly superposed features with $s_r = 0$, and features beyond the represented set $A(N)$. The frontiers $r_\tau(N)$ and $r_0(N)$ scale as $\Theta(N^{1/\beta})$, yielding the sublinear Rosetta Neuron count $R_\tau(N) = \Theta(N^{1/\beta})$.

Capacity-allocation objective. Given this setup, we model the network as solving an allocation problem with two ingredients. First, detecting a feature from a noisy single-neuron activation has diminishing returns: making a poorly represented feature cleaner is valuable, but further purifying an already-clean feature helps less. Second, the total budget for isolating features scales linearly with the number of coordinates N. This yields:

$$\max_{s_r \geq 0} \sum_r w_r \log(1 + s_r) \qquad \text{s.t.} \qquad \sum_r s_r \leq \kappa N. \tag{2}$$

In Appendix D, we derive the logarithmic utility from a Gaussian channel model in which the neuron most aligned with feature r provides a noisy estimate of that feature's activation. Increasing s_r reduces the optimal negative log-likelihood by an amount proportional to $\log(1 + s_r)$. The linear budget follows from a bounded-activation-energy condition in the underlying linear superposition model. Solving this allocation problem in the continuum limit yields a simple frontier structure:

$$s^\star(r; N) = \left[\left(\frac{r_0(N)}{r} \right)^\beta - 1 \right]_+, \qquad [u]_+ = \max(u, 0). \tag{3}$$

so features below a cutoff rank $r_0(N)$ are more cleanly isolated in individual neurons, while features beyond it remain in superposition with $s_r = 0$. This cutoff is set by the total budget:

$$\kappa N = \int_1^{r_0} \left[\left(\frac{r_0}{r} \right)^\beta - 1 \right] dr = \Theta(r_0^\beta), \tag{4}$$

and hence $r_0(N) = \Theta(N^{1/\beta})$. Let $R_\tau(N)$ count threshold-crossing features, which correspond to Rosetta Neurons in this idealized model. Solving $s^\star(r_\tau; N) = \tau$ for r_τ yields

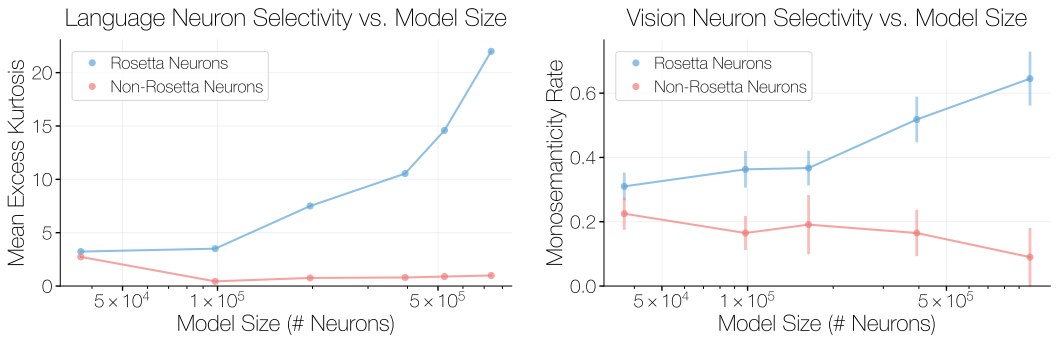
$$r_\tau(N) = r_0(N)(1 + \tau)^{-1/\beta}, \qquad R_\tau(N) = \Theta(r_\tau(N)) = \Theta(N^{1/\beta}). \tag{5}$$

Since $\beta > 1$, the Rosetta count grows as a sublinear power law. Figure 5 summarizes this frontier structure: as model size N increases, the Rosetta-detectability frontier moves outward, while a long tail of lower-importance features remains either partially isolated or strongly superposed with $s_r = 0$.

Prediction: Neuron Polarization. As network scale increases, the Rosetta frontier moves outward in rank, but the features already inside the frontier become cleaner. Averaging the optimal allocation over the Rosetta set and the non-Rosetta tail, respectively, gives

$$\bar{s}_{\text{Rosetta}}(N) = \Theta\left(N^{(\beta-1)/\beta} \right), \qquad \bar{s}_{\text{non-Rosetta}}(N) = O\left(\frac{N^{1/\beta}}{A(N)} \right) \to 0. \tag{6}$$

Thus, Rosetta Neurons become increasingly clean and selective, while the unresolved feature tail remains packed in a polysemantic background. We validate this prediction next.



(a) Vocabulary-space neuron selectivity in Pythia.

(b) VLM-judged monosemanticity in OpenCLIP.

Figure 6: **The Neuron Polarization Effect in language and vision models.** (a) In language models, Rosetta Neurons show increasing mean excess kurtosis of vocabulary-space projections with scale. Non-Rosetta neurons remain near zero, indicating weak selectivity. (b) In vision models, VLM-judged monosemanticity increases with scale for Rosetta Neurons and decreases for non-Rosetta neurons.

5 Properties of Rosetta Neurons

We next investigate how the properties of Rosetta Neurons change with scale. We test the predicted Neuron Polarization Effect across modalities, examine semantic specialization in language models, and conclude with a case study demonstrating that an individual Rosetta Neuron can be both selective and specialized enough to support domain-specific data filtering.

5.1 The Neuron Polarization Effect in Large-Scale Models

We next test the predicted Neuron Polarization Effect: scaling separates increasingly monosemantic Rosetta Neurons from a comparatively polysemantic non-Rosetta background. We observe this trend using heuristic neuron selectivity measures in both language and vision. Since monosemanticity lacks an agreed-upon operational definition (see Section I.2 for discussion), we treat these measures as proxy evidence.

Language models. We measure language-model neuron selectivity using vocabulary-space projections (Gurnee et al., 2024). For each neuron $u = (\ell, c)$, we compute cosine similarities between its output weight W^u_{out} and each token unembedding $W_U[v]$, and use the excess kurtosis of these scores as an output-side selectivity metric. Higher kurtosis indicates concentration on a small set of tokens, suggestive of monosemanticity (Avrahamy et al., 2026). For Pythia, we define Rosetta Neurons using OPT–Pythia matches and compute the mean selectivity separately for Rosetta and non-Rosetta neurons. The Rosetta mean increases with scale, while the non-Rosetta mean stays near zero (Figure 6a), consistent with increasing monosemanticity in the Rosetta population and weak selectivity among the non-Rosetta neurons. The same trend holds for other language families (Appendix E).

Vision models. Since vision models lack a natural vocabulary-space selectivity metric, we use a VLM-as-a-judge proxy for monosemanticity (Shaham et al., 2024). At each scale, we evaluate OpenCLIP neurons from the Diffusion–OpenCLIP Rosetta matches by showing GPT-5.4 each neuron's top 20 activating images, activation maps, and overlays, and asking whether they reflect a single coherent visual feature. We evaluate five disjoint subsets of 100 Rosetta Neurons and five matched non-Rosetta subsets. The monosemanticity rate increases with scale for Rosetta Neurons and decreases for non-Rosetta neurons (Figure 6b), consistent with the Neuron Polarization Effect. Additional results, methodological details, and a metric reliability study are in Appendix H. Section E.2 further shows the same polarization in both modalities under an additional, common proxy.

5.2 Rosetta Neurons Become More Specialized with Scale in Language Models

We now investigate whether Rosetta Neurons become more specialized with scale by measuring their top-activating document types across Pythia models of different sizes. Our analytical model predicts that as the Rosetta frontier expands, lower-ranked features can become sufficiently clean and reproducible to be shared across independently trained models. As rarer data patterns lie deeper in

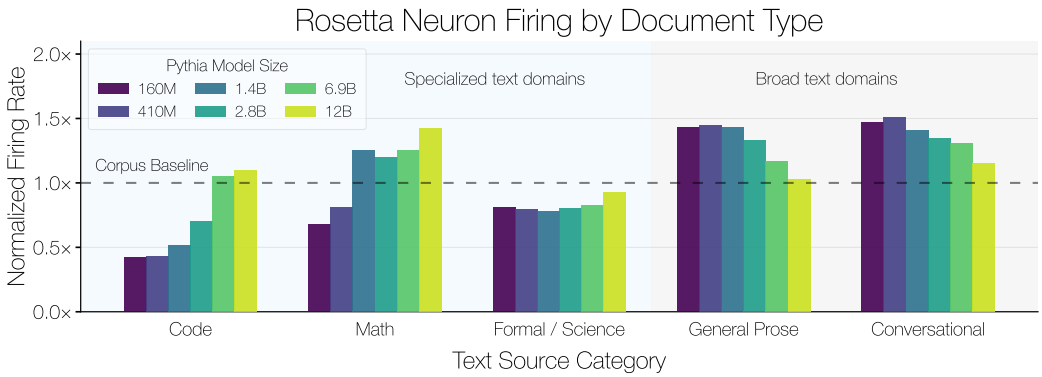


Figure 7: **Rosetta Neuron document-type firing in Pythia.** For each model size, we plot how often top-activating Rosetta Neuron contexts fall into a category, normalized by that category's token frequency in the dataset. The dashed line marks the corpus baseline. With scale, Rosetta Neuron firing shifts toward specialized categories such as code and math.

the feature importance spectrum, larger models should increasingly contain Rosetta Neurons selective for specialized domains, appearing as a scale-dependent shift in document-type firing.

Measuring Rosetta Neuron firing by document type. We measure how Rosetta Neuron firing patterns vary across text categories with model size. At each Pythia scale, we use Rosetta Neurons from the OPT–Pythia scaling runs and collect each neuron's top-20 activating contexts from the Pile validation set. We assign each context to one of five categories: code, math, formal/scientific text, general prose, or conversational text. For each category, we compare its share of Rosetta Neuron top activations with its token share in the validation cache. A value of 1 corresponds to the corpus baseline, where top Rosetta activations fall in a category at the same fraction as that category's token share in the dataset. Additional details on the category construction are in Appendix E.

Rosetta Neurons shift toward specialized domains with scale. Figure 7 shows that Rosetta Neuron firing becomes increasingly concentrated in specialized text domains. Code and math are increasingly overrepresented among top activations relative to their dataset frequency, while broader categories such as general prose and conversational text decline. Since all Pythia models use the same training mixture and token budget, this shift is unlikely to be explained solely by greater exposure to specialized data; instead, it suggests a scale-dependent change in internal organization. Larger models appear to expand the Rosetta population toward specialized domains, rather than merely adding shared neurons for basic features, consistent with qualitative examples in Appendix A. In contrast, a non-Rosetta baseline in Appendix E shows no corresponding shift. We next test whether an individual Rosetta Neuron is selective enough for domain-specific data filtering.

5.3 Testing Rosetta Neuron Selectivity with Data Filtering

We next study specialization at the single-neuron level, testing whether an individual Rosetta Neuron can identify a coherent, specialized domain well enough to support data filtering. In a controlled code-domain setting, we use ground-truth labels to evaluate domain recovery and continued pretraining to test downstream utility.

Data-filtering setup. We use CodeSearchNet (Husain et al., 2019), a multilingual code corpus with source-language labels. We take JavaScript as a representative target domain, whose training split contains roughly 58K parsed code functions totaling 16M tokens. Each filtering method selects functions from the pool under a 16M token budget, and we evaluate recovery by F1 against the true JavaScript subset. We then test how well the recovered data supports learning the target domain. We continue pretraining GPT2-1.5B, which has limited code exposure, on the selected data and report JavaScript test-set perplexity.

Under this matched token-budget setup, we compare four filtering methods. The Rosetta Neuron filter uses a single JavaScript-selective Rosetta Neuron from Pythia-6.9B, discovered in the Pythia–OPT matching runs on the Pile. We score CodeSearchNet training functions by this neuron's activations and select the highest-scoring functions up to the 16M-token budget. We apply the same procedure to a non-Rosetta neuron that activates on JavaScript text, identified on the same Pile cache. We compare

these neuron-based filters against two baselines: a uniform random sample from the multilingual pool and the oracle JavaScript subset. Additional experimental details and baselines are in Appendix F.

Rosetta Neuron selectivity predicts useful target-domain data. Table 1 reports each filter's F1 recovery of the ground-truth JavaScript subset and test-set perplexity over three training runs. All filters improve over the base model, consistent with GPT2-1.5B's limited code exposure. Notably, the Rosetta filter recovers nearly the entire JavaScript subset, achieving 0.98 F1. This selectivity translates into downstream utility: continued pretraining on Rosetta-filtered data reduces test perplexity to 3.02, nearly matching the oracle. In contrast, the non-Rosetta JavaScript neuron improves over random filtering but is less selective and yields weaker downstream gains. The retrieved code functions in Appendix F suggest that the non-Rosetta neuron is less specialized, firing on broader web-programming syntax and JavaScript-adjacent languages rather than specifically JavaScript. While this single-neuron experiment serves as a controlled case study of downstream data filtering, qualitative examples in Appendix A suggest that other Rosetta Neurons exhibit similarly interpretable, selective firing. Beyond individual neurons, the Rosetta population is also functionally important: ablating it sharply degrades model performance (Section E.3).

Method	**F1**	**Test PPL**
Base Model	–	6.73
Random	0.06	3.59 ± 0.07
Non-Rosetta	0.09	3.23 ± 0.07
Rosetta	0.98	3.02 ± 0.05
Oracle	1.00	3.01 ± 0.04

Table 1: **JavaScript filtering.** Matched-budget filters on CodeSearchNet; PPL reports mean $\pm$ 95% CI over three runs.

6 Discussion, Limitations, and Future Work

We studied whether the internal organization of neural networks evolves predictably with scale. Using Rosetta Neurons as a probe, we found sublinear power-law growth of shared single-neuron structure in language and vision models. Together, our analytical model and empirical measurements suggest that scaling polarizes neurons: a shared Rosetta population grows more selective, while the remaining neurons form a larger, superposed background. We further showed that Rosetta Neurons become more specialized with scale, demonstrating this specialization in a controlled data-filtering study for continued pretraining. We close by discussing the limitations and scope of the analysis, along with future directions; additional discussion and failure cases are in Appendix I.

Our analysis targets a specific form of neuron-level structure: Rosetta Neurons. This is only a subset of the structure inside neural networks. Some computations may be carried by circuits (Elhage et al., 2021), attention heads (Olsson et al., 2022), or subspaces that don't appear as neuron correspondences (Wang et al., 2018; Bricken et al., 2023a). Thus, the Rosetta population should not be viewed as a map of all shared computation. We instead view Rosetta Neurons as a tractable observable: a shared neuron-level population for studying structure that is not specific to any single training run, architecture, or model family. Combining universality with scale makes Rosetta Neurons a powerful discovery tool: cross-model recurrence identifies candidate interpretable units without target concepts or auxiliary training, while comparisons across model size expose predictable changes in this structure. However, the resulting trends should be interpreted within the scope of our measurements. For instance, our monosemanticity measurements rely on proxies, and should be viewed as relative population-level trends.

More broadly, our results point toward a bridge between macroscopic scaling behavior and the microscopic organization of learned representations. Standard scaling laws describe external quantities such as loss as a function of scale, while Rosetta Neurons provide a complementary lens on how shared structure can evolve predictably at the level of individual units. This opens several directions: discovering other internal observables whose structure changes predictably with scale, studying how universal neurons emerge during training, how they are transformed by post-training, and how they relate to distributed forms of shared computation. Our analytical model offers a simple account of Rosetta Neuron scaling and polarization, but deriving these phenomena from gradient-based training dynamics remains an important next step. Such an account could explain how universal structure emerges from optimization, and may inform objectives or regularizers that shape not only loss scaling but also the internal organization and adaptability of learned representations.

Acknowledgments. We thank members of Berkeley AI Research and the Redwood Center for Theoretical Neuroscience for helpful discussions. We are particularly grateful to Mason Kamb, Phillip Isola, David Bau, Yizhou Liu, Sophie Wang, Grace Luo, Stephanie Fu, Himanshu Gaurav Singh, Jasmine Shone, Tamar Rott Shaham, and Tyler Bonnen for their thoughtful feedback. YB is a visiting scholar at UC Berkeley and member of the Simons Collaboration on the Physics of Learning & Neural Computation. AD is supported by the US Department of Energy Computational Science Graduate Fellowship. Additional support came from ONR MURI, NSF IIS-2403305, and the Google-BAIR Commons Program.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Asaf Avrahamy, Yoav Gur-Arieh, and Mor Geva. Disentangling mlp neuron weights in vocabulary space. *arXiv preprint arXiv:2604.06005*, 2026.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pp. 1024–1034. PMLR, 2020.

Trenton Bricken, Rylan Schaeffer, Bruno Olshausen, and Gabriel Kreiman. Emergence of sparse representations from noise. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 3148–3191, 2023a.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023b. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Francesco Cagnetta, Allan Raventós, Surya Ganguli, and Matthieu Wyart. Deriving neural scaling laws from the statistics of natural language. *arXiv preprint arXiv:2602.07488*, 2026.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16185–16195, 2025.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2818–2829, 2023.

Tali Dekel, Shaul Oron, Michael Rubinstein, Shai Avidan, and William T Freeman. Best-buddies similarity for robust template matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2021–2029, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1934–1943, 2023.

Shimon Edelman. Representation is representation of similarities. *Behavioral and brain sciences*, 21 (4):449–467, 1998.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. Universal neurons in gpt2 language models. *Transactions on Machine Learning Research*, 2024.

Uri Hasson, Yuval Nir, Ifat Levy, Galit Fuhrmann, and Rafael Malach. Intersubject synchronization of cortical activity during natural vision. *Science*, 303(5664):1634–1640, 2004. doi: 10.1126/science.1089506. URL https://www.science.org/doi/abs/10.1126/science.1089506.

James V Haxby, M Ida Gobbini, Maura L Furey, Alumit Ishai, Jennifer L Schouten, and Pietro Pietrini. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430, 2001.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jing Huang, Daniel Wurgaft, Rachit Bansal, Laura Ruis, Naomi Saphra, David Alvarez-Melis, Andrew Kyle Lampinen, Christopher Potts, and Ekdeep Singh Lubana. Why larger models learn more: Effects of capacity, interference, and rare-task retention. *arXiv preprint arXiv:2605.29548*, 2026.

DH Hubel and TN Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 1962.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning*, 2024.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008a.

Nikolaus Kriegeskorte, Marieke Mur, Douglas A Ruff, Roozbeh Kiani, Jerzy Bodurka, Hossein Esteky, Keiji Tanaka, and Peter A Bandettini. Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126–1141, 2008b.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *ICLR*, 2016. URL http://arxiv.org/abs/1511.07543.

Yizhou Liu, Ziming Liu, and Jeff Gore. Superposition yields robust neural scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=knPz7gtjPW.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. *arXiv preprint arXiv:2601.22158*, 2026.

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31, 2018.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 39643–39666, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Ben Sorscher, Surya Ganguli, and Haim Sompolinsky. Neural representational geometry underlies few-shot concept learning. *Proceedings of the National Academy of Sciences*, 119(43): e2200800119, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. *Advances in neural information processing systems*, 31, 2018.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

Lihe Yang, Shang-Wen Li, Yang Li, Xinjie Lei, Dong Wang, Abdelrahman Mohamed, Hengshuang Zhao, and Hu Xu. In pursuit of pixel supervision for visual pre-training. *arXiv preprint arXiv:2512.15715*, 2025.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Appendix

A Rosetta Neuron Visualizations . . . 17
A.1 Rosetta Neurons across Model Scales . . . 17
A.2 Qualitative Comparison of Rosetta and Non-Rosetta Neurons . . . 22

B Rosetta Neuron Matching Pipeline . . . 24
B.1 MLP Neurons in Transformer Models . . . 24
B.2 Quantifying Pairwise Neuron Similarity . . . 24
B.3 Filtering Pairwise Neuron Correspondences . . . 25
B.4 Aligning Token-Wise Activations Across Models . . . 25
B.5 Computational Complexity of Rosetta Neuron Matching . . . 25

C Further Details on Rosetta-Neuron Scaling . . . 27
C.1 Model Families . . . 27
C.2 Robustness to the Mutual Top-k Criterion . . . 28
C.3 Input-Permutation Null . . . 28
C.4 Disentangling Model Size from Token Exposure . . . 29

D Detailed Derivation of the Rosetta Neuron Scaling Model . . . 29
D.1 A Minimal Capacity-Allocation Model . . . 30
D.2 Solving the Allocation Problem . . . 31
D.3 Prediction: Neuron Polarization . . . 32
D.4 Deriving the Isolation Score from Superposition . . . 33
D.5 Deriving the Logarithmic Utility and Linear Isolation Budget . . . 35
D.6 Connection to Empirical Rosetta Matching . . . 36
D.7 Synthetic Validation of the Analytical Model . . . 38

E Additional Results on Rosetta Neuron Properties . . . 39
E.1 Additional Vocabulary-Space Selectivity Results . . . 39
E.2 A Common Selectivity Measure Across Modalities . . . 39
E.3 Testing Rosetta Neuron Functional Importance . . . 40
E.4 Document-Type Firing Analysis . . . 41
E.5 Depth-Wise Distribution of Rosetta Neurons . . . 41

F Data Filtering Experimental Details . . . 43

G Dataset Ablations . . . 48
G.1 Ablation on the Number of Tokens Used for Language Model Matching . . . 48
G.2 Ablation on the Number of Images Used for Vision Model Matching . . . 48
G.3 Ablation on the Image Distribution Used for Vision Model Matching . . . 48

H Additional Details on VLM-as-a-Judge . . . 50
H.1 Detailed Experimental Setup . . . 50
H.2 Sensitivity to the Number of Top-k Activating Images . . . 51
H.3 Validation of VLM-as-a-Judge as a Predictive Metric . . . 51
H.4 Results for Neuron Selectivity in Other Vision Models . . . 52

I Further Discussion and Limitations . . . 52
I.1 DINOv3 as an Informative Failure Case . . . 52
I.2 Operationalizing Monosemanticity and Polysemanticity . . . 53

J Compute Resources . . . 53

A Rosetta Neuron Visualizations

A.1 Rosetta Neurons across Model Scales

We provide qualitative examples of Rosetta Neurons across model scales in both language and vision. For language models, we show LLM annotations for a subset of Rosetta Neurons across matched model families along with representative top-activating input spans (Figs. 8–13). The system prompt used for the LLM annotator is provided below. For vision models, we visualize the top three activating images and corresponding activation maps for a subset of Rosetta Neurons at different scales (Figs. 14–18). We report neuron identities in the captions using the format `L{layer}/U{unit}`, together with the coarse model names. Detailed model names are provided in Appendix C.1.

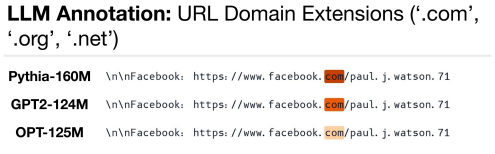
```
You are evaluating what a neuron in a language model responds to.  Below are the top 20 sequences that
most strongly activate this neuron that is shared across multiple language models.  Each box shows a text
sequence with a heatmap overlay indicating where the corresponding neuron in each model fires (bright = strong
activation). Give an overall summary of the concept the neuron responds to in less than 10 words and give up
to three examples of what sort of tokens it would fire on.
```



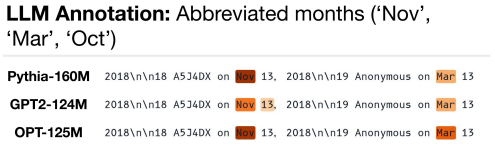
(a) Pythia-160M: L5/U1767. GPT2-124M: L7/U1625. OPT-125M: L1/U2325.



(b) Pythia-160M: L5/U1898. GPT2-124M: L8/U557. OPT-125M: L8/U2115.

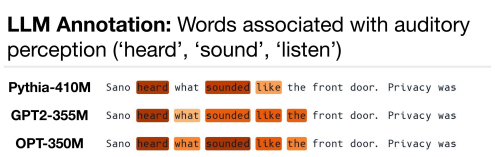


(c) Pythia-160M: L10/U2431. GPT2-124M: L11/U424. OPT-125M: L8/U2837.



(d) Pythia-160M: L0/U2626. GPT2-124M: L9/U649. OPT-125M: L0/U888.

Figure 8: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-160M, GPT2-124M, OPT-125M.



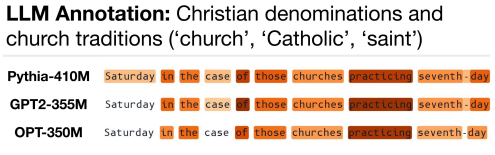
(a) Pythia-410M: L10/U3047. GPT2-355M: L19/U1698. OPT-350M: L12/U927.



(b) Pythia-410M: L9/U1693. GPT2-355M: L15/U1724. OPT-350M: L13/U846.

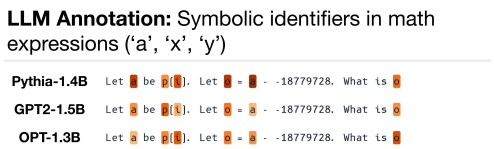


(c) Pythia-410M: L23/U39. GPT2-355M: L23/U149. OPT-350M: L23/U2921.

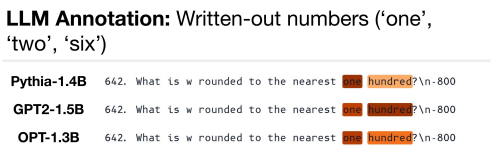


(d) Pythia-410M: L23/U3863. GPT2-355M: L23/U1026. OPT-350M: L22/U1841.

Figure 9: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-410M, GPT2-355M, OPT-350M.



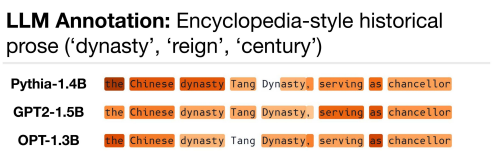
(a) Pythia-1.4B: L2/U3564. GPT2-1.5B: L12/U1978. OPT-1.3B: L3/U652.



(b) Pythia-1.4B: L9/U835. GPT2-1.5B: L19/U2508. OPT-1.3B: L17/U7443.



(c) Pythia-1.4B: L15/U2969. GPT2-1.5B: L34/U491. OPT-1.3B: L17/U3726.



(d) Pythia-1.4B: L15/U189. GPT2-1.5B: L35/U1327. OPT-1.3B: L18/U4373.

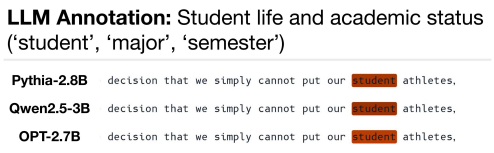
Figure 10: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-1.4B, GPT2-1.5B, OPT-1.3B.



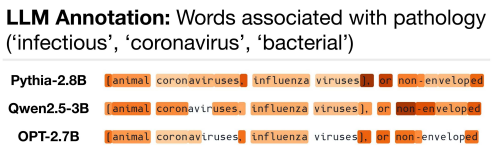
(a) Pythia-2.8B: L20/U9199. Qwen2.5-3B: L31/U5812. OPT-2.7B: L25/U7525.



(b) Pythia-2.8B: L22/U6182. Qwen2.5-3B: L31/U1823. OPT-2.7B: L27/U3916.

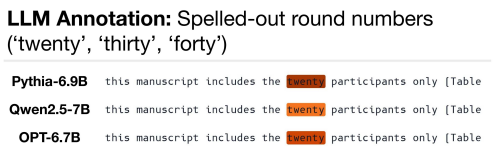


(c) Pythia-2.8B: L12/U5719. Qwen2.5-3B: L26/U2175. OPT-2.7B: L21/U7710.

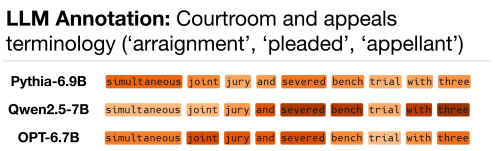


(d) Pythia-2.8B: L28/U10099. Qwen2.5-3B: L31/U6319. OPT-2.7B: L31/U2355.

Figure 11: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-2.8B, Qwen2.5-3B, OPT-2.7B.



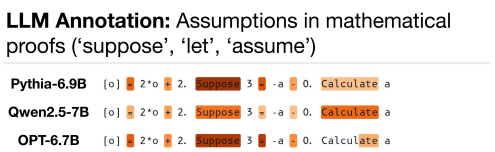
(a) Pythia-6.9B: L1/U1161. Qwen2.5-7B: L22/U12072. OPT-6.7B: L2/U7258.



(b) Pythia-6.9B: L14/U15135. Qwen2.5-7B: L23/U5673. OPT-6.7B: L26/U14602.

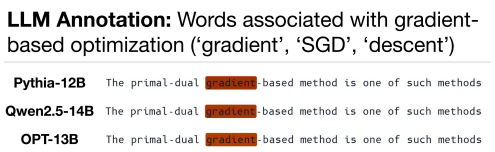


(c) Pythia-6.9B: L16/U11168. Qwen2.5-7B: L22/U14665. OPT-6.7B: L26/U2660.

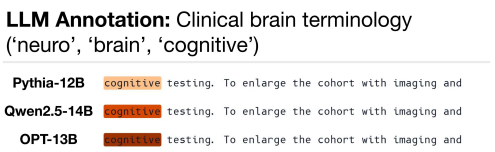


(d) Pythia-6.9B: L27/U826. Qwen2.5-7B: L26/U5796. OPT-6.7B: L29/U14520.

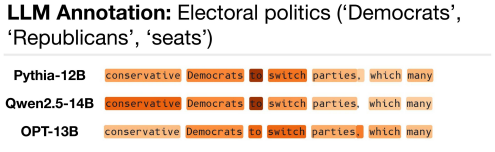
Figure 12: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-6.9B, Qwen2.5-7B, OPT-6.7B.



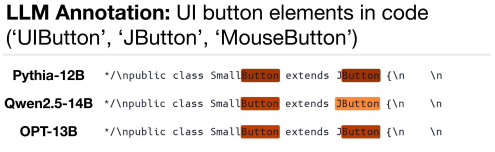
(a) Pythia-12B: L1/U17927. Qwen2.5-14B: L0/U3786. OPT-13B: L0/U4028.



(b) Pythia-12B: L1/U19682. Qwen2.5-14B: L3/U1521. OPT-13B: L0/U2590.

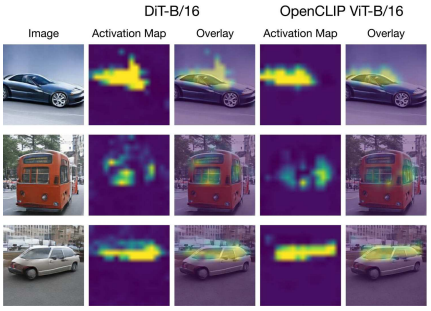


(c) Pythia-12B: L24/U10513. Qwen2.5-14B: L33/U11681. OPT-13B: L29/U14.

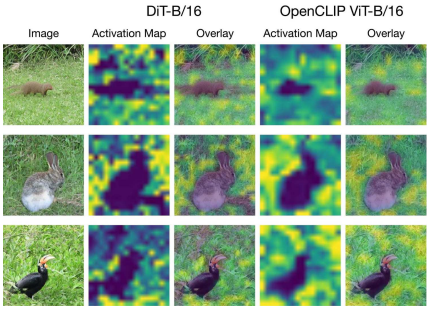


(d) Pythia-12B: L1/U4031. Qwen2.5-14B: L0/U6188. OPT-13B: L3/U11416.

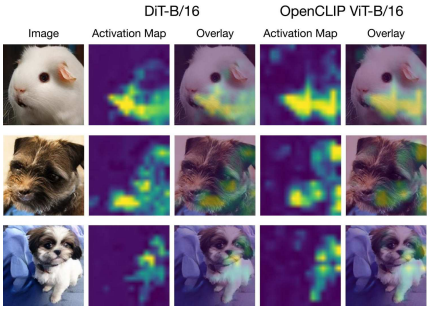
Figure 13: **LLM annotations for Rosetta Neurons.** Comparison between Pythia-12B, Qwen2.5-14B, OPT-13B.



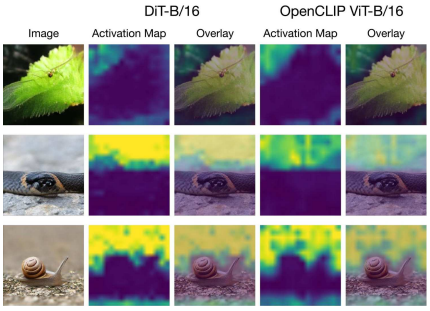
(a) DiT-B/16: L7/U879. OpenCLIP ViT-B/16: L7/U1879.



(b) DiT-B/16: L11/U610. OpenCLIP ViT-B/16: L4/U415.

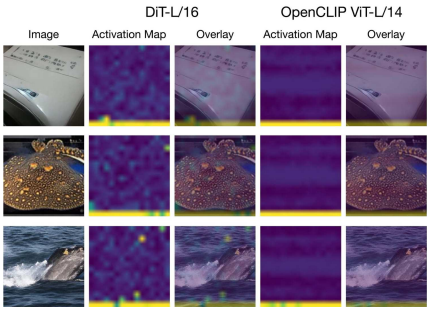


(c) DiT-B/16: L11/U1641. OpenCLIP ViT-B/16: L4/U730.

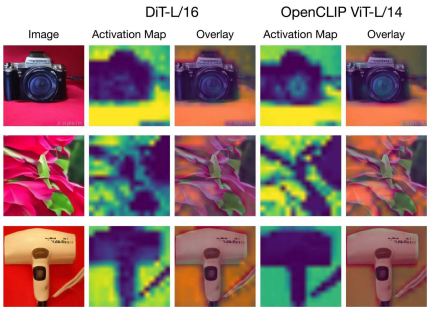


(d) DiT-B/16: L10/U1516. OpenCLIP ViT-B/16: L4/U689.

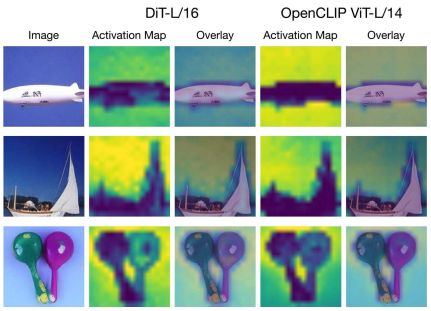
Figure 14: **Top-activating images for Rosetta Neurons.** Comparison between DiT-B/16 and OpenCLIP ViT-B/16.



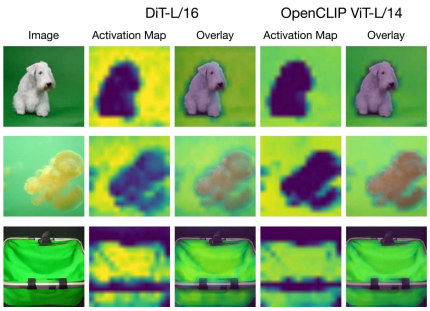
(a) DiT-L/16: L0/U1228. OpenCLIP ViT-L/14: L1/U1669.



(b) DiT-L/16: L31/U15. OpenCLIP ViT-L/14: L0/U1207.

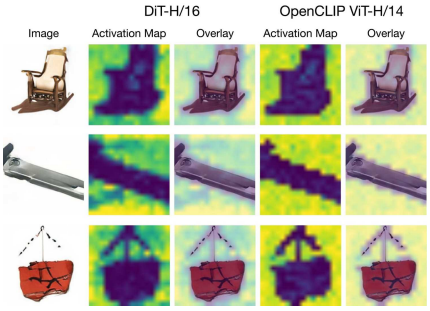


(c) DiT-L/16: L31/U1752. OpenCLIP ViT-L/14: L0/U3894.

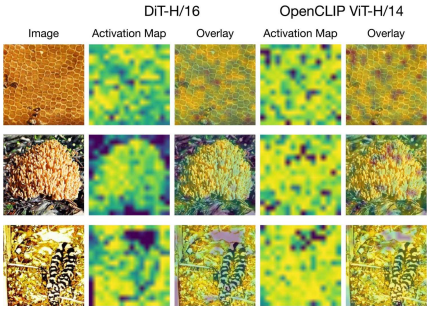


(d) DiT-L/16: L31/U816. OpenCLIP ViT-L/14: L0/U2394.

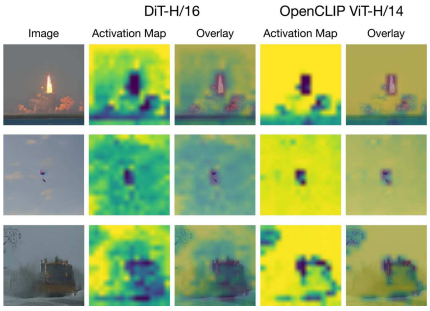
Figure 15: **Top-activating images for Rosetta Neurons.** Comparison between DiT-L/16 and OpenCLIP ViT-L/14.



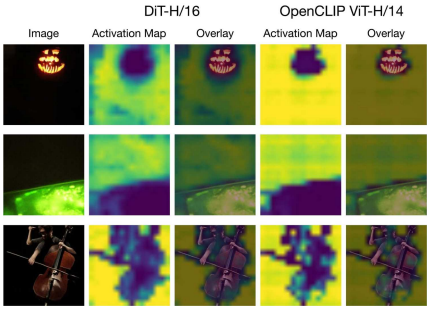
(a) DiT-H/16: L45/U1346. OpenCLIP ViT-H/14: L1/U3383.



(b) DiT-H/16: L47/U1097. OpenCLIP ViT-H/14: L1/U4324.

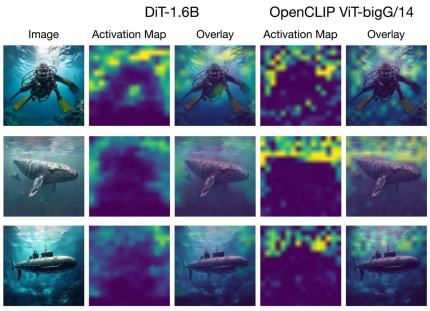


(c) DiT-H/16: L47/U1230. OpenCLIP ViT-H/14: L0/U3495.

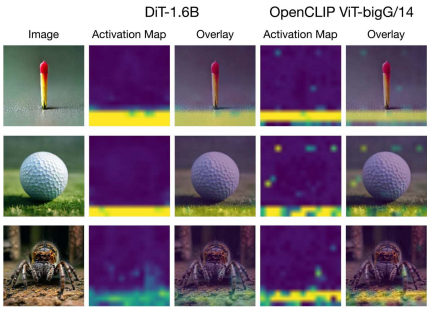


(d) DiT-H/16: L47/U300. OpenCLIP ViT-H/14: L0/U477.

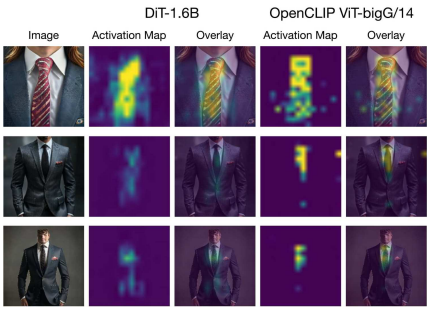
Figure 16: **Top-activating images for Rosetta Neurons.** Comparison between DiT-H/16 and OpenCLIP ViT-H/14.



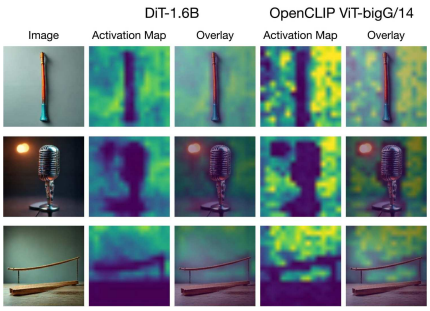
(a) DiT-1.6B: L44/U2447. OpenCLIP ViT-bigG/14: L29/U2762.



(b) DiT-1.6B: L12/U2668. OpenCLIP ViT-bigG/14: L20/U203.



(c) DiT-1.6B: L6/U3382. OpenCLIP ViT-bigG/14: L39/U7045.

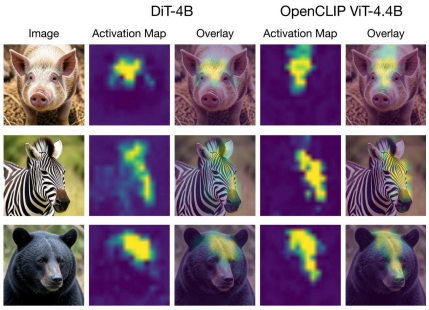


(d) DiT-1.6B: L9/U2940. OpenCLIP ViT-bigG/14: L28/U2265.

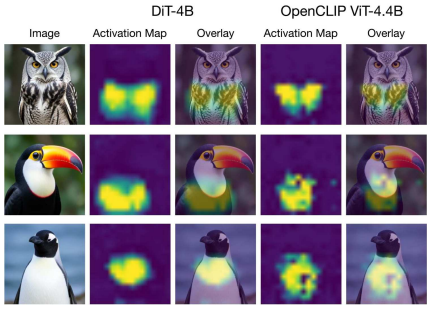
Figure 17: **Top-activating images for Rosetta Neurons.** Comparison between DiT-1.6B and OpenCLIP ViT-bigG/14.



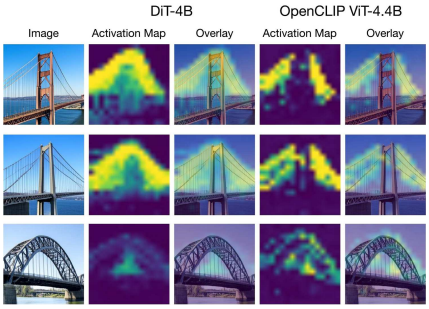
(a) DiT-4B: L8/U4935. OpenCLIP ViT-4.4B: L41/U7020.



(b) DiT-4B: L2/U4833. OpenCLIP ViT-4.4B: L36/U14930.



(c) DiT-4B: L9/U3974. OpenCLIP ViT-4.4B: L45/U10399.



(d) DiT-4B: L9/U8380. OpenCLIP ViT-4.4B: L41/U5498.

Figure 18: **Top-activating images for Rosetta Neurons.** Comparison between DiT-4B and OpenCLIP ViT-4.4B.

A.2 Qualitative Comparison of Rosetta and Non-Rosetta Neurons

We compare Rosetta and non-Rosetta neurons by visualizing top-activating examples. In language, we use randomly selected Pythia-6.9B Rosetta Neurons and non-Rosetta neurons from the same layers (Figures 19 and 20); in vision, we conduct a similar comparison for OpenCLIP ViT-L/14 (Figures 21 and 22). Rosetta Neurons appear more coherent and selective, whereas non-Rosetta neurons often respond to a broader mixture of features. This small sample is not standalone evidence, but it is consistent with the quantitative selectivity trends in Section 5.

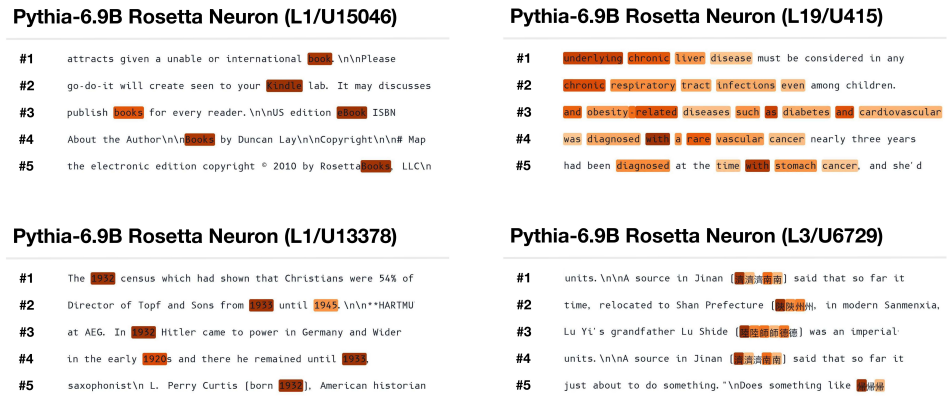


Figure 19: **Top-5 activating sequences for Pythia-6.9B Rosetta Neurons.** Rosetta Neurons demonstrate selective firing for coherent concepts.



Figure 20: **Top-5 activating sequences for Pythia-6.9B non-Rosetta neurons.** Neurons are randomly selected from the same layers as those in Figure 19. These neurons appear polysemantic, firing for a variety of unrelated concepts.

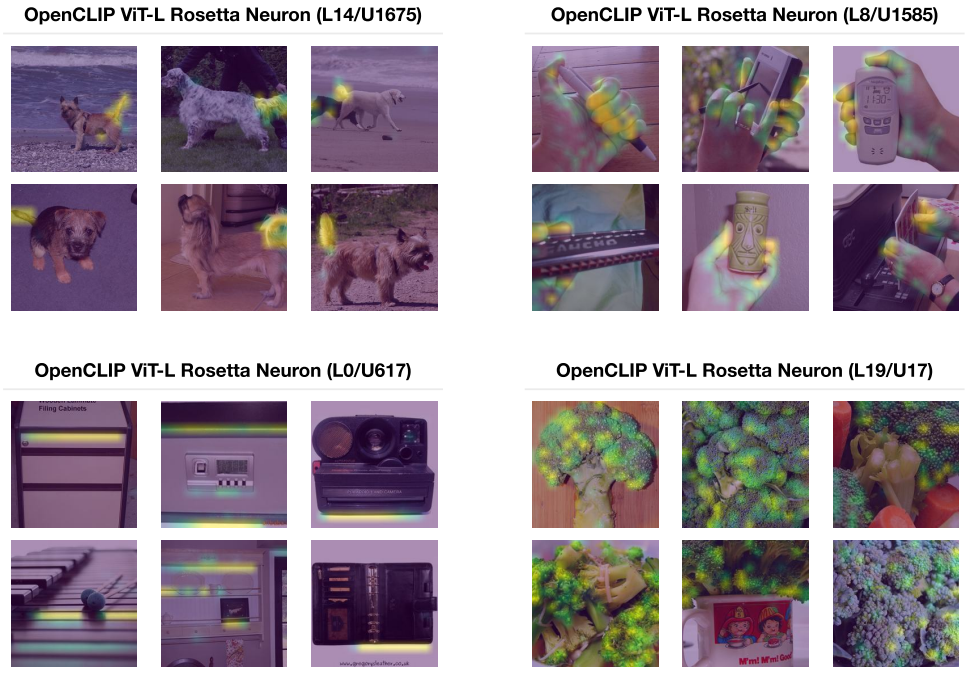


Figure 21: **Top-5 activating images for OpenCLIP ViT-L/14 Rosetta Neurons.** Rosetta Neurons demonstrate selective firing for coherent concepts.

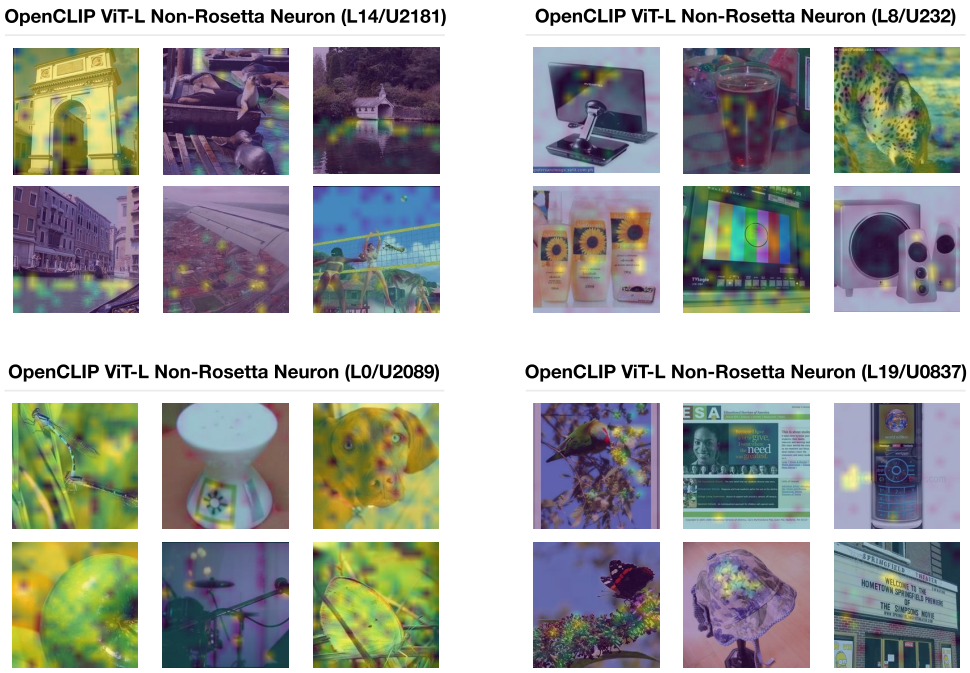


Figure 22: **Top-5 activating images for OpenCLIP ViT-L/14 non-Rosetta neurons.** Neurons are randomly selected from the same layers as those in Figure 21. These neurons appear polysemantic, firing for a variety of unrelated concepts.

B Rosetta Neuron Matching Pipeline

In this section, we recap and provide further details on how to identify Rosetta Neurons – common neurons with similar responses across models. We define MLP neurons and the token-wise activations used to compare them, then describe how to measure pairwise similarity and filter these similarities into reliable shared neuron pairs.

B.1 MLP Neurons in Transformer Models

We study both language and vision models built on the Transformer architecture (Vaswani et al., 2017). Given an input sequence of text tokens or image patches, the model maps each token to an embedding, producing a sequence $e_1, \ldots, e_T \in \mathbb{R}^d$. These are then processed through a series of blocks that alternate between multi-head self-attention and multilayer perceptron (MLP) layers. We focus exclusively on the MLP layers. Let $h_t^{(\ell)}$ denote the hidden state at token position t in layer ℓ. The MLP first applies an affine transformation followed by an element-wise nonlinearity ϕ:

$$m_t^{(\ell)} = \phi\Big(W_{\text{in}}^{(\ell)} h_t^{(\ell)} + b_{\text{in}}^{(\ell)}\Big) \in \mathbb{R}^{d_{\text{mlp}}}, \tag{7}$$

It then projects this activation to the residual stream as $\tilde{h}_t^{(\ell)} = W_{\text{out}}^{(\ell)} m_t^{(\ell)} + b_{\text{out}}^{(\ell)}$. Each coordinate $m_t^{(\ell,c)}$ of the activation vector is a *neuron*, indexed by layer ℓ and channel c.

B.2 Quantifying Pairwise Neuron Similarity

Let model A and model B be two Transformers of the same modality. We run the same dataset of inputs $\mathcal{X} = \{x_i\}_{i=1}^n$ through both models, where each x_i is either an image or a sequence of text. The Transformer processes each input jointly to produce token-wise activations, which are then aligned across models. In vision models, activations are aligned to a common spatial grid, while in language models they are aligned to a shared sequence of text positions. Details of this alignment procedure are given in Appendix B.4.

Consider a neuron $u = (\ell_A, c_A)$ in model A and a neuron $v = (\ell_B, c_B)$ in model B. Let $m_t^u(x_i)$ and $m_t^v(x_i)$ denote their activations at aligned token t for input x_i. We compare neurons u and v by computing the Pearson correlation between these activations on all aligned tokens across the dataset:

$$\text{sim}(u, v) = \frac{1}{N} \sum_{i,t} \frac{m_t^u(x_i) - \mu_u}{\sigma_u} \cdot \frac{m_t^v(x_i) - \mu_v}{\sigma_v}, \tag{8}$$

where μ_u and σ_u^2 are the empirical mean and variance for neuron u over the dataset of N total aligned tokens, and μ_v and σ_v^2 are defined analogously for neuron v. We compute this similarity for all neuron pairs between models A and B.

B.3 Filtering Pairwise Neuron Correspondences

To detect reliable matches between neurons, we retain only those pairs that are nearest neighbors of one another under our similarity metric. Specifically, for neuron $u \in \mathcal{N}(A)$ and $v \in \mathcal{N}(B)$, we include the match (u, v) in $\mathcal{R}(A, B)$ iff $v \in \text{NN}_k(u; B)$ and $u \in \text{NN}_k(v; A)$, where $\text{NN}_k(u; B)$ denotes the top-k neurons in model B most similar to u, and analogously for $\text{NN}_k(v; A)$. Unless otherwise stated, all experiments use the default setting $k = 1$; we ablate this choice in Appendix C.2. Mutual nearest-neighbor matching provides a simple way to retain robust correspondences while filtering asymmetric or noisy nearest-neighbor matches (Dekel et al., 2015). We therefore interpret the pairs of neurons in $\mathcal{R}(A, B)$ as common units across models, and refer to them as *Rosetta Neurons*.

B.4 Aligning Token-Wise Activations Across Models

In this section, we describe how activations are mapped to a shared set of aligned positions. This enables direct comparison of neuron activations across models, even when they use different tokenizers or patch grids.

Language models. The same text may be tokenized differently across language models. To obtain a tokenizer-independent alignment, we represent the input text in UTF-8 byte space. For a text sequence x, let the tokenizers for models A and B split this into tokens $a_1, \ldots, a_{T_A}$ and $b_1, \ldots, b_{T_B}$, respectively. Each token is associated with a contiguous byte span in the original text. We then define a canonical sequence of byte spans by taking the shared byte boundaries induced by the two tokenizations. These are represented by the dotted red lines in Figure 23. This procedure results in aligned text positions $t = 1, \ldots, T^*$ that are independent of either tokenizer. For each canonical span t, we mean pool the activations of any tokens overlapping that span, yielding one activation value $m_t^u(x)$ per neuron u and aligned position t. These pooled activations are then used for comparison according to the neuron similarity metric defined in Equation (8).

Vision models. Vision Transformers may use different patch sizes or input resolutions, producing activations on different spatial grids. For an image x, let model A produce patch-token activations on a grid of size $H_A \times W_A$, and model B produce activations on a grid of size $H_B \times W_B$. We choose one of these grids as a canonical grid of size $H^* \times W^*$. If a model's native grid differs from the canonical grid, we reshape its patch-token activations into a spatial feature map and resample that map to resolution $H^* \times W^*$ using bilinear interpolation as illustrated in Figure 24. This yields an aligned activation map $m_{p,q}^u(x)$ for each neuron u, defined on the same spatial cells (p, q). We then flatten the canonical grid into a sequence of aligned spatial positions $t = 1, \ldots, H^*W^*$, and use the resulting activations for neuron comparison according to Equation (8). In practice, we remove any non-spatial prefix tokens such as class or register tokens, and then bilinearly interpolate the activation maps to have the same spatial dimensions according to the maximum of the two map sizes.

B.5 Computational Complexity of Rosetta Neuron Matching

We provide here a formal analysis of the computational cost for Rosetta Neuron matching. Let T denote the number of aligned tokens and let N_A and N_B denote the numbers of MLP neurons in model A and model B. We first compute per-neuron means and variances over the T tokens and z-score the activations. These statistics are calculated in a single streaming pass with time complexity $O(T(N_A + N_B))$. Pearson correlation computation for all $N_A N_B$ neuron pairs over the aligned tokens is $O(TN_A N_B)$, quadratic in the number of neurons when the two models are

similar in size. Our implementation does not materialize the full $N_A \times N_B$ correlation matrix. Correlations are computed blockwise, with per-block top-k candidates merged into a global top-k list per neuron. For each neuron in model A we retain only its top-k most similar neurons in model B, and for each neuron in model B its top-k most similar neurons in model A. Intersecting these two lists to obtain mutual matches is $O(k(N_A + N_B))$. Code for matching can be found at: https://github.com/avdravid/rosetta-neuron-scaling.

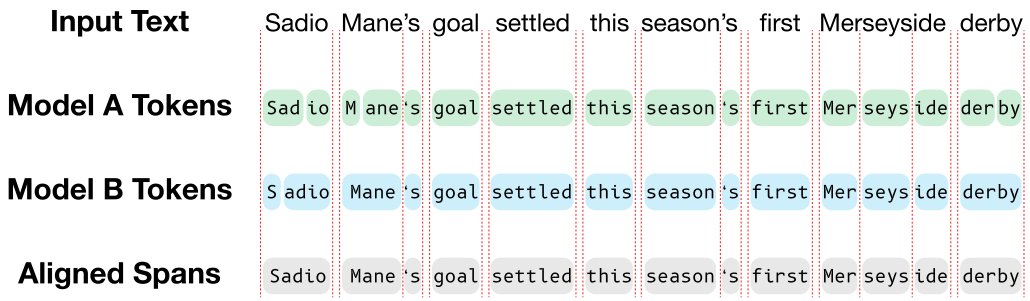


Figure 23: **Aligning text tokens via shared byte boundaries.** We align the tokens from Model A and Model B by keeping only the byte boundaries that both tokenizers share (red dashed lines). By finding the tokens that live in these new *Aligned Spans*, we can pool the activations on these tokens, creating a shared set of positions that we can then use to compare neuron responses.

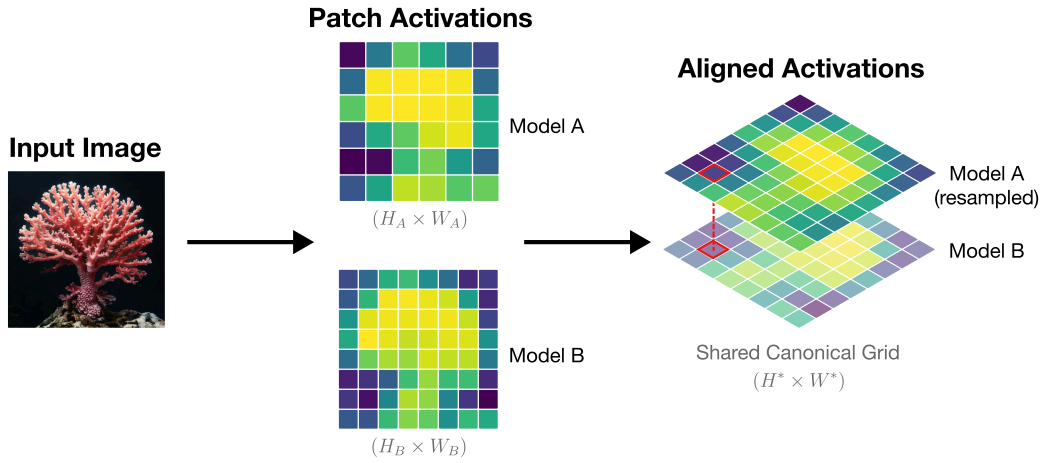


Figure 24: **Aligning spatial grids for neuron comparison.** We align the different patch grids from Model A and Model B by choosing a single target resolution (the canonical grid defined by Model B in this case). Any mismatched native grids are resampled using bilinear interpolation to fit this shared grid. This results in aligned activation maps that can be used for measuring neuron similarity.

C Further Details on Rosetta-Neuron Scaling

We provide additional details on the model families used in the Rosetta Neuron scaling experiments from Section 4. We then present robustness checks for the matching procedure, including an ablation of the mutual-k nearest-neighbor criterion and a dataset-permutation null. Finally, we conduct a controlled analysis of Rosetta Neuron scaling in Pythia checkpoints that separates model-size scaling from the number of training tokens seen.

C.1 Model Families

Language Model Families. For language models, we only consider pretrained models that have not undergone post-training. We conduct the neuron matching using models from the Pythia, GPT-2, OPT, and Qwen-2.5 model families, spanning roughly 100 million to 30 billion parameters (Radford et al., 2019; Biderman et al., 2023; Zhang et al., 2022; Yang et al., 2024). All models are included in Table 2.

Vision Model Families. We analyze discriminative vision models from the OpenCLIP, DINOv2, and Pixio families, spanning scales from approximately 80 million to 5 billion parameters (Cherti et al., 2023; Radford et al., 2021; Oquab et al., 2024; Yang et al., 2025). For the generative model, we use one-step diffusion models built on the Diffusion Transformer architecture (Peebles & Xie, 2023). We use models from the Pixel Mean Flow (pMF), Sana, and Flux families for diffusion (Lu et al., 2026; Chen et al., 2025; Black Forest Labs, 2025). All models are included in Table 3.

Run #	Models	Run #	Models
1	OPT-1.3B, Qwen2.5-1.5B	12	Pythia-160M, GPT2-124M
2	OPT-2.7B, Qwen2.5-3B	13	Pythia-410M, GPT2-355M
3	OPT-6.7B, Qwen2.5-7B	14	Pythia-1.4B, GPT2-1.5B
4	OPT-13B, Qwen2.5-14B	15	OPT-125M, Pythia-160M, GPT2-124M
5	OPT-30B, Qwen2.5-32B	16	OPT-350M, Pythia-410M, GPT2-355M
6	Pythia-160M, OPT-125M	17	OPT-1.3B, Pythia-1.4B, GPT2-1.5B
7	Pythia-410M, OPT-350M	18	Pythia-1.4B, OPT-1.3B, Qwen2.5-1.5B
8	Pythia-1.4B, OPT-1.3B	19	Pythia-2.8B, OPT-2.7B, Qwen2.5-3B
9	Pythia-2.8B, OPT-2.7B	20	Pythia-6.9B, OPT-6.7B, Qwen2.5-7B
10	Pythia-6.9B, OPT-6.7B	21	Pythia-12B, OPT-13B, Qwen2.5-14B
11	Pythia-12B, OPT-13B		

Table 2: Model combinations used in the Rosetta Neuron scaling laws in Figure 3a.

Run #	Models	Run #	Models
1	pMF DiT-B/16, DINOv2 ViT-B/14	12	Sana DiT-1.6B, OpenCLIP ViT-bigG/14
2	pMF DiT-L/16, DINOv2 ViT-L/14	13	Flux.2-Klein-DiT-4B, OpenCLIP EVA-02-CLIP-E/14
3	pMF DiT-H/16, DINOv2 ViT-g/14	14	pMF DiT-B/16, Pixio ViT-B/16, OpenCLIP ViT-B/16
4	pMF DiT-B/16, Pixio ViT-B/16	15	pMF DiT-L/16, Pixio ViT-L/16, OpenCLIP ViT-L/14
5	pMF DiT-L/16, Pixio ViT-L/16	16	pMF DiT-H/16, Pixio ViT-H/16, OpenCLIP ViT-H/14
6	pMF DiT-H/16, Pixio ViT-H/16	17	Sana DiT-1.6B, Pixio ViT-1B/16, OpenCLIP ViT-bigG/14
7	Sana DiT-1.6B, Pixio ViT-1B/16	18	Flux.2-DiT-4B, Pixio ViT-5B/16, OpenCLIP EVA-02-E/14
8	Flux.2-Klein-DiT-4B, Pixio ViT-5B/16	19	pMF DiT-B/16, DINOv2 ViT-B/14, OpenCLIP ViT-B/16
9	pMF DiT-B/16, OpenCLIP ViT-B/16	20	pMF DiT-L/16, DINOv2 ViT-L/14, OpenCLIP ViT-L/14
10	pMF DiT-L/16, OpenCLIP ViT-L/14	21	pMF DiT-H/16, DINOv2 ViT-g/14, OpenCLIP ViT-H/14
11	pMF DiT-H/16, OpenCLIP ViT-H/14		

Table 3: Model combinations used in the Rosetta Neuron scaling laws in Figure 3b.

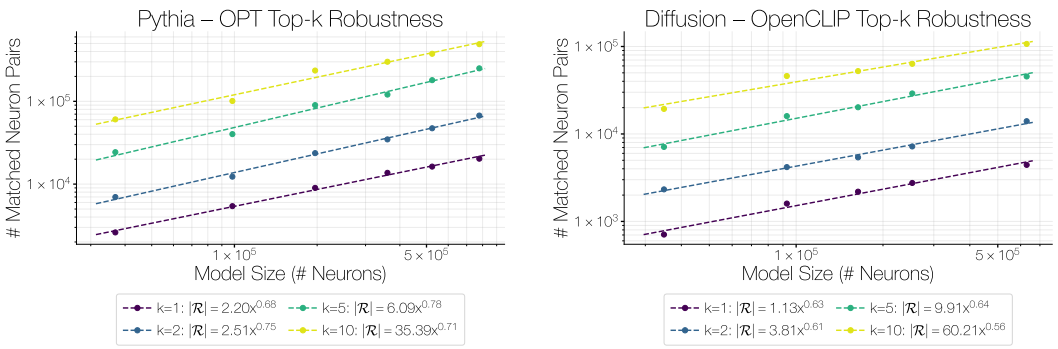


Figure 25: **Robustness to the mutual top-k matching criterion.** We repeat the scaling analysis for Pythia–OPT and Diffusion–OpenCLIP for different values of k in the nearest neighbor criterion. Increasing k results in more discovered neuron pairs, but the fitted power-law exponents remain within a narrow sublinear range.

C.2 Robustness to the Mutual Top-k Criterion

Our main scaling experiments in Section 4 identify Rosetta Neurons using mutual top-1 nearest-neighbor matching. To test whether the observed scaling behavior depends on this particular choice, we repeat the scaling analysis while varying the mutual top-k parameter on one representative scaling trajectory per modality: Pythia–OPT for language models and Diffusion–OpenCLIP for vision models. As seen in Figure 25, increasing k makes the matching rule more permissive and increases the absolute number of discovered neuron pairs. However, across all tested values of k, the fitted power-law exponents remain within a narrow sublinear range in both modalities. This suggests that the Rosetta Neuron scaling trend is not an artifact of the default top-k matching rule. It is also consistent with the detectability-threshold view in our analytical model (Appendix D): varying the permissiveness of the matching criterion changes which features are counted as detectable and changes the prefactor of the scaling law, but should not change the underlying scaling exponent.

C.3 Input-Permutation Null

In Section 4.2, we applied the neuron matching pipeline to untrained random networks of different sizes to test whether Rosetta Neuron scaling laws could arise from the matching procedure. That baseline suggests the trend is not simply due to architecture, initialization, or the larger number of candidate neurons in bigger networks. Here, we test whether the observed scaling could still arise from activation marginals or high-dimensional nearest-neighbor effects even when input alignment is corrupted.

Power-law scaling is absent under dataset permutation. For each model pair, we first compute token-level activations on the same dataset used in the main experiments. Before computing cross-model correlations, we randomly permute the flattened activation positions for one model so that correlations are computed between mismatched input positions across the two models. This preserves each neuron's marginal activation distribution while corrupting input-wise alignment. We then apply the same mutual nearest-neighbor matching procedure with $k = 1$ as in the main experiments. We run this null over three random permutations at each model scale and report means with 95% confidence intervals in Figure 26. Under this null, the number of discovered Rosetta Neurons collapses to roughly 20–100 matches. Moreover, these counts no longer exhibit the systematic sublinear power-law trend observed with aligned activations. This suggests that Rosetta Neuron scaling depends on shared responses to the aligned inputs, rather than being induced by the matching procedure or activation statistics alone.

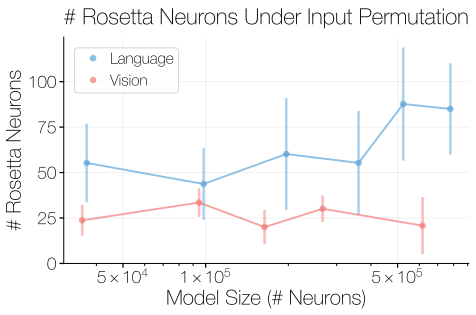


Figure 26: **Rosetta Neurons lack systematic scaling under input permutation.**

Model	# Neurons	Training step (tokens seen)						
		0 (0)	1K (2B)	10K (21B)	25K (52B)	50K (105B)	100K (210B)	143K (300B)
14M	3,072	51	254	194	513	491	505	517
31M	6,144	101	630	704	784	790	770	778
70M	12,288	143	417	1,072	1,394	1,498	1,512	1,535
160M	36,864	97	5,192	2,905	3,186	3,226	3,173	3,156
410M	98,304	106	9,770	3,787	5,154	5,125	5,195	5,178
Rosetta Exponent α		0.14	1.11	0.83	0.69	0.69	0.69	0.68
Fit R^2		0.26	0.90	0.92	0.99	0.99	0.99	0.99

Table 4: **Rosetta Neuron counts in Pythia at matched training-token exposure.** We calculate the number of Rosetta Neurons between two matched-size Pythia seeds across different model sizes and training checkpoints. By roughly 25K steps ($\sim$50B tokens), Rosetta Neuron counts start to stabilize and a sublinear power law in model size emerges ($\alpha \approx 0.68$–0.69, $R^2 \approx 0.99$).

C.4 Disentangling Model Size from Token Exposure

Across the model families in Section 4.2, model size co-varies with other aspects of training, such as the number of tokens seen, the number of optimization steps, and the general training recipe. The observed scaling law could therefore reflect greater training or data exposure rather than model size itself. To separate these factors, we perform a controlled analysis within a single model family in which data distribution, training objective, and recipe are held fixed and token exposure is matched across model sizes.

We conduct this analysis using intermediate checkpoints from two Pythia seeds at five sizes (14M–410M) and seven training stages (0–143K steps). Because every size uses roughly 2M tokens per step, checkpoints at the same step have matched token exposure. At each step, we identify Rosetta Neurons between two seeds of each model size. Then, holding training step and token exposure fixed, we fit power laws for the Rosetta counts as a function of model size (# of neurons) using ordinary least squares in log–log space. Table 4 shows that the model-size scaling relationship stabilizes early in training by roughly 25K steps, which corresponds to approximately 50B tokens seen. From that point onward, the relationship with model size follows a stable sublinear power law, with an exponent of roughly 0.68-0.69 with $R^2 \approx 0.99$.

In contrast, increasing token exposure through training does not produce a comparable scaling law at fixed model size. The Rosetta count changes early in training and largely stabilizes with some small non-monotonic variation. We therefore interpret the Rosetta scaling law as a model-size scaling regularity that emerges after sufficient training and cannot be explained by token exposure alone.

D Detailed Derivation of the Rosetta Neuron Scaling Model

This appendix gives the formal version of the analytical model for Rosetta Neuron scaling in Section 4.3. The model provides an explanation for the empirical observation that the number of Rosetta Neurons grows as a sublinear power law with scale. It further predicts that scaling separates an increasingly selective Rosetta population from a more crowded non-Rosetta background, a phenomenon we call the *Neuron Polarization Effect*. We organize this section into four conceptual parts. We first state and solve a minimal allocation problem over latent features. Second, we motivate the assumptions of this allocation problem by interpreting scalar neuron readouts as Gaussian information channels in a superposition model. Third, we connect the idealized Rosetta Neuron detectability criterion to our empirical cross-model neuron matching. Finally, we use simulations to verify that the theory reproduces the predicted scaling and polarization trends when paired with our empirical matching procedure.

D.1 A Minimal Capacity-Allocation Model

We begin with a minimal capacity-allocation model that abstracts away architectural and optimization details while capturing the central constraint: a growing set of latent features must share a finite budget of clean neuron capacity.

Feature spectrum. Let N denote the number of MLP neuron coordinates under consideration.[2] Let $A(N)$ be the number of active latent features representing patterns from a data distribution D. A network of size N allocates its neuron capacity across this feature set. We index these features by decreasing importance:

$$r \in \{1, 2, \ldots, A(N)\}.$$

Since modern scaling regimes increase both model size and effective data complexity (Hoffmann et al., 2022), we assume that the feature set does not saturate with model size. Specifically, we take

$$A(N) = \Omega(N), \qquad A(N) > N \tag{9}$$

for sufficiently large N. Thus the model remains in a superposition regime asymptotically (Liu et al., 2025): there are more latent features than neuron coordinates. Following prior spectral scaling theories (Michaud et al., 2023; Bordelon et al., 2020; Bahri et al., 2024), we assume a power-law feature-importance spectrum for these features:

$$w_r = Cr^{-\beta}, \qquad \beta > 1, \qquad C > 0. \tag{10}$$

Here, w_r measures the importance, or downstream value of making feature r cleanly readable, reflecting factors such as frequency, predictiveness, or task relevance.

Feature isolation and Rosetta detectability. Let $s_r \geq 0$ measure how cleanly feature r is isolated in its most dedicated neuron coordinate. Larger values of s_r indicate that this coordinate is dominated by feature r and is more monosemantic, rather than reflecting a mixture of unrelated features. At this stage, s_r is an abstract isolation score; in Appendix D.4, we derive s_r as a signal-to-interference ratio in a linear superposition model.

We assume that independently trained models of similar scale and modality are trained on comparable data distributions D, and therefore share approximately the same coarse latent feature spectrum induced by D. Under this shared-spectrum assumption, we analyze a representative network in isolation: features that are cleanly isolated in one model are expected to be similarly isolated in another matched-scale model, while model-specific interference averages out in cross-model correlations over aligned inputs. We make this connection explicit in Appendix D.6. We therefore model Rosetta detectability by thresholding the isolation score. For a fixed detectability threshold $\tau > 1$, we define the idealized Rosetta count as

$$R_\tau(N) = \sum_{r=1}^{A(N)} \mathbf{1}\{s_r \geq \tau\}. \tag{11}$$

The thresholded features are those cleanly isolated enough to be reproducible and detectable across independently trained models. For $\tau > 1$, this feature count corresponds to distinct dedicated neuron coordinates in the superposition model introduced in Appendix D.4, so we interpret $R_\tau(N)$ as an idealized Rosetta Neuron count.

Utility and budget. We now specify the allocation rule used in our analytical model. We state the allocation objective and constraint in Equation (12), and derive them in Appendix D.5 from a simple superposition picture: neuron coordinates mix many latent features, and each coordinate provides a noisy scalar readout of any one feature. Under this view, the logarithmic utility $w_r \log(1 + s_r)$ comes from the importance-weighted information gain from improving a noisy readout; the linear budget $\sum_{r=1}^{A(N)} s_r \leq \kappa N$, for a constant $\kappa > 0$, comes from the bounded total variance of an N-coordinate activation vector. These ingredients give the following capacity-allocation problem:

$$\max_{s_r \geq 0} \sum_{r=1}^{A(N)} w_r \log(1 + s_r) \qquad \text{subject to} \qquad \sum_{r=1}^{A(N)} s_r \leq \kappa N. \tag{12}$$

[2] Throughout this appendix, we use "neuron" and "coordinate" interchangeably to refer to an individual MLP activation coordinate.

The objective captures a coverage–purity tradeoff. Because the utility is logarithmic, increasing the isolation of a noisy feature initially gives substantial gain, but further purifying an already clean feature gives diminishing returns. The optimum therefore allocates high isolation to the most important features while also extending partial isolation into the lower-importance tail.

D.2 Solving the Allocation Problem

Continuum approximation. To analyze the asymptotic scaling behavior, we approximate the ranked feature spectrum by a continuous variable $r \in [1, A(N)]$. The continuum version of Eq. (12) yields the following variational problem:

$$\max_{s(r)\geq 0} \int_1^{A(N)} Cr^{-\beta} \log(1 + s(r))\, dr \qquad \text{subject to} \qquad \int_1^{A(N)} s(r)\, dr \leq \kappa N. \tag{13}$$

The corresponding effective Rosetta Neuron count is

$$R_\tau(N) = \int_1^{A(N)} \mathbf{1}\{s(r; N) \geq \tau\}\, dr. \tag{14}$$

Proposition 1: Sublinear Rosetta Neuron scaling. Assume $\beta > 1$ and $A(N) = \Omega(N)$, with $A(N) > N$ for sufficiently large N. Then the solution to the allocation problem in Eq. (13) is

$$s^\star(r; N) = \left[\left(\frac{r_0(N)}{r} \right)^\beta - 1 \right]_+, \qquad [u]_+ := \max\{u, 0\}, \tag{15}$$

where $r_0(N) = \Theta(N^{1/\beta})$ is the largest feature rank receiving positive isolation. For every fixed threshold $\tau > 1$, the effective Rosetta Neuron count satisfies

$$R_\tau(N) = \Theta\Big(N^{1/\beta}\Big). \tag{16}$$

Since $\beta > 1$, the Rosetta Neuron count grows sublinearly.

Proof. Introduce a Lagrange multiplier λ for the budget constraint. For any feature rank with positive isolation, $s(r) > 0$, the first-order optimality condition gives

$$\frac{Cr^{-\beta}}{1 + s(r)} = \lambda. \tag{17}$$

Solving for $s(r)$ gives $s(r) = Cr^{-\beta}/\lambda - 1$. Let $r_0(N)$ denote the boundary of the positive-support region, so that $s(r_0(N)) = 0$. Then $\lambda = Cr_0(N)^{-\beta}$, and therefore

$$s^\star(r; N) = \left[\left(\frac{r_0(N)}{r} \right)^\beta - 1 \right]_+. \tag{18}$$

Because $\log(1 + s(r))$ is strictly increasing, the optimum uses the full available isolation budget. Substituting the solution $s^\star(r; N)$ into the budget constraint gives

$$\kappa N = \int_1^{r_0(N)} \left[\left(\frac{r_0(N)}{r} \right)^\beta - 1 \right] dr \tag{19}$$

$$= r_0(N)^\beta \int_1^{r_0(N)} r^{-\beta}\, dr - (r_0(N) - 1) \tag{20}$$

$$= \frac{r_0(N)^\beta}{\beta - 1} - \frac{\beta r_0(N)}{\beta - 1} + 1 = \Theta\big(r_0(N)^\beta\big). \tag{21}$$

Hence $r_0(N) = \Theta(N^{1/\beta})$. Since $\beta > 1$, this cutoff grows sublinearly in N, while $A(N) = \Omega(N)$ grows at least linearly. Therefore $r_0(N) = o(A(N))$, so for sufficiently large N, the positively isolated features form an initial segment $1 \leq r \leq r_0(N)$ within the full feature range $[1, A(N)]$.

Let $r_\tau(N)$ denote the Rosetta frontier, defined by $s^\star(r_\tau(N); N) = \tau$. Using the optimal allocation profile, this frontier satisfies

$$\left(\frac{r_0(N)}{r_\tau(N)}\right)^\beta - 1 = \tau, \tag{22}$$

and therefore

$$r_\tau(N) = r_0(N)(1+\tau)^{-1/\beta} = \Theta(N^{1/\beta}). \tag{23}$$

Thus, counting the features above the Rosetta threshold amounts to counting feature ranks up to order $r_\tau(N)$, so

$$R_\tau(N) = \int_1^{A(N)} \mathbf{1}\{s^\star(r; N) \geq \tau\}\, dr = \Theta(r_\tau(N)) = \Theta(N^{1/\beta}). \tag{24}$$

Since $\beta > 1$, this growth is sublinear. The optimal allocation partitions the spectrum into Rosetta-detectable features with $s_r \geq \tau$, partially isolated features with $0 < s_r < \tau$, strongly superposed features with $s_r = 0$, and features beyond the represented set $A(N)$. The frontiers $r_\tau(N)$ and $r_0(N)$ scale as $\Theta(N^{1/\beta})$, yielding the sublinear Rosetta Neuron count $R_\tau(N) = \Theta(N^{1/\beta})$. This is illustrated in Figure 5.

D.3 Prediction: Neuron Polarization

The same allocation profile predicts a polarization effect. As N grows, the number of detectable Rosetta Neurons increases, and their average isolation also increases. At the same time, a growing tail of latent features remains weakly isolated, corresponding to a more crowded non-Rosetta background.

Rosetta purification. Consider any fixed top-ranked feature $r_\star$ (i.e., a feature with small rank index and high importance). Its isolation in the dedicated neuron coordinate scales as

$$s^\star(r_\star; N) = \left(\frac{r_0(N)}{r_\star}\right)^\beta - 1 = \Theta\left(\frac{N}{r_\star^\beta}\right) - 1 \to \infty \qquad \text{as } N \to \infty. \tag{25}$$

Thus, for any fixed high-importance feature, its isolation grows on the order of N, so its dedicated neuron coordinate becomes increasingly dominated by that feature. Intuitively, this increasing isolation can be interpreted as greater selectivity and monosemanticity.

The Rosetta-detectable features are those with ranks $1 \leq r \leq r_\tau(N)$. We therefore define their average isolation as

$$\bar{s}_{\text{Rosetta}}(N) := \frac{1}{r_\tau(N)} \int_1^{r_\tau(N)} \left[\left(\frac{r_0(N)}{r}\right)^\beta - 1\right] dr \tag{26}$$

$$= \Theta\left(\frac{r_0(N)^\beta}{r_\tau(N)}\right) = \Theta\left(r_0(N)^{\beta-1}\right) = \Theta\left(N^{(\beta-1)/\beta}\right), \tag{27}$$

where we used $r_\tau(N) = r_0(N)(1+\tau)^{-1/\beta}$. Since $\beta > 1$, this average isolation increases with scale, corresponding to a Rosetta population that becomes more selective and monosemantic on average.

Non-Rosetta crowding. We now analyze the isolation trend in the non-Rosetta feature tail. For Rosetta-detectable features, the threshold $\tau > 1$ lets us associate each feature with a distinct dedicated neuron coordinate. Below the threshold, this one-to-one interpretation no longer applies: non-Rosetta features may instead be distributed across multiple shared, superposed coordinates. So, we first analyze crowding on the feature side by measuring the average clean isolation assigned to features outside the Rosetta set:

$$\bar{s}_{\text{tail}}(N) := \frac{1}{A(N) - R_\tau(N)} \int_{r_\tau(N)}^{A(N)} s^\star(r; N)\, dr. \tag{28}$$

Only the partially isolated non-Rosetta region $r \in [r_\tau(N), r_0(N)]$ contributes to the integral, since features beyond $r_0(N)$ have zero isolation. Thus,

$$\int_{r_\tau(N)}^{A(N)} s^\star(r;N)\,dr = \int_{r_\tau(N)}^{r_0(N)} \left[\left(\frac{r_0(N)}{r}\right)^\beta - 1\right] dr \tag{29}$$

$$= r_0(N) \int_{(1+\tau)^{-1/\beta}}^{1} \left(u^{-\beta} - 1\right) du \tag{30}$$

$$= \Theta(r_0(N)) = \Theta(N^{1/\beta}), \tag{31}$$

using the change of variables $u = r/r_0(N)$. Since $A(N) = \Omega(N)$, $R_\tau(N) = \Theta(N^{1/\beta})$, and $\beta > 1$, the measure of the non-Rosetta feature tail, $A(N) - R_\tau(N)$, grows asymptotically faster than $N^{1/\beta}$. Therefore,

$$\bar{s}_{\text{tail}}(N) = O\left(\frac{N^{1/\beta}}{A(N) - R_\tau(N)}\right) = O\left(\frac{N^{1/\beta}}{A(N)}\right) \to 0. \tag{32}$$

Equation (32) shows feature-side crowding: the average clean isolation assigned to an unresolved tail feature vanishes. We can also translate this into a neuron-level interpretation. The $R_\tau(N)$ Rosetta-detectable features correspond to distinct dedicated neurons leaving $N - R_\tau(N) = \Theta(N)$ non-Rosetta coordinates. The total clean isolation assigned to the unresolved tail is $\Theta(N^{1/\beta})$, so the average tail-isolation mass per non-Rosetta neuron scales as

$$O\Big(\frac{N^{1/\beta}}{N}\Big) = O\Big(N^{(1-\beta)/\beta}\Big) \to 0, \tag{33}$$

since $\beta > 1$. Thus, on average, non-Rosetta coordinates receive vanishing clean tail-isolation mass. By definition, $A(N)$ counts the active latent features that the network allocates capacity to represent, so the unresolved tail features are still part of the represented feature population. Since they are not cleanly isolated into dedicated coordinates, they must be represented in a distributed or superposed form. This gives the neuron-level interpretation: the non-Rosetta population forms a crowded, polysemantic background of weakly isolated features.

D.4 Deriving the Isolation Score from Superposition

The minimal model above treats s_r as an abstract isolation score. We now derive this score from a simple linear superposition model of neuron activations.

Linear feature decomposition. Let x denote an aligned input position, such as a text token or image patch, and let $\mathbf{h}(x) \in \mathbb{R}^N$ be the centered vector of MLP neuron activations at that position, with centering taken over the data distribution D. For each latent feature $r \in \{1, \ldots, A(N)\}$, let $\tilde{z}_r(x) \in \mathbb{R}$ be the centered, input-dependent scalar response of feature r. Let $\mathbf{v}_r \in \mathbb{R}^N$ describe how that feature response is distributed across neuron coordinates. We model the activation vector as a linear superposition of feature responses,

$$\mathbf{h}(x) = \sum_{r=1}^{A(N)} \tilde{z}_r(x)\mathbf{v}_r. \tag{34}$$

We normalize the feature directions as $\|\mathbf{v}_r\|_2^2 = 1$, so $\mathbf{v}_r$ specifies the relative distribution of feature r across coordinates, while the overall strength of the feature response on input x is carried by the scale of $\tilde{z}_r(x)$.

For the isolation-score calculation that follows, we first standardize each feature activation over the data distribution. Since $\tilde{z}_r(x)$ is centered, define

$$g_r^2 := \mathbb{E}_{x\sim D}[\tilde{z}_r(x)^2], \qquad z_r(x) := \tilde{z}_r(x)/g_r, \tag{35}$$

for represented features with $g_r > 0$. Then $z_r(x)$ has mean zero and unit variance over D, while g_r carries the overall activation scale of feature r. For tractability, we use a diagonal second-moment approximation, keeping only the variances of the standardized feature activations and neglecting cross-feature covariances:

$$\mathbb{E}_{x\sim D}[z_r(x)] = 0, \qquad \mathbb{E}_{x\sim D}[z_r(x)^2] = 1, \qquad \mathbb{E}_{x\sim D}[z_r(x)z_s(x)] = 0 \quad (r \neq s). \tag{36}$$

The representation can be written as

$$\mathbf{h}(x) = \sum_{r=1}^{A(N)} \tilde{z}_r(x)\mathbf{v}_r = \sum_{r=1}^{A(N)} g_r z_r(x)\mathbf{v}_r. \tag{37}$$

This normalization will let us define the variance contribution of each feature to each neuron coordinate in the next step.

Define the effective loading of feature r on neuron coordinate j by

$$W_{jr} := g_r \mathbf{e}_j^\top \mathbf{v}_r, \qquad a_{jr} := W_{jr}^2. \tag{38}$$

Note that W_{jr} is an effective feature loading onto neuron coordinates in the superposition model, not a learned MLP weight matrix.

Because $\|\mathbf{v}_r\|_2^2 = 1$,

$$\sum_{j=1}^{N} a_{jr} = g_r^2. \tag{39}$$

The activation of coordinate j is

$$h_j(x) = \sum_{r=1}^{A(N)} W_{jr} z_r(x). \tag{40}$$

Using the decorrelation approximation in Eq. (36),

$$\mathrm{Var}_x(h_j) = \sum_{r=1}^{A(N)} W_{jr}^2 = \sum_{r=1}^{A(N)} a_{jr}. \tag{41}$$

Thus a_{jr} is the variance contribution of feature r to neuron coordinate j. We next use this to bound the total variance across coordinates. We assume the normalized activation vector has bounded average variance per coordinate:

$$\sum_{j=1}^{N} \mathrm{Var}_x(h_j) = \sum_{j=1}^{N} \sum_{r=1}^{A(N)} W_{jr}^2 = \sum_{r=1}^{A(N)} g_r^2 \leq C_h N, \tag{42}$$

for a constant C_h independent of N. This says that an N-coordinate normalized activation vector carries $O(N)$ total variance, consistent with standard scale-preserving initialization and normalization schemes (He et al., 2015; Yang et al., 2021; Ba et al., 2016).

Signal-to-interference isolation score. A single MLP neuron coordinate is a scalar readout of a richer, potentially high-dimensional latent feature state. When using coordinate j to read out feature r, we decompose the coordinate into the desired contribution from feature r and a residual term:

$$y_j = W_{jr} z_r + \eta_{jr}, \tag{43}$$

where η_{jr} contains all variation in coordinate j not explained by feature r. We model this residual as having two sources: interference from other modeled features that also contribute to coordinate j, and a fixed finite-resolution noise floor $\sigma_\infty^2 > 0$ that captures unmodeled variation. Under the diagonal second-moment approximation above, the effective residual variance is

$$\nu_{jr} = \sigma_\infty^2 + \sum_{k\neq r} W_{jk}^2. \tag{44}$$

We define the signal-to-interference ratio for reading feature r from coordinate j as

$$s_{jr} := \frac{W_{jr}^2}{\sigma_\infty^2 + \sum_{k\neq r} W_{jk}^2}. \tag{45}$$

The most dedicated neuron coordinate for feature r is the coordinate with maximal signal-to-interference ratio,

$$c(r) = \arg\max_j s_{jr}, \tag{46}$$

and the feature-level isolation score is

$$s_r := s_{c(r),r} = \frac{W_{c(r),r}^2}{\sigma_\infty^2 + \sum_{k \neq r} W_{c(r),k}^2}. \tag{47}$$

Large s_r means that the most dedicated coordinate for feature r is dominated by that feature and more monosemantic, while small s_r means that the feature is read out through more polysemantic neurons. The allocation solution assigns $s_r = 0$ to *tail* features, those with rank beyond $r_0(N)$. This denotes negligible single-neuron isolation, not the absence of being represented. A tail feature can be spread across many neurons with small per-neuron loadings while remaining represented in the full activation vector. For example, suppose each tail feature's variance is spread evenly over all N neurons. Then $W_{c(r),r}^2 = g_r^2/N$ and $s_r \leq g_r^2/(N\sigma_\infty^2)$. Summing over the tail and using Equation (42), the tail's total isolation is at most κ, with κ as in Equation (55), which is only a $1/N$ fraction of the full budget κN. We let $|\mathcal{T}_N|$ denote the number of these tail features. The average isolation per tail feature is therefore at most $\kappa/|\mathcal{T}_N|$, which vanishes as $|\mathcal{T}_N| \to \infty$, so setting $s_r = 0$ for these features is a leading-order idealization.

This definition also explains why the threshold $\tau > 1$ in Eq. (11) gives a one-to-one feature-to-coordinate count in the idealized model. Suppose two distinct features r and r' shared the same threshold-passing most dedicated coordinate j. Since $s_r > 1$,

$$W_{jr}^2 > \sigma_\infty^2 + \sum_{k \neq r} W_{jk}^2 \geq W_{jr'}^2.$$

Similarly, $s_{r'} > 1$ implies $W_{jr'}^2 > W_{jr}^2$, a contradiction. Therefore, each threshold-passing feature has its own dedicated neuron coordinate.

D.5 Deriving the Logarithmic Utility and Linear Isolation Budget

We now derive the logarithmic utility and linear isolation budget used in Eq. (12) based on the signal-to-interference model above.

Logarithmic utility from predictive loss reduction. Recall that z_r has been standardized so that $\mathbb{E}_{x\sim D}[z_r(x)] = 0$ and $\mathrm{Var}_{x\sim D}(z_r(x)) = 1$. For tractability, we use a Gaussian approximation to obtain a closed-form relationship between readout quality and predictive uncertainty. Specifically, we model z_r as Gaussian and treat the residual interference η in the most dedicated coordinate as independent Gaussian noise. After choosing the most dedicated coordinate $c(r)$, its readout is

$$y_{c(r)} = W_{c(r),r} z_r + \eta_{c(r),r}. \tag{48}$$

We divide by the feature coefficient $W_{c(r),r}$ and obtain the normalized observation model

$$\tilde{y}_r := \frac{y_{c(r)}}{W_{c(r),r}} = z_r + \epsilon_r, \qquad z_r \sim \mathcal{N}(0,1), \qquad \epsilon_r \sim \mathcal{N}(0, 1/s_r). \tag{49}$$

For this Gaussian model, the posterior variance is

$$\mathrm{Var}(z_r \mid \tilde{y}_r) = \frac{1}{1+s_r}. \tag{50}$$

Thus the optimal Gaussian negative log-likelihood, or equivalently the conditional entropy, is

$$H(z_r \mid \tilde{y}_r) = \frac{1}{2} \log\left(2\pi e \, \frac{1}{1+s_r}\right). \tag{51}$$

Relative to having no informative readout, the reduction in optimal predictive loss is

$$H(z_r) - H(z_r \mid \tilde{y}_r) = \frac{1}{2}\log(1+s_r). \tag{52}$$

We therefore model a feature with downstream importance w_r contributing utility proportional to $w_r \log(1+s_r)$, with the factor $1/2$ absorbed into the overall scale.[3]

[3]Equivalently, this is the standard scalar Gaussian-channel calculation underlying the Shannon–Hartley theorem: mutual information grows as $\frac{1}{2}\log(1+\mathrm{SNR})$, with s_r playing the role of the SNR.

Linear Isolation Budget. From the definition of the feature-level isolation score in Eq. (47),

$$s_r \le \frac{W_{c(r),r}^2}{\sigma_\infty^2}. \tag{53}$$

Summing over features and using the activation-energy bound in Eq. (42),

$$\sum_{r=1}^{A(N)} s_r \le \sigma_\infty^{-2} \sum_{r=1}^{A(N)} W_{c(r),r}^2 \le \sigma_\infty^{-2} \sum_{j=1}^{N} \sum_{r=1}^{A(N)} W_{jr}^2 \le \frac{C_h}{\sigma_\infty^2} N. \tag{54}$$

Therefore

$$\sum_{r=1}^{A(N)} s_r \le \kappa N, \qquad \kappa := \frac{C_h}{\sigma_\infty^2}. \tag{55}$$

The linear budget thus follows from normalized activation energy and an irreducible finite-resolution floor for scalar neuron readouts.

D.6 Connection to Empirical Rosetta Matching

The theory defines Rosetta-detectable features by thresholding the feature-level isolation score s_r. In the experiments, however, Rosetta Neurons are identified by cross-model activation matching rather than by directly observing s_r. In this section, we relate these two notions by showing that, in an idealized setting, the same isolation score that defines Rosetta detectability also controls the cross-model activation correlation used for empirical matching. A high-dimensional random-packing approximation captures the effect of model-specific superposition: independent models mix unresolved tail features in nearly orthogonal ways, so reliable cross-model neuron matches require sufficiently isolated common features.

Superposition as model-specific interference. Consider two independently trained matched-scale models M and M', trained on comparable data distributions D. As in the allocation model, we assume these models share approximately the same coarse latent feature spectrum induced by D, although their model-specific interference patterns may differ because low-isolation tail features can be packed in many approximately equivalent ways. Let $c_M(r)$ and $c_{M'}(r)$ denote their most dedicated neuron coordinates for feature r, with feature-level isolation scores s_r^M and $s_r^{M'}$.

Since empirical matching uses Pearson correlation, arbitrary rescalings of a neuron activation do not affect the match score. For feature r, the relevant neurons are the most dedicated coordinates $c_M(r)$ and $c_{M'}(r)$, whose raw readouts are

$$y_{c_M(r)}^M = W_{c_M(r),r}^M z_r + \eta_{c_M(r),r}^M, \qquad y_{c_{M'}(r)}^{M'} = W_{c_{M'}(r),r}^{M'} z_r + \eta_{c_{M'}(r),r}^{M'}. \tag{56}$$

We divide each readout by the corresponding feature loading, $W_{c_M(r),r}^M$ or $W_{c_{M'}(r),r}^{M'}$, which removes arbitrary activation scale that does not affect Pearson correlation. This leaves normalized readouts consisting of the shared feature signal plus rescaled model-specific interference:

$$\tilde{y}_{c_M(r)}^M = z_r + \epsilon_M, \qquad \tilde{y}_{c_{M'}(r)}^{M'} = z_r + \epsilon_{M'}. \tag{57}$$

Here z_r is the shared feature signal, while ϵ_M and $\epsilon_{M'}$ are model-specific interference terms. Recall that the shared feature z_r has unit variance, and that the rescaled interference variances are determined by the feature-level isolation scores:

$$\mathrm{Var}(z_r) = 1, \qquad \mathrm{Var}(\epsilon_M) = 1/s_r^M, \qquad \mathrm{Var}(\epsilon_{M'}) = 1/s_r^{M'}.$$

Tail leakage and cross-model covariance. We separate the feature-dependent part of the interference by writing it as a mixture over unresolved tail features:

$$\epsilon_M^{\mathrm{feat}} = \sum_{k \in T_r^M} b_{c_M(r),k}^M z_k, \qquad \epsilon_{M'}^{\mathrm{feat}} = \sum_{k \in T_r^{M'}} b_{c_{M'}(r),k}^{M'} z_k, \tag{58}$$

Here T_r^M and $T_r^{M'}$ denote the non-target features that leak into the scalar readout of feature r from the selected coordinates $c_M(r)$ and $c_{M'}(r)$. The coefficients $b_{c_M(r),k}^M$ and $b_{c_{M'}(r),k}^{M'}$ are the corresponding

normalized leakage coefficients, obtained after rescaling by the target-feature loadings $W^{M}_{c_M(r),r}$ and $W^{M'}_{c_{M'}(r),r}$. This tail-packing calculation focuses on the feature-dependent part of the interference; the residual noise-floor variation is treated as independent across models and therefore does not contribute to cross-model covariance. Since the total normalized interference variances are $1/s_r^M$ and $1/s_r^{M'}$, the feature-leakage coefficients satisfy

$$\sum_{k\in T_r^M}\left(b^M_{c_M(r),k}\right)^2 \le 1/s_r^M, \qquad \sum_{k\in T_r^{M'}}\left(b^{M'}_{c_{M'}(r),k}\right)^2 \le 1/s_r^{M'}. \tag{59}$$

To compute $\mathrm{Cov}(\epsilon_M^{\mathrm{feat}}, \epsilon_{M'}^{\mathrm{feat}})$, we expand the two residual mixtures, which gives a double sum over pairs of leaked features. Under the diagonal feature approximation, $\mathrm{Cov}(z_k, z_\ell) = 0$ for $k \neq \ell$ and $\mathrm{Var}(z_k) = 1$. Therefore, only the same latent feature appearing in both residual mixtures contributes to cross-model covariance:

$$\mathrm{Cov}(\epsilon_M^{\mathrm{feat}}, \epsilon_{M'}^{\mathrm{feat}}) = \sum_{k\in T_r^M\cap T_r^{M'}} b^M_{c_M(r),k} b^{M'}_{c_{M'}(r),k}. \tag{60}$$

Random tail packing. This term is small in a simple random tail-packing approximation. Suppose the tail available to the readout of feature r has effective size L_r, where L_r is the effective number of unresolved non-target features that could contribute interference to this scalar readout. We view each set of leakage coefficients as a vector over this L_r-dimensional tail, with zero entries for features absent from the mixture. If the two models choose tail-mixing directions approximately independently and isotropically in this space, then

$$\mathbb{E}\big[\mathrm{Cov}(\epsilon_M^{\mathrm{feat}}, \epsilon_{M'}^{\mathrm{feat}})\big] = 0, \tag{61}$$

and

$$\mathrm{Var}\big[\mathrm{Cov}(\epsilon_M^{\mathrm{feat}}, \epsilon_{M'}^{\mathrm{feat}})\big] = O\left(\frac{\|b^M_{c_M(r)}\|_2^2 \|b^{M'}_{c_{M'}(r)}\|_2^2}{L_r}\right) \le O\left(\frac{1}{s_r^M s_r^{M'} L_r}\right). \tag{62}$$

Isolation controls matching. Thus, as the unresolved-tail dimension L_r grows, independently packed feature-dependent interference becomes asymptotically uncorrelated across models. Any remaining noise-floor variation is also treated as independent across models. We therefore model ϵ_M and $\epsilon_{M'}$ as uncorrelated across models and uncorrelated with the shared feature signal. Under this approximation, we obtain

$$\mathrm{Corr}(\tilde{y}^M_{c_M(r)}, \tilde{y}^{M'}_{c_{M'}(r)}) = \frac{1}{\sqrt{(1+1/s_r^M)(1+1/s_r^{M'})}}. \tag{63}$$

Cross-model activation correlation is monotonically increasing in the feature-level isolation score of each model. It follows that a high-isolation feature is more likely to produce a high-correlation cross-model neuron score. We next connect this score-level relationship to the notion of Rosetta Neuron matching.

For two models and a fixed detectability threshold $\tau > 1$, the corresponding idealized two-model Rosetta Neuron count is

$$R_\tau^{M,M'}(N) = \sum_{r=1}^{A(N)} \mathbf{1}\{\min(s_r^M, s_r^{M'}) \ge \tau\}. \tag{64}$$

The condition $\tau > 1$ gives the idealized count a one-feature, one-coordinate interpretation: threshold-passing features have distinct dedicated neuron coordinates. This mirrors the empirical matching rule, where we use mutual nearest-neighbor matching with $k = 1$ under Pearson correlation. This enforces one-to-one cross-model neuron matches. By Eq. (63), higher isolation scores imply higher cross-model correlation. So, mutual nearest-neighbor matching can be interpreted as an empirical detectability filter: it prefers to keep features whose shared signal dominates model-specific interference in both models.

D.7 Synthetic Validation of the Analytical Model

We perform a synthetic feature-packing experiment to test whether our phenomenological model produces sublinear Rosetta Neuron scaling and neuron polarization. We embed the theory's predicted isolation profile into synthetic activations for two independent networks. We then recover Rosetta Neurons using the same Pearson-correlation mutual-nearest-neighbor procedure used in our main experiments and compare the recovered scaling and polarization trends to the theory's predictions.

Experimental setup. For each neuron budget N, we instantiate $A = \gamma N$ latent features with importance weights $w_r \propto r^{-\beta}$. We then solve the isolation-allocation problem with budget $\sum_r s_r \leq \kappa N$ to obtain predicted isolation scores s_r. For each synthetic input x, we sample latent feature responses $z_r(x)$ for all features r. We assign every feature with positive isolation, $s_r > 0$, to a synthetic neuron coordinate. Features with $s_r = 0$ are not assigned as target features, but may still appear as background interference. For a neuron j assigned to feature r, we sample a background set $\mathcal{B}_j$ of non-target features and write the activation as

$$h_j(x) = W_{j,r} z_r(x) + \sum_{k \in \mathcal{B}_j} W_{j,k} z_k(x) + \xi_j(x), \tag{65}$$

where $\xi_j(x)$ is an irreducible noise-floor term with variance σ_∞^2. We draw the off-target loadings $W_{j,k}$ for $k \in \mathcal{B}_j$ as a random isotropic Gaussian direction and normalize them to a fixed interference norm. We then choose the target-feature loading $W_{j,r}$ so that the resulting signal-to-interference ratio matches the predicted isolation score:

$$\frac{W_{j,r}^2}{\sum_{k \in \mathcal{B}_j} W_{j,k}^2 + \sigma_\infty^2} \approx s_r. \tag{66}$$

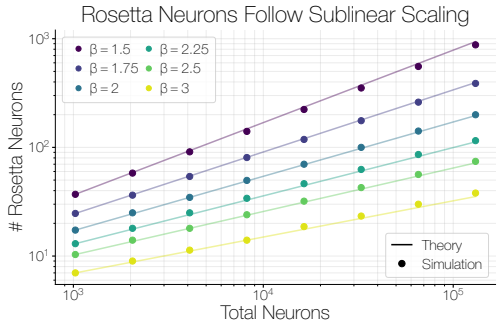
The background mixtures and noise are sampled independently across the two synthetic models, while the latent feature responses $z_r(x)$ are shared.

Evaluation protocol. We run the simulation across neuron budgets $N \in \{2^{10}, \ldots, 2^{17}\}$ and feature-spectrum exponents $\beta \in \{1.5, 1.75, 2, 2.25, 2.5, 3\}$. For each (N, β) setting, we report the mean over 10 independent runs; error bars are omitted because the standard errors are smaller than the plotted markers. For each run, we sample M synthetic inputs by drawing a shared latent feature-response matrix $Z \in \mathbb{R}^{M \times A}$ with i.i.d. Gaussian entries, then standardize each feature column across inputs. The same Z is used for both synthetic models, while the neuron-level feature mixtures and noise are sampled independently across models. We then apply Pearson-correlation mutual-nearest-neighbor matching between the two synthetic models. We compare the recovered Rosetta count against the theoretical scaling prediction $R_\tau(N) = \Theta(N^{1/\beta})$. To test the polarization effect, we separately track the mean isolation of recovered Rosetta neurons and non-Rosetta neurons as N increases.

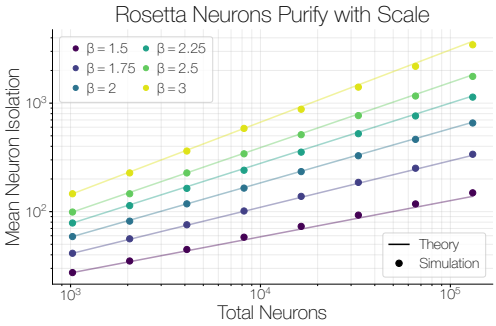
Rosetta Neuron matching recovers theoretical predictions. The simulations closely match the analytical predictions as seen in Figure 27. Across the sweep of feature-spectrum exponents β, the recovered Rosetta Neuron count follows the predicted scaling $R_\tau(N) = \Theta(N^{1/\beta})$: the fitted empirical slopes align with the theoretical exponents $1/\beta$ (Figure 27a). Since the theory predicts the scaling exponent but not the prefactor, we plot theoretical reference curves with slope $1/\beta$ and choose the intercept so that each curve passes through the first empirical point. This agreement suggests that Pearson-correlation mutual-nearest-neighbor matching recovers the theory-predicted Rosetta detectability threshold.

The simulation also recovers the predicted polarization behavior. Our analytical model predicts that the mean isolation of Rosetta neurons grows as $\Theta\big(N^{(\beta-1)/\beta}\big)$, while the average clean isolation associated with the non-Rosetta neuron population decays as $O\big(N^{(1-\beta)/\beta}\big)$. Consistent with these predictions, Rosetta Neuron matches have increasing assigned-feature isolation with slope close to $(\beta-1)/\beta$ (Figure 27b). The unmatched non-Rosetta neurons have decreasing average isolation with slope close to the theoretical $-(\beta-1)/\beta$ (Figure 27c).

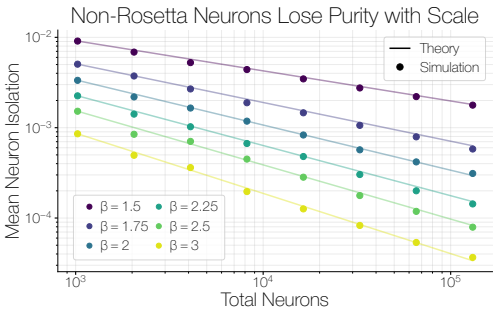
Additional Details. For the reported results, we used $\kappa = 1.0$, $\gamma = 4.0$, $M = 10,000$. We additionally verified that the exponent recovery is robust to these simulation hyperparameters. Across $\kappa \in \{2^{-1}, 2^0, \ldots, 2^5\}$, $\gamma \in \{2^2, 2^3, \ldots, 2^5\}$, and $M \in \{10^3, 10^4, 10^5\}$, all the fitted exponents remained within $\epsilon \approx 0.05$ of the corresponding theoretical predictions.



(a) Rosetta Neuron counts scale as $N^{1/\beta}$.



(b) Rosetta isolation scales as $N^{(\beta-1)/\beta}$.



(c) Non-Rosetta isolation scales as $N^{(1-\beta)/\beta}$.

Figure 27: **Scaling behavior of Rosetta and non-Rosetta neurons in simulation**. Top: the number of Rosetta Neurons follows the predicted scaling law according to our analytical model. Bottom left: Rosetta Neurons become more isolated with scale. Bottom right: Non-Rosetta neurons become less isolated with scale, as predicted by our theory.

E Additional Results on Rosetta Neuron Properties

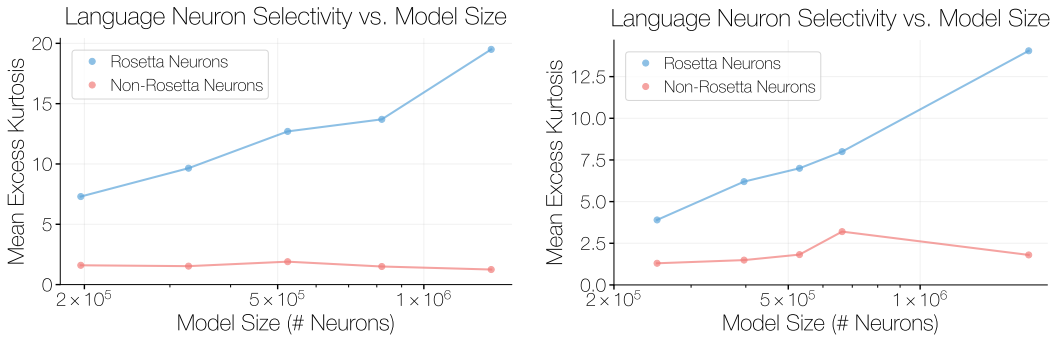
In this section, we provide additional analyses supporting the neuron-level trends described in Section 5. We first extend the language-model selectivity results from Section 5.1 to other model families and present a common proxy across modalities. We then test the functional importance of Rosetta Neurons through a causal intervention and provide further details on the document-type firing analysis from Section 5.2. Finally, we analyze where Rosetta Neurons appear across network depth in both language and vision models.

E.1 Additional Vocabulary-Space Selectivity Results

We measure output-side selectivity using the same vocabulary-space excess-kurtosis metric from Section 5.1. As previously shown in Figure 6, a polarization occurs: the mean excess kurtosis of the Rosetta population increases with scale, while non-Rosetta neurons remain near zero, suggesting weak selectivity. We observe the same qualitative trend for OPT and Qwen-2.5 in Figure 28. Across both families, Rosetta Neurons become increasingly selective with scale, while non-Rosetta neurons remain close to the floor. This supports the interpretation that the Neuron Polarization Effect is not tied to a single model family, but reflects a more general trend across language models.

E.2 A Common Selectivity Measure Across Modalities

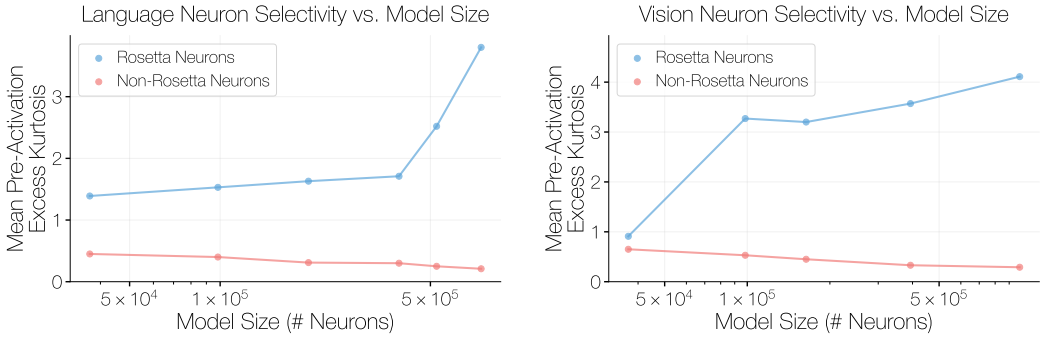
In Section 5.1, we used modality-specific proxies to measure how selective a neuron is. In this section, in order to establish a common selectivity metric, we compute pre-activation excess kurtosis (Gurnee et al., 2024) for Rosetta and non-Rosetta neurons in Pythia and OpenCLIP across model sizes. Gurnee et al. (2024) describe high pre-activation kurtosis as a property "we would expect of monosemantic



(a) Neuron selectivity in OPT.

(b) Neuron selectivity in Qwen-2.5.

Figure 28: **The Neuron Polarization Effect in Language Models.** Rosetta Neurons exhibit increasing mean excess kurtosis of vocabulary-space projections with scale, suggestive of monosemanticity. Non-Rosetta neurons show weaker vocabulary-level selectivity.



(a) Pre-activation kurtosis in Pythia.

(b) Pre-activation kurtosis in OpenCLIP.

Figure 29: **The Neuron Polarization Effect under a common selectivity measure.** We use mean excess kurtosis of neuron pre-activations, an activation-based measure of selectivity applicable to both modalities. In both Pythia and OpenCLIP, Rosetta Neuron kurtosis increases with scale while non-Rosetta kurtosis stays relatively flat near 0 or slightly decreases. This is consistent with the Neuron Polarization Effect and trends observed under the modality-specific proxies in Figure 6.

neurons," which motivates its use here. We follow Section 5.1 and use the Pythia Rosetta set found through the OPT–Pythia matching and the OpenCLIP set from Diffusion–OpenCLIP matching. Figure 29 shows that mean excess kurtosis rises with scale for Rosetta Neurons and stays relatively flat or falls for non-Rosetta neurons in both modalities, consistent with the Neuron Polarization Effect.

E.3 Testing Rosetta Neuron Functional Importance

According to our analytical model, the most important or frequent features are isolated into Rosetta Neurons. To test whether Rosetta Neurons are causally important to model behavior, we ablate them. Specifically, we ablate the 2,672 Rosetta Neurons (out of 36,864 MLP neurons) identified in Pythia-160M from the Pythia–GPT2 matching. We set their activations to zero during the forward pass and measure perplexity on the Pile validation set. As a control, we ablate 2,672 non-Rosetta neurons drawn to match the layerwise distribution of the Rosetta set, and repeat the sampling five times. Despite comprising a small fraction of the population, Rosetta Neurons are disproportionately important to model performance at this scale.

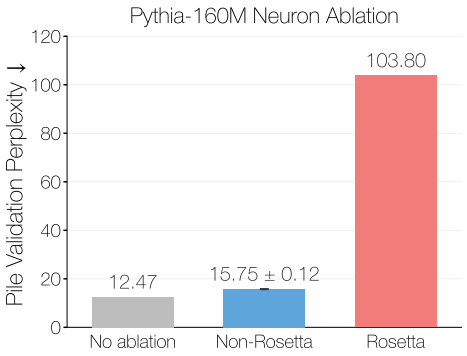


Figure 30: **Ablating Rosetta Neurons sharply degrades perplexity.**

Category	Corpus Share	Pile Subsets
Code	0.1195	GitHub
Math	0.0317	DM Mathematics
Formal/scientific	0.4644	ArXiv, PubMed Central, PubMed Abstracts, FreeLaw, USPTO Backgrounds, NIH ExPorter, StackExchange, PhilPapers
General prose	0.3430	Pile-CC, OpenWebText2, Wikipedia (en), Books3, Gutenberg (PG-19), EuroParl, BookCorpus2
Conversational	0.0415	OpenSubtitles, Ubuntu IRC, HackerNews, Enron Emails, YouTubeSubtitles

Table 5: **Pile source categories used for document-type firing analysis.** We aggregate the 22 Pile subsets into five coarse source categories. The corpus share denotes each category's token frequency in the validation cache and is used to normalize top-activation frequencies.

E.4 Document-Type Firing Analysis

We provide additional details and results for the document-type firing analysis in Section 5.2. For each Rosetta Neuron in a given model, we retrieve its top 20 activating contexts from the validation set of The Pile. Each context is assigned the Pile subset label of its source document (e.g., "GitHub"). We aggregate the original Pile subsets into five general source categories: code, math, formal/scientific text, general prose, and conversational text. The mapping from Pile subsets to categories, together with each category's token share of the validation cache, is shown in Table 5.

For each category c, we compute the fraction of Rosetta Neuron top activations assigned to that category, and then normalize by the category's token frequency in the validation cache:

$$\mathrm{NormalizedFire}(c) = \frac{\#\{\text{top activations from category } c\}/\#\{\text{all top activations}\}}{\#\{\text{tokens from category } c\}/\#\{\text{all tokens}\}}.$$

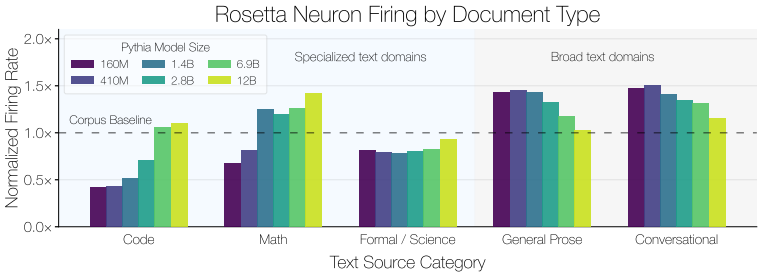
A value of one corresponds to the corpus baseline, meaning that top activations fall in category c at the same rate as expected from its token frequency.

As a control, for each model family and scale, we randomly sample 1000 non-Rosetta neurons from the corresponding model and compute the same normalized document-type firing statistic. We repeat this procedure three times and report the mean and 95% confidence interval across random non-Rosetta neuron samples. This baseline tests whether the observed document-type trends reflect a general property of neurons at a given scale, rather than a property specific to the Rosetta population. We apply this analysis on Pythia and Qwen-2.5 and report the results in Figures 31 and 32. Rosetta Neuron firing becomes increasingly concentrated on specialized categories such as code and math as model size increases for both model families. In contrast, random neurons may exhibit category-specific biases, but they do not show the same systematic shift in firing preference. Since the models within each family are trained on the same data mixture, this shift is unlikely to be explained solely by changing exposure to specialized documents. It suggests that scale may affect which domain-specific features are represented among Rosetta Neurons.

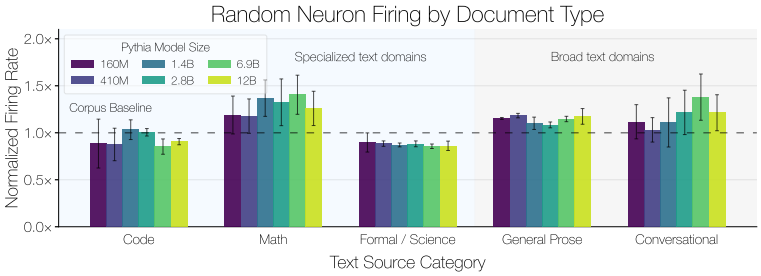
E.5 Depth-Wise Distribution of Rosetta Neurons

We analyze where Rosetta Neurons appear across network depth as models scale. For language models, we use Rosetta Neurons discovered from the Pythia–OPT matching runs. For vision models, we use Rosetta Neurons discovered from the Diffusion–OpenCLIP matching runs. Because models at different scales have different numbers of layers, we compare layers using normalized depth. For each model, we map layer index ℓ to normalized depth $\ell/(L-1)$, where L is the total number of transformer blocks, and divide the interval $[0, 1]$ into 12 equally spaced bins. We then compute the fraction of discovered Rosetta Neurons that fall into each depth bin.

As shown in Figures 33 and 34, Rosetta Neurons are distributed across multiple depth bins in both Pythia and OpenCLIP, rather than being confined to a single depth range across scale. The distribution is not uniform: some models show stronger concentration in early layers, such as Pythia-6.9B in language and OpenCLIP ViT-L/14 (300M) in vision. Overall, however, the depth-wise analysis does not reveal a single consistent depth profile shared across all scales and modalities. We therefore interpret this analysis primarily as a non-degeneracy check: the Rosetta Neuron population is not

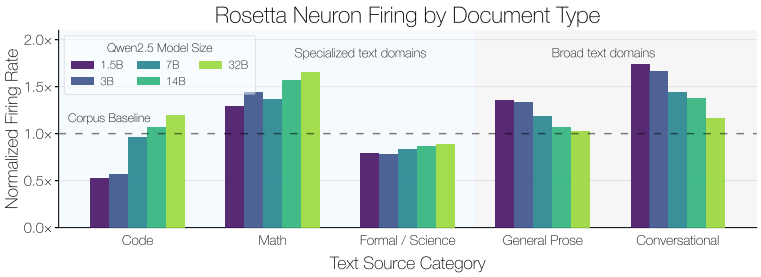


(a) Pythia Rosetta Neurons.

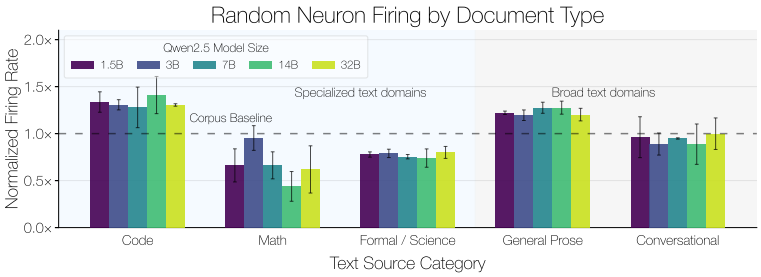


(b) Pythia random non-Rosetta neurons.

Figure 31: **Rosetta Neuron document-type firing in Pythia.** For each Pythia model size, each bar shows how often top-activating contexts fall into a document category, normalized by that category's token frequency in the validation cache. The dashed line marks the corpus baseline. With scale, Rosetta Neuron firing shifts toward specialized categories such as code and math.



(a) Qwen2.5 Rosetta Neurons.



(b) Qwen2.5 random non-Rosetta neurons.

Figure 32: **Document-type firing in Qwen2.5.** We use the same normalized document-type firing statistic as in Figure 31. Rosetta Neurons show an increasing shift toward specialized categories such as code and math with scale. Random non-Rosetta neurons may exhibit category-specific biases, but do not exhibit the same consistent scale-dependent shift toward specialized domains.

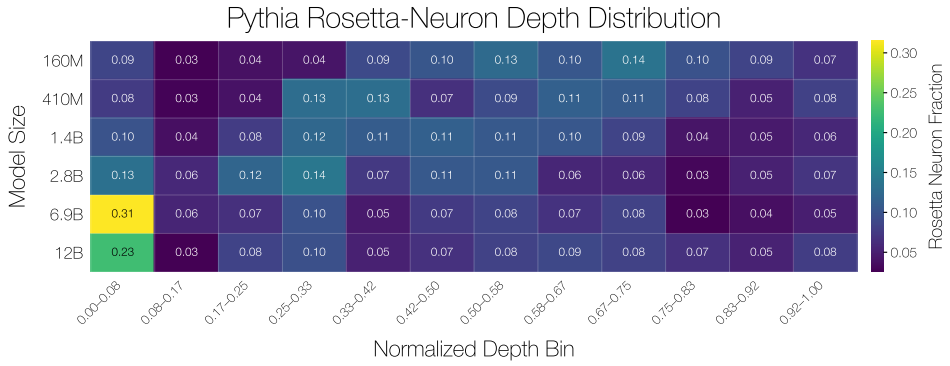


Figure 33: **Depth-wise distribution of Rosetta Neurons in Pythia across scale.** Rosetta Neurons discovered from the Pythia–OPT matching runs.

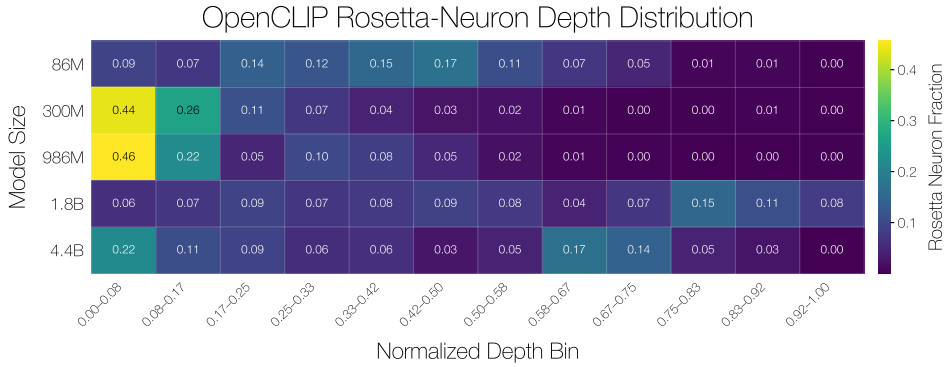


Figure 34: **Depth-wise distribution of Rosetta Neurons in OpenCLIP across scale.** Rosetta Neurons discovered from the Diffusion–OpenCLIP matching runs.

explained solely by matches from one fixed layer region, and the observed scaling trends are not driven by a single depth profile.

F Data Filtering Experimental Details

Data. We use CodeSearchNet (Husain et al., 2019), a function-level code corpus extracted from publicly available GitHub repositories. This dataset spans six programming languages: Python, JavaScript, Java, Go, Ruby, PHP. Each example in the dataset is a single parsed function from a real-world repository. We use JavaScript as a representative target domain. Its training split contains 58,025 functions, corresponding to roughly 16M tokens under the GPT-2 tokenizer, and accounts for approximately 6% of the total tokens in the full multilingual training set. The corresponding JavaScript test split contains 3291 functions and 1.11M tokens.

Filtering. Each filtering method selects functions from the full multilingual CodeSearchNet training set under a matched-token setup. The target budget is set to the size of the JavaScript training split, approximately 16M GPT-2 tokens. We compare four filtering methods: a Rosetta Neuron filter, a non-Rosetta neuron filter, an i.i.d. random sample from the full multilingual training set, and the oracle JavaScript training split. For the neuron-based filters, we score each function by the selected neuron's mean activation over tokens in the function, and select the highest-scoring functions until reaching the matched token budget.

The neurons used for filtering are selected prior to the CodeSearchNet experiment using a different data distribution. We identify candidate JavaScript-selective neurons in Pythia-6.9B using the LLM-based neuron labeling procedure from Appendix A, which infers neuron function from top-activating contexts in The Pile. Among Rosetta Neurons whose top-activating Pile contexts are annotated as JavaScript-related, we select the single neuron with the highest mean normalized activation over

its top-20 Pile contexts. We apply the same rule to non-Rosetta neurons in the same layer whose top activations are annotated as JavaScript-related. This selection is performed using only The Pile activations and LLM annotations before any CodeSearchNet filtering or downstream evaluation. To evaluate domain recovery, we compute F1 between the selected training functions and the oracle JavaScript training split (Table 1).

We present qualitative examples of dataset recovery below. The Rosetta Neuron, Layer 16 Unit 11168 in Pythia-6.9B, retrieves JavaScript functions with high selectivity. In contrast, the non-Rosetta neuron, Layer 16 Unit 12873, shows weaker JavaScript selectivity and fires on a broader range of code, especially web-oriented languages such as PHP.

Training details. To evaluate the downstream utility of the filters, we conduct continued pretraining of GPT2-1.5B using rank-1 LoRA in the attention and MLP layers (Hu et al., 2022). All runs use AdamW (Loshchilov & Hutter, 2019) with peak learning rate 10^{-3} and a 3% linear warmup followed by cosine decay to 0. We train for one epoch at sequence length 1024 using an effective batch size of 32 sequences. All filtering variants use the same model, hyperparameters, token budget, and number of optimizer steps; only the composition of the training data varies. We report perplexity in Table 1 with 95% confidence intervals calculated over three runs.

Comparison to other filtering methods. We additionally compare against other filtering methods beyond those in Table 1 and find that a single Rosetta Neuron matches the strongest baseline with lower inference cost and without target-dataset examples. We follow the procedure outlined above, scoring every code function, ranking them, and selecting functions until reaching a matched 16M-token budget, equal to the size of the JavaScript training split. We compare with two additional baselines (results below): DSIR (Xie et al., 2023) and zero-shot prompt-based labeling with Pythia-6.9B, which is where the Rosetta Neuron was identified. DSIR scores each document using importance weights for hashed n-grams estimated from JavaScript examples relative to the candidate pool; we swept its hyperparameters and report the results from the best configuration. Zero-shot prompting asks Pythia-6.9B to name the programming language of each function, scoring by the log-probability of the JavaScript label token against the five alternatives. All six labels are single tokens, avoiding label-length bias.

Method	**F1** ↑
Chance	0.06
Non-Rosetta Neuron	0.09
DSIR	0.83
Prompting	0.98
Rosetta Neuron	0.98
Oracle	1.00

Table 6: **JavaScript filtering.** Recovery of the ground-truth JavaScript subset as measured by F1 score.

As shown in Table 6, the Rosetta Neuron outperforms DSIR even though DSIR requires JavaScript examples to fit its target distribution, while the Rosetta population is identified by unsupervised cross-model matching without JavaScript examples. It further matches zero-shot prompting, but because it sits at layer 16 of 32, filtering requires no forward pass beyond that layer: 178.8 doc/s against 89.0 doc/s in a matched-batch benchmark, a $2.01\times$ speedup.

Rosetta Neuron-Filtered Function #1: Label=JavaScript

```
function (firstStart) {
    this._startTime = (firstStart) ? Aria._start : new Date();
    if (firstStart) {
        this._logs = [{
            classpath : "Aria",
            msg : "Framework initialization",
            start : Aria._start
        }, {
            classpath : "Aria",
            stop : (new Date()).getTime()
        }];
        this._nbLogs = 2;
    } else {
        this._logs = [];
        this._nbLogs = 0;
    }

    // map function on JsObject prototype
    ariaCoreJsObject.prototype.$logTimestamp = function (msg, classpath) {
        classpath = classpath ? classpath : this.$classpath;
        aria.utils.Profiling.logTimestamp(classpath, msg);
    };
    ariaCoreJsObject.prototype.$startMeasure = function (msg, classpath) {
```

Rosetta Neuron-Filtered Function #1 (continued)

```
        classpath = classpath ? classpath : this.$classpath;
        return aria.utils.Profiling.startMeasure(classpath, msg);
    };
    ariaCoreJsObject.prototype.$stopMeasure = function (id, classpath) {
        classpath = classpath ? classpath : this.$classpath;
        aria.utils.Profiling.stopMeasure(classpath, id);
    };
}
```

Rosetta Neuron-Filtered Function #2: Label=JavaScript

```
function(fn, mOptions) {
    // Functionality taken from lodash open source library and adapted as needed

    mOptions = Object.assign({
        wait: 0,
        leading: true
    }, mOptions);
    mOptions.maxWait = mOptions.wait;
    mOptions.trailing = true;
    mOptions.requestAnimationFrame = false;

    return TableUtils.debounce(fn, mOptions);
}
```

Rosetta Neuron-Filtered Function #3: Label=JavaScript

```
function forEach(f, arr) {
  if (NATIVE_ARRAY_FOREACH) {
    if (arr) {
      NATIVE_ARRAY_FOREACH.call(arr, f);
    }

    return;
  }

  for (var i = 0; i < arr.length; i++) {
    f(arr[i], i, arr);
  }
}
```

Non-Rosetta Neuron-Filtered Function #1: Label=PHP

```
protected static function parseSequence($sequence, &$i = 0)
  {
    $output = array();
    $len = strlen($sequence);
    $i += 1;

    // [foo, bar, ...]
    while ($i < $len) {
      switch ($sequence[$i]) {
        case '[':
          // nested sequence
          $output[] = self::parseSequence($sequence, $i);
          break;
        case '{':
          // nested mapping
          $output[] = self::parseMapping($sequence, $i);
          break;
        case ']':
          return $output;
        case ',':
        case ' ':
          break;
        default:
          $isQuoted = in_array($sequence[$i], array('"', "'"));
          $value = self::parseScalar($sequence, array(',', ']'), array('"', "'"), $i);

          if (!$isQuoted && false !== strpos($value, ': ')) {
            // embedded mapping?
            try {
              $value = self::parseMapping('{'.$value.'}');
            } catch (InvalidArgumentException $e) {
```

Non-Rosetta Neuron-Filtered Function #1 (continued)

```
            // no, it's not
          }
        }

        $output[] = $value;

        --$i;
    }

    ++$i;
  }

  throw new InvalidArgumentException(sprintf('Malformed inline YAML string %s',
  $sequence));
}
```

Non-Rosetta Neuron-Filtered Function #2: Label=PHP

```
private function parsePhpValue($key, $value, array &$result)
    {
        $node =& $result;
        $keyBuffer = '';

        for ($i = 0, $t = strlen($key); $i < $t; $i++) {
            switch ($key[$i]) {
                case '[':
                    if ($keyBuffer) {
                        $this->prepareNode($node, $keyBuffer);
                        $node =& $node[$keyBuffer];
                        $keyBuffer = '';
                    }
                    break;
                case ']':
                    $k = $this->cleanKey($node, $keyBuffer);
                    $this->prepareNode($node, $k);
                    $node =& $node[$k];
                    $keyBuffer = '';
                    break;
                default:
                    $keyBuffer .= $key[$i];
                    break;
            }
        }

        if (isset($node)) {
            $node = $value;
        } else {
            $node = $value;
        }
    }
```

Non-Rosetta Neuron-Filtered Function #3: Label=JavaScript

```
function generateUniqueKey(index, initialKey) {
  var currentCandidate = initialKey;

  var counter = 0;
  while (index[currentCandidate]) {
    var numberAtEndOfKeyMatches = currentCandidate.match(
      NUMBER_AT_END_OF_KEY_REGEX
    );
    if (numberAtEndOfKeyMatches !== null) {
      var nextNumber = parseInt(numberAtEndOfKeyMatches[1], 10) + 1;

      currentCandidate = currentCandidate.replace(
        numberAtEndOfKeyMatches,
        "(" + nextNumber + ")"
      );
    } else {
      currentCandidate += " (1)";
    }

    // This loop should always find something eventually, but because it's a bit dangerous looping endlessly...
    counter++;
    if (counter >= 100000) {
      throw new DeveloperError(
```

Non-Rosetta Neuron-Filtered Function #3 (continued)

```
        "Was not able to find a unique key for " +
          initialKey +
          " after 100000 iterations." +
          " This is probably because the regex for matching keys was somehow unable to work for that key."
      );
    }
  }

  return currentCandidate;
}
```

G Dataset Ablations

G.1 Ablation on the Number of Tokens Used for Language Model Matching

We ablate the number of tokens for matching neurons between language models, taking the Pythia-6.9B–OPT-6.7B model pair as a representative run. Specifically, we sample i.i.d. token sequences from the validation split of The Pile and apply the procedure described in Section 3 to identify Rosetta Neurons. We sweep the total number of tokens used for matching from 10^3 to 10^8. At each data scale, we identify the set of Rosetta Neurons and report the total count in Figure 35 (left). We also compute the overlap between the Rosetta Neuron set identified at each scale and the set identified at the immediately preceding scale, shown in Figure 35 (right). We find that the discovered Rosetta Neuron set becomes increasingly stable as the number of tokens used for matching increases. We use 10M tokens for the main language-model scaling runs, as this provides a practical balance between stability and computational cost.

G.2 Ablation on the Number of Images Used for Vision Model Matching

We ablate the number of images used to match neurons between a diffusion model and a discriminative model. Specifically, we take pMF DiT-B/16 and OpenCLIP ViT-B/16 as a representative pairing and follow the procedure described in Section 3 to identify Rosetta Neurons. We sweep the number of images generated by the diffusion model from 1 to 50,000. At each data scale, we identify the set of Rosetta Neurons and report the total count in Figure 36 (left). We also measure the overlap between the set of Rosetta Neurons identified at each data scale and the set identified at the immediately preceding data scale, shown in Figure 36 (right). We find that the discovered set of Rosetta Neurons stabilizes by 50,000 images: the total count plateaus, and the overlap between successive data scales approaches 1.

G.3 Ablation on the Image Distribution Used for Vision Model Matching

Our vision experiments match neurons between a generative model and a discriminative model, following the GAN-based setup of (Dravid et al., 2023). For modern diffusion-based generators, this requires generated images, since activations from the generative model are only available along the generation trajectory. A natural concern is that this introduces a distribution shift for the discriminative model, which may affect which Rosetta Neurons are identified. As a proxy for this distribution-shift concern, we compare matching under real and generated image distributions using two discriminative models, OpenCLIP ViT-B/16 and DINOv2 ViT-B/14. We run the Rosetta Neurons procedure at data scales from 1 to 50,000 images, once with real images and once with diffusion-generated images. As shown in Figure 37, at larger data scales of 25,000 to 50,000 images, generated images recover approximately 93% of the Rosetta Neuron count found with real images, with an overlap of roughly 0.8. Matching on generated images recovers most of the same correspondences, suggesting that distribution shift has a modest effect.

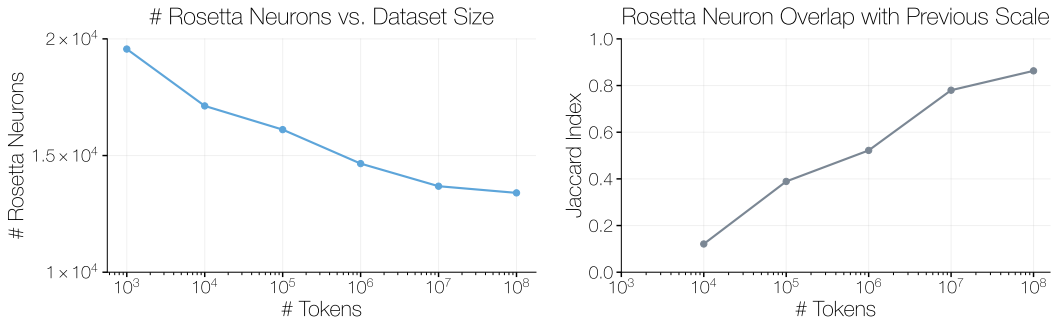


Figure 35: **Language matching stability as a function of dataset size.** We vary the number of tokens used to match neurons between Pythia-6.9B and OPT-6.7B. Left: number of Rosetta Neurons discovered at each data scale. Right: overlap with the Rosetta Neuron set from the previous data scale. The discovered Rosetta Neuron set becomes increasingly stable as the token budget grows.

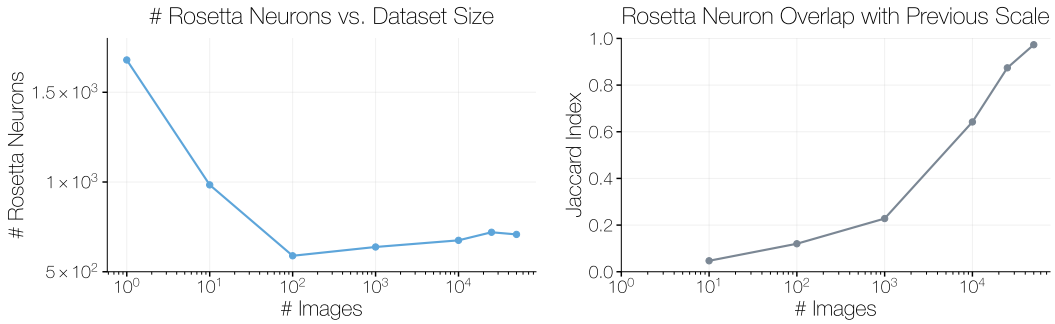


Figure 36: **Diffusion-to-discriminative matching stability as a function of dataset size.** We vary the number of generated images used to match neurons between pMF DiT-B/16 and OpenCLIP ViT-B/16. Left: number of Rosetta Neurons discovered at each data scale. Right: overlap with the Rosetta Neuron set from the previous data scale. Stability improves as the number of images used for neuron matching approaches 50,000.

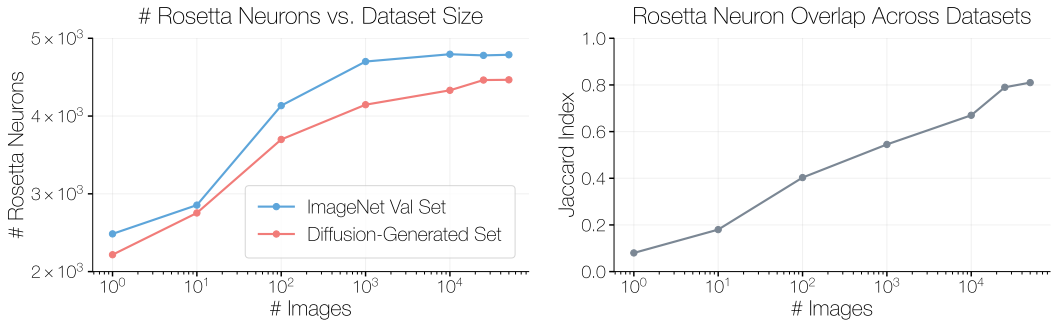


Figure 37: **Effect of image distribution on vision model matching.** We compare Rosetta Neuron matching between OpenCLIP ViT-B/16 and DINOv2 ViT-B/14 using real and diffusion-generated images. Left: number of Rosetta Neurons identified at each data scale. Right: Jaccard index between the Rosetta Neuron sets obtained from the two image distributions. At larger data scales, the two distributions yield similar numbers of matches and substantial overlap.

H Additional Details on VLM-as-a-Judge

This section provides additional details on our VLM-as-a-judge setup for measuring neuron selectivity in vision models. We examine sensitivity to the number of top-activating examples shown to the VLM, validate the approach with a held-out prediction task, and finally show that the same qualitative selectivity trends hold for diffusion models and DINOv2.

H.1 Detailed Experimental Setup

Collecting activations and top-k images for Rosetta neurons. Given a generative–discriminative model pair (e.g., DiT-B/16 and OpenCLIP ViT-B/16), we generate 50,000 images with the generative model and pass them through the discriminative model. We randomly sample 100 Rosetta Neuron pairs and evaluate each neuron in the pair independently. For the discriminative model, we record each sampled neuron's patch-level activations over the 50,000 images. For each neuron and image, we compute a scalar activation score by averaging the neuron's activation over all spatial patches. We rank images by this score and retrieve the top 20 activating images for VLM evaluation. We apply the same procedure to the corresponding generative-model neurons, using activations cached during image generation.

Collecting activations and top-k images for random neurons. For the non-Rosetta neuron baseline, we apply the same procedure independently within each model rather than using matched neuron pairs. In each model, we randomly sample 100 non-Rosetta neurons and record their patch-level activations over the 50,000 images. For each neuron and image, we compute a scalar activation score by averaging the neuron's activation over all spatial patches. We rank images by this score and retrieve the top 20 activating images for VLM evaluation.

VLM judging. For each selected Rosetta Neuron, we form a grid of the top 20 activating images together with the activation heatmaps from one model. The composite image is provided to GPT-5.4 along with a prompt asking it to determine whether the neuron responds to a single coherent visual feature. We provide this prompt below. For each neuron, the VLM returns a binary monosemanticity judgment together with a short natural-language description of the inferred feature or set of features. We follow a similar procedure for the non-Rosetta neuron baseline. We report results for both Rosetta Neurons and the baseline in a model over five independent trials. For the Rosetta Neuron analysis, each trial consists of a disjoint random subset of 100 Rosetta Neurons. For the non-Rosetta baseline, each trial consists of disjoint random subsets of 100 neurons from the non-Rosetta set. We calculate 95% confidence intervals for the percentage identified as monosemantic in each setting.

```
You are evaluating whether a neuron in a vision model is monosemantic (responds to one visual feature) or
polysemantic (responds to multiple unrelated visual features). Below are the top 20 images that most strongly
activate this neuron. Each row shows: the original image, then the activation heatmap from a model and its
heatmap overlay over the image. These maps show where the neuron fires (bright = strong activation). Focus on
the heatmap regions — what visual feature (texture, shape, color, pattern, object part, object category, etc.)
is consistently highlighted across images? Even if the objects in the images differ, the neuron may still be
monosemantic if it consistently responds to the same low-level or mid-level visual feature (e.g., "striped
patterns", "curved edges", "glossy surfaces"). It could be an abstract, high level concept, or it could be
a common visual feature such as texture, which may not be easily nameable but is still a single unifying pattern.

Is there a single visual feature or concept that explains why this neuron fires in all of these
images? Respond in JSON:
{
"is_monosemantic": true/false,
"description": "the shared visual feature if monosemantic, or why not if polysemantic"
}
```

H.2 Sensitivity to the Number of Top-k Activating Images

We ablate the number of top-activating images shown to the VLM judge. Specifically, we repeat the monosemanticity evaluation with $k \in \{2, 5, 10, 15, 20, 50\}$, constructing each composite from the top-k images and their corresponding activation maps and overlays. As shown in Figure 38, the monosemanticity rates decrease as more examples are shown, but the gap between the Rosetta and non-Rosetta neurons remains stable across all values of k. The estimates begin to stabilize around $k = 20$, while using substantially fewer examples provides less evidence for judging whether a neuron is monosemantic.

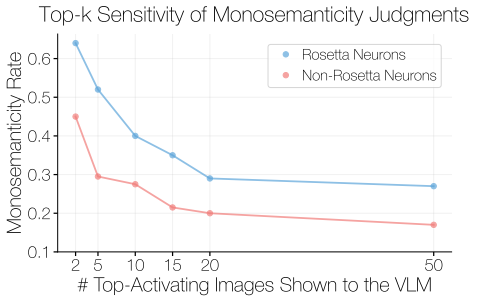


Figure 38: **Effect of # of top-activating images shown to the VLM judge.**

H.3 Validation of VLM-as-a-Judge as a Predictive Metric

Experimental setup. To validate our VLM-as-a-judge setup as a reliable metric, we design a prediction task that tests whether the VLM can recover and generalize the visual features that neurons fire for, rather than relying on spurious cues. We sample 100 random neurons from OpenCLIP ViT-B/16. For each neuron, the VLM is shown the top 20 most-activating images together with their activation maps in a composite grid. We first provide the system prompt described in Appendix H.1 to the model. We then construct a held-out test set consisting of the next 5 highest-activating images and 5 random images drawn from the bottom 50% of activations. These images are shuffled and presented as a labeled grid containing only the raw images. The VLM must predict which images would activate the neuron based only on the training composite. To do this, we provide the prior conversation based on the 20-image training composite in context, and then issue a test prompt asking the model to predict on the 10 test images. We provide this prompt below.

```
IMAGE 2 (TEST): The second composite image is a grid of 10 test images labeled A through J. Given the previous
composite image you saw of top activating images and the corresponding activation maps, predict which of these
ten test images will activate this neuron.
Respond in JSON:
{
"neuron_description": "brief description of what visual feature(s) the neuron responds to",
"predicted_activating": ["A", "C", ...],
"reasoning": "brief explanation of why you chose these images and rejected the others"
}
```

VLM-as-a-Judge is predictive of neuron selectivity. Performance is evaluated over 5 independent trials, each using a disjoint random subset of 100 neurons. We report 95% confidence intervals for accuracy, precision, recall, and F1 in Table 7. VLM-as-a-judge performs meaningfully above a random baseline. The baseline chance is a random predictor that independently marks each test image as activating with probability 0.5. In this 5-positive/5-negative setup, its expected accuracy and recall are 0.5, while its expected precision and F1 are 0.4995 and 0.4865.

Model	Accuracy	Precision	Recall	F1
VLM-as-a-Judge	0.792 ± 0.032	0.865 ± 0.042	0.698 ± 0.038	0.764 ± 0.037
Chance (expected)	0.500	0.500	0.500	0.487

Table 7: **Validation of GPT-5.4 as a neuron selectivity judge.** For each sampled OpenCLIP ViT-B/16 neuron, GPT-5.4 is shown the top 20 activating images with activation maps. It is then asked to identify the activating images in a shuffled set of 10 unseen raw images containing 5 positives and 5 negatives. We report 95% confidence intervals across 5 independent evaluation runs, with each run sampling 100 distinct neurons. We also report the expected metrics for a random predictor.

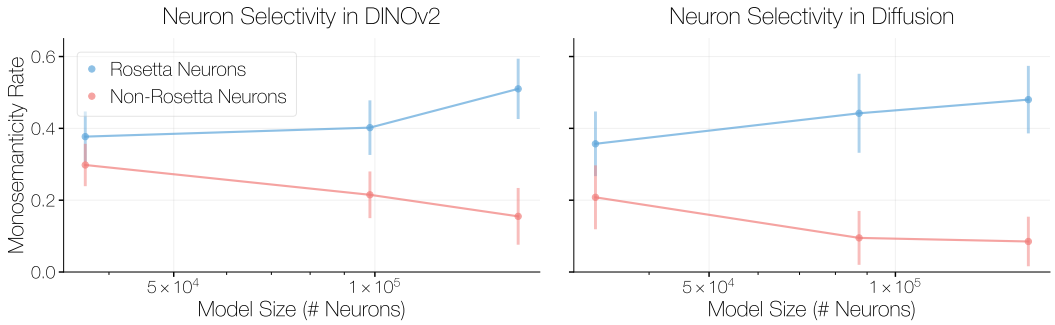


Figure 39: **VLM-judged monosemanticity rate in DINOv2 and diffusion models.** Rosetta Neurons discovered from DINOv2–Diffusion matching runs exhibit increasing monosemanticity with scale, while non-Rosetta neurons become more polysemantic according to the metric. This provides additional evidence for the Neuron Polarization Effect across vision model families.

H.4 Results for Neuron Selectivity in Other Vision Models

In Section 5.1, we showed evidence for the Neuron Polarization Effect in OpenCLIP: Rosetta Neurons become more selective with scale, while the non-Rosetta population remains comparatively less selective. In this section, we apply the same VLM-as-a-judge setup to measure neuron monosemanticity in DINOv2 and diffusion models as they scale. We use the Rosetta Neurons discovered from the DINOv2–Diffusion matching runs. In Figure 39, we plot the fraction of neurons judged monosemantic as a function of model size. The monosemanticity fraction increases with model scale for Rosetta Neurons and decreases for non-Rosetta neurons in both families, consistent with the trend observed in OpenCLIP and language models. The monosemanticity rates for Rosetta Neurons in DINOv2 and diffusion are similar, as these neurons are identified from the same cross-family matching runs. In contrast, the non-Rosetta populations exhibit model-specific levels of polysemanticity.

I Further Discussion and Limitations

In this section, we further contextualize the Rosetta Neuron scaling picture. We begin with a DINOv3 failure case, which helps clarify the conditions under which Rosetta Neuron scaling arises. We then discuss the challenge of operationalizing monosemanticity and polysemanticity, and relate our selectivity measures to these concepts.

I.1 DINOv3 as an Informative Failure Case

A notable exception to the scaling behavior observed in our main experiments from Section 4 is DINOv3 (Siméoni et al., 2025), which does not exhibit a clear Rosetta Neuron scaling law. We repeat the discriminative-to-diffusion matching procedure from Section 4 with DINOv3, and report the resulting Rosetta Neuron counts alongside DINOv2 results in Figure 40. In contrast to DINOv2, which follows the trend observed in other vision model families, DINOv3 Rosetta Neuron counts do not follow a monotonic scaling trend. This deviation is consistent with the fact that DINOv3 modifies the DINOv2 training setup with additional constraints on intermediate representations, encouraging them to match statistics from earlier in training. We hypothesize that these constraints alter how neuron-level features are organized. More broadly, the absence of a clean trend in DINOv3 suggests that Rosetta Neuron scaling depends on the interaction between data, architecture, optimization, and training objective.

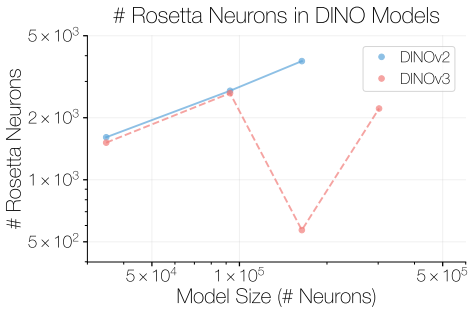


Figure 40: **DINOv3 does not exhibit a clear Rosetta Neuron scaling law.**

I.2 Operationalizing Monosemanticity and Polysemanticity

To interpret neuron selectivity, we need a working definition of monosemanticity and polysemanticity (Bricken et al., 2023b). These are general semantic descriptions of representations: a unit is considered monosemantic if it responds to one coherent feature, and polysemantic if it responds to multiple unrelated features. These terms are related to, but distinct from the more formal notion of superposition (Elhage et al., 2022), which describes feature interference in activation space. Since the underlying feature directions are not directly available in real models, prior work has relied on proxies such as top-activating examples, output-space projections (Geva et al., 2021), activation maps (Bau et al., 2017), and VLM-based evaluations (Shaham et al., 2024). These judgments therefore depend on the input distribution, the evaluation procedure, and the level of semantic granularity. For example, a unit that responds to both cats and dogs may be viewed as monosemantic at the level of mammals, but polysemantic at the level of animal species.

Given this ambiguity, we do not treat monosemanticity and polysemanticity as fixed formal definitions. Our goal is instead to understand population-level trends for selectivity as models scale. To do this, we use proxies such as vocabulary-space excess kurtosis for language models and VLM judgments for vision models. Additionally, qualitative visualizations of top-activating examples and activation maps show patterns consistent with our measurements. This evaluation does not settle the general problem of defining monosemanticity in real models, but it is well suited to the relative prediction studied here: Rosetta Neurons should become increasingly selective with scale compared with the non-Rosetta background. Developing more principled operational definitions remains an important direction for future work.

J Compute Resources

All experiments were run on NVIDIA A100 80GB GPUs. Neuron matching experiments and related ablations used 8 GPUs per model pair. Matching between two models in the $\sim$100M parameter family takes around 30 minutes, while matching models in the $\sim$30B parameter family takes about 24 hours. This corresponds to roughly 4–192 A100 GPU-hours per model-pair matching run depending on the model scale. The continued pretraining of GPT2-1.5B takes around 30 minutes on 8 GPUs, corresponding to roughly 4 A100 GPU-hours per run. The rest of the experiments were run using a single GPU. We did not train any of the pretrained language or vision models used in the scaling-law experiments. These experiments use existing checkpoints.