Decode-gLM: Tools to Interpret, Audit, and Steer Genomic Language Models

Aaron Maiwald*[1], Piotr Jedryszek[2], Florent Draye[3], Garrett M. Morris[1], and Oliver M. Crook†[4]

[1]Department of Statistics, University of Oxford

[2]Department of Biology, University of Oxford

[3]ELLIS Institute Tübingen, Max Planck Institute for Intelligent Systems

[4]Department of Chemistry & Kavli Institute for Nanoscience Discovery, University of Oxford

Abstract

While genomic language models are enabling the *de novo* design of entire genomes, they remain challenging to interpret, limiting their trustworthiness. Here, we show that sparse autoencoders (SAEs) trained on Nucleotide Transformer activations decompose hidden representations into interpretable biological features without supervision. Across layers and model sizes, SAEs identified over 100 diverse functional annotations encoded in the model's activations. This included viral regulatory elements such as the CMV enhancer, despite viral genomes being excluded from training data. Tracing this signal revealed contamination in reference databases, demonstrating that interpretability methods can audit training data and identify hidden data leakage. We then show that meta-SAEs, trained on the decoder weights of another SAE, can identify conceptual hierarchies encoded in the model, including a more abstract feature related to multiple HIV annotations. We confirmed that the features identified by our SAEs were learned during pretraining through probing a randomly initialised model. Finally, we demonstrate that our SAEs allow us to steer model predictions in biologically meaningful ways, showing that we can use an antibiotic-resistance SAE-feature to steer the model toward the A1408G aminoglycoside-resistance mutation in the ribosomal gene 16s rRNA. Together, these results establish SAEs as a method for both discovery and auditing, providing a toolkit for interpretable and trustworthy genomic foundation models. Readers can explore our findings at https://interpretglm.netlify.app/.

1 Introduction

For the first time, we can intelligently compose entire, functional bacteriophage genomes using artificial intelligence (King et al., 2025). This achievement is the culmination of rapid advances in improving genomic language models (Nguyen et al., 2023; Benegas et al., 2023; Brixi et al.,

*First author: aaron.maiwald@st-hildas.ox.ac.uk

†Senior author: oliver.crook@chem.ox.ac.uk

2025; Dalla-Torre et al., 2025), training them to predict masked nucleotides across thousands of genomes. This progress mirrors breakthroughs using protein language models (Ferruz et al., 2022; Lin et al., 2023; Nijkamp et al., 2023), capable of designing novel, functional fluorescent proteins (Hayes et al., 2025) or lysozymes (Madani et al., 2023).

During pretraining, genomic language models learn to encode high-level genomic functions in their embeddings (Nguyen et al., 2023; Benegas et al., 2023; Brixi et al., 2025; Dalla-Torre et al., 2025). Given that a large fraction of even the human genome remains uncharacterised, genomic language models may encode functional information beyond our current scientific understanding. Decoding what genomic language models have learned could then be a new route to scientific insight in genomics. One approach to understanding neural network representations is *probing*—training classifiers to predict specific input properties from model representations (Alain & Bengio, 2018; Immer et al., 2022). High predictive performance of such classifiers is taken to suggest that the model encodes the corresponding concept. While effective for testing whether representations encode known concepts, probing requires labelled data. For uncharacterised genomic regions, such functional labels are, by definition, absent, making supervised methods less useful.

Sparse autoencoders (SAEs) offer an unsupervised alternative for extracting interpretable features from model representations (Yun et al., 2023; Bricken et al., 2023; Cunningham et al., 2023). Unlike standard autoencoders that compress inputs, SAEs project them into higher-dimensional spaces to overcome *polysemanticity*—the fact that, in neural networks, individual neurons usually activate on multiple, unrelated input features. Elhage et al. (2022) hypothesise that polysemanticity allows the model to encode more features than it has neurons, by distributing the representation of features. By expanding the representational space, SAEs disentangle these features, allowing individual dimensions of that space (*SAE-latents*) to capture individual input features more faithfully than the original neurons, thereby increasing interpretability (see Related Work A for details).

This approach has been highly productive in the domain of protein language models. Simon & Zou (2025) and Gujral et al. (2025) apply this technique to ESM-2 uncovering thousands of encoded features ranging from entire protein families like the PTH family to biophysical properties like zinc finger domains. Comparable analyses of genomic language models remain limited. Contemporaneous with this work, Brixi et al. (2025) applied sparse autoencoders to the genomic model EVO-2, identifying SAE-latents moderately to strongly associated with frameshifts, coding sequences, introns, and exon starts and exon ends, among others.

We make four main contributions. First, we demonstrate that SAEs can discover an extensive vocabulary of biological concepts from the activations of the Nucleotide Transformer (Dalla-Torre et al., 2025), far larger and more granular than previously shown for genomic language models. In particular, SAEs identify over 100 biological concepts with F1-scores above 20%. Second, we show that this includes unexpected viral elements that we trace back to training data contamination. Third, we show that the meta-SAEs, trained to sparsely encode and reconstruct SAE decoder weights, can identify a more abstract concept related to HIV. Fourth, we verify the causal role of these features within the Nucleotide Transformer through causal interventions. In particular, we demonstrate that some SAE-features related to antibiotic-resistance enable biologically meaningful

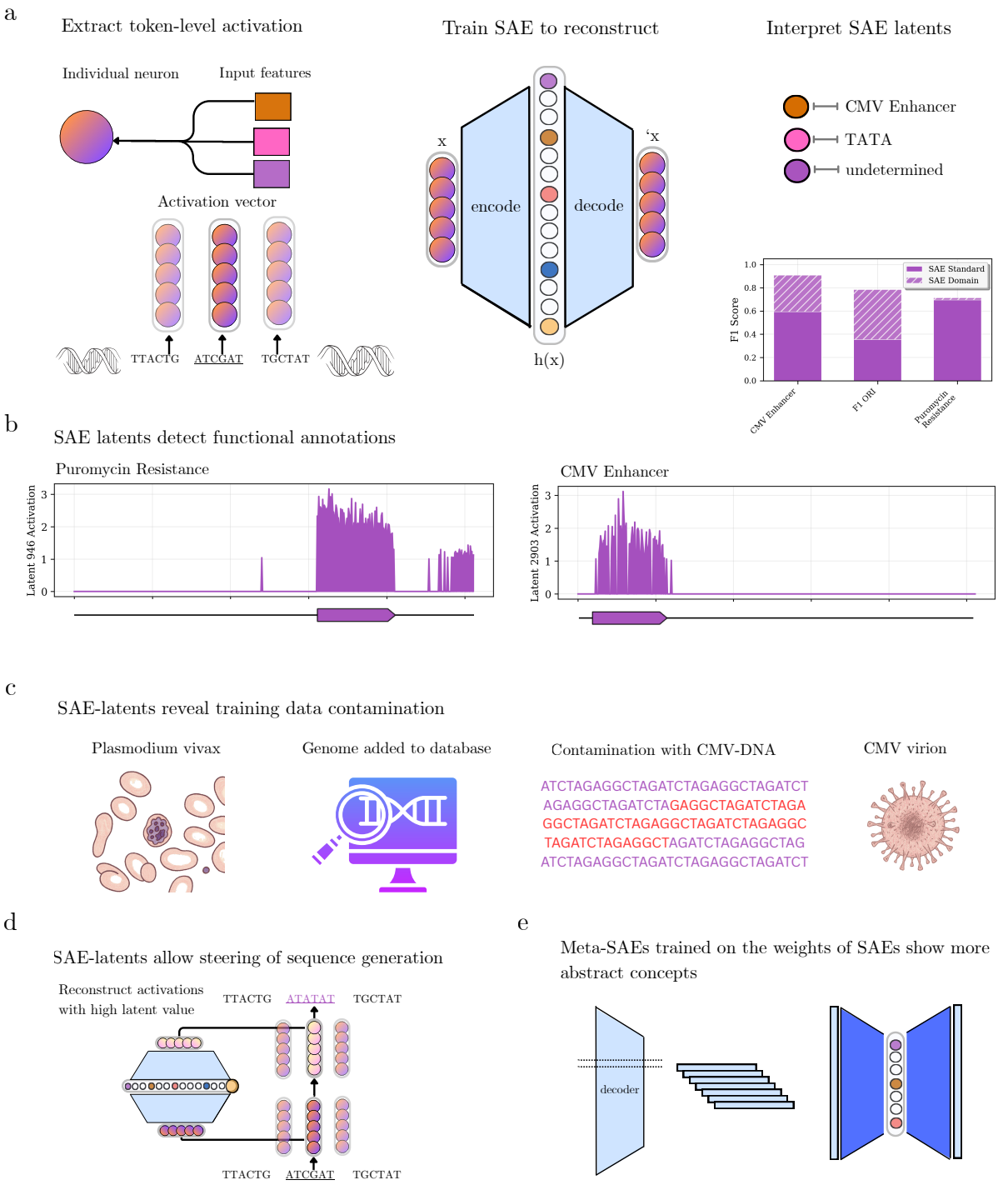

Figure 1: **Interpreting Genomic Language Models using Sparse Autoencoders.** a) Left: Individual neurons tend to be hard to interpret as they activate on multiple unrelated input features. Middle: Sparse autoencoders (SAEs) can project activation vectors into a higher-dimensional, sparse space, in which individual dimensions ("latents") associate more closely with single input features. Right: In some cases, these latents correspond to known biological concepts, though not always. b) Two examples of latents activating selectively on puromycin resistance genes (left) and CMV enhancers (right). c) The model has learned to encode CMV enhancers, despite viral genomes being intentionally excluded from the training data. We trace this back to contamination of the training data with viral DNA. d) We show that our SAEs can be used to steer model generation by intervening on model activations. e) Meta-SAEs are trained on the decoder weights of other SAEs and sometimes decompose them into more abstract features.

forms of steering, inducing the model to shift probability toward the A1408G aminoglycoside-resistance mutation in the ribosomal gene 16s rRNA. Finally, we provide a web interface for readers to explore the features we discovered at https://interpretglm.netlify.app/.

2 Results & Discussion

We analysed the Nucleotide Transformer (Dalla-Torre et al., 2025), a bidirectional encoder model pretrained on 850 genomes spanning multiple kingdoms of life. The model achieves strong performance across a range of benchmarks, indicating that it has captured extensive genomic structure. Prior work showed that its embeddings cluster into a small number of broad functional categories, including introns and coding sequences, but given the diversity of its training data, this likely represents only a fraction of the biological concepts it encodes.

Sparse autoencoders (SAEs) provide a method for disentangling such representations. Trained under a sparsity constraint so that only a small subset of neurons activate for any given input, SAEs typically project inputs into a higher-dimensional space, allowing features distributed across multiple neurons to be separated into components associated with a single input feature. In the natural language domain, this approach identified SAE-latents activating on features ranging from sycophancy to the Golden Gate Bridge (Templeton et al., 2024). Here, we adapt this approach to genomic language models.

To train SAEs, we extracted activations from the Nucleotide Transformer using plasmid sequences from AddGene (Kamens, 2015). Plasmids are absent from Nucleotide Transformer's pre-training data, creating a distributional shift, and can be densely annotated with functional elements that are useful to characterise SAE-latents. Following previous work, we chose to extract multi-layer perceptron (MLP) outputs. For a single input of n tokens, these consist of n vectors of dimension d_{mlp}, where d_{mlp} is the dimensionality of the MLP outputs. To monitor how features change throughout the model, we extracted activations across transformer blocks. We then trained SAEs to reconstruct MLP outputs while maintaining sparse activation, thereby learning an overcomplete dictionary of latent features. After hyperparameter tuning, we selected the SAEs with the best combination of reconstruction and sparsity loss.

To interpret the SAE-latents, we examined their most strongly activating tokens on two non-overlapping test sets (1,000 plasmids each) and associated them with functional annotations from pLannotate (McGuffie & Barrick, 2021). We then quantified the strength of the association between SAE-latents and annotations using F1-scores. For more details, see Methods B.

2.1 Discovering Unexpected Concepts using Sparse Autoencoders

To efficiently evaluate thousands of SAE latents for their association with annotations, we used the following heuristic. Our pipeline examined the 20 most activating tokens for each SAE latent and flagged any latent where at least half of the most activating tokens share an annotation. For each flagged latent, we then compute the F1-score of the latent-annotation pair, selecting an activation threshold on the validation set that maximises F1-score and then measuring F1-scores

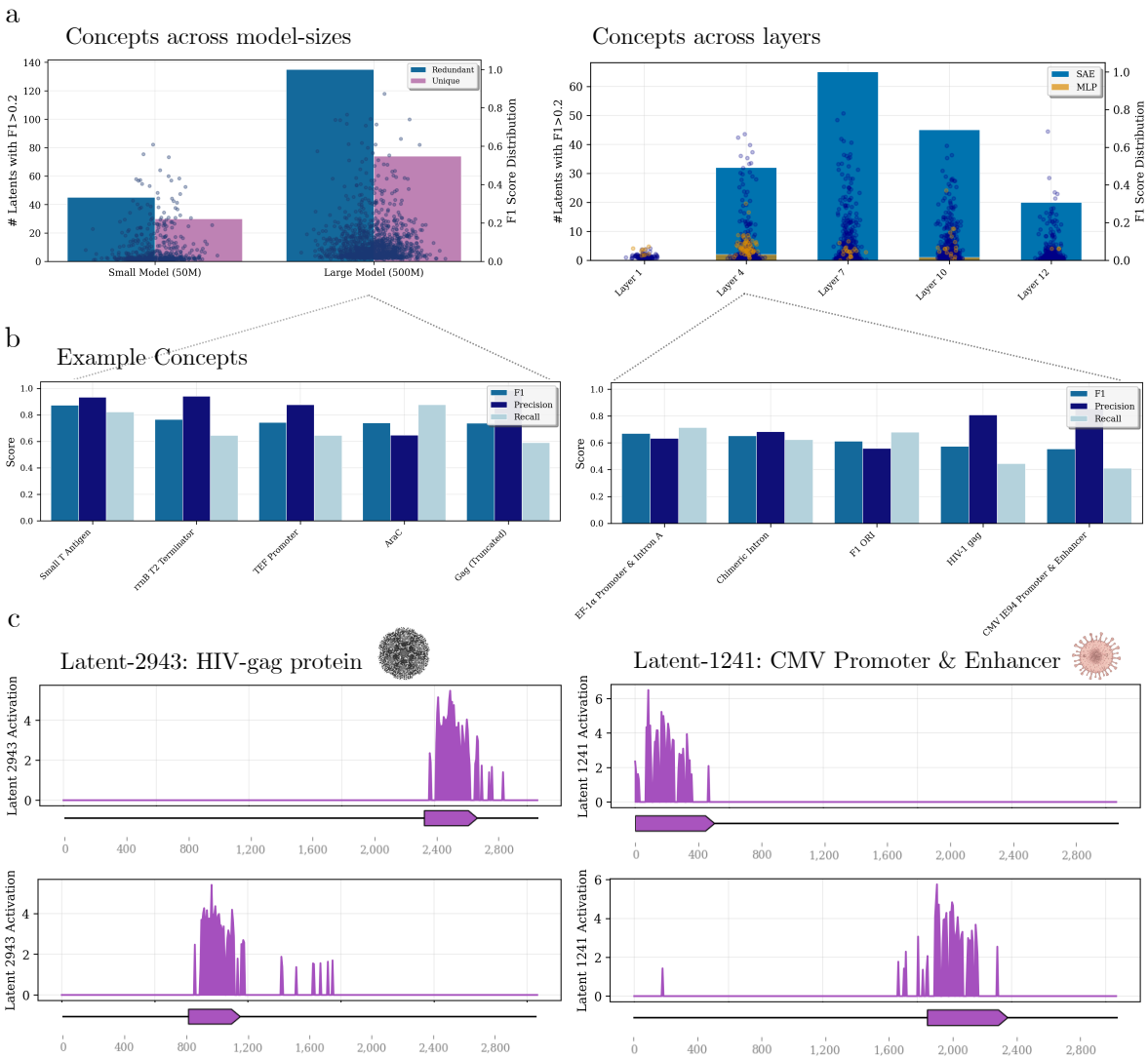

Figure 2: **Sparse Autoencoders extract a large set of interpretable biological concepts across layers and model sizes. a)** Left: we compare the number of interpretable SAE-latents across two model sizes (50m vs 500m) at the same model location (layer 10), as well as the distribution of F1-scores across latents. Right: shows how the number of interpretable latents and the distribution of F1-scores change across layers of the small model. As a baseline, we compare to the number of interpretable MLP-latents (yellow). **b)** Shows the five SAE-latents with the highest F1-scores in the large Nucleotide Transformer (left) and the fourth layer of the small model (right). **c)** Shows examples of two SAE-latents firing selectively on genes coding for the HIV-gag protein (left) and on CMV Promoters and Enhancers (right).

on separate test sets.

Based on previous studies, we expect feature representations to change across layers. We therefore train SAEs across the first, fourth, seventh, tenth and twelfth (final) MLP-layer[1] of the smallest Nucleotide Transformer (50m). We call an SAE-latent interpretable if it has an F1-score of at least 20% for some annotation. We find that the number of interpretable SAE-latents varies

[1]Specifically, we train on the MLP outputs.

strongly across layers, with none in the first layer and peaking in the 7th layer with over 60 (see Fig. 2b). This is consistent with previous findings that middle-to-late layers encode the most extractable features (Simon & Zou, 2025). Across layers, the number of interpretable SAE-latents far outnumbers the number of interpretable MLP-latents, and the most interpretable SAE-latents have consistently far higher F1-scores (see Fig. 2a). This confirms that our SAEs increase interpretability compared to the base model MLP.

We expected a larger Nucleotide Transformer to encode more input features. We therefore trained SAEs on the 10th layer of the 500m Nucleotide Transformer, using the same hyperparameters as before. As expected, we find that our SAEs extract far more interpretable latents from the larger model. To save computational cost, we only train an SAE on layer 10 of this larger model. We find 135 unique latent-concept pairs, versus at most 65 in any layer of the smaller model. Secondly, the most interpretable SAE-latents have a much higher F1-score for the large model than for the smaller model: 87.3 % (for the small T-antigen) versus 77.9 % (for Puromycin Resistance).

Unexpectedly, some of the SAE-latent with the highest F1 score were associated with viral annotations—CMV enhancer (F1: 59%) and HIV-gag with 57.3% F1-score. The CMV enhancer, ubiquitous in mammalian expression vectors, drives strong gene expression for research and gene therapy applications. HIV Gag proteins are structural polyproteins that assemble to form the core components of new HIV virus particles, playing a central role in viral assembly, budding, and maturation. These features appear despite Dalla-Torre et al. (2025) explicitly excluding viral genomes from training. To confirm that these features are encoded via pretraining, we want to test how well they can be extracted from the trained Nucleotide Transformer compared to a randomly initialised model of the same architecture. To do this, we train logistic regressions on the Nucleotide Transformers MLP outputs ("linear probes") for both conditions. For each concept shown in 2b, we find that across 30 probes in each condition, the average F1-score is far greater for the pretrained model. The smallest increase in F1-score is an absolute 17.2 %, and the average increase is over 50 % (see Supp. Figure 5 for Details.) This confirmed that these features are encoded via pretraining.

We consider two explanations for how the Nucleotide Transformer could have learned to encode CMV features during pretraining despite viral exclusion: (1) the model has somehow extrapolated from non-CMV related DNA, (2) database contamination introduced CMV DNA into supposedly virus-free reference genomes. We next examine the second explanation.

2.2 Explaining the Source of Viral Features

To determine whether CMV enhancer sequences appeared in the training data despite viral genome exclusion, we used BLAST to query all CMV enhancer sequences from our plasmid dataset against the model's training corpus. We identified five entries with near-perfect matches to CMV enhancer (Table 1, each with E-values indicating extreme statistical significance $\leq 2.1 \times 10^{-15}$). The matching organisms—bacteria and one protozoan—cannot be naturally infected by CMV, which exclusively targets mammalian cells. This makes viral integration unlikely and points to database contamination as the source. Such contamination occurs through multiple pathways:

during sample collection (from infected organisms), laboratory preparation (via shared equipment or cross-contamination), or sequence assembly. This problem affects approximately 0.54% of GenBank entries (Steinegger & Salzberg, 2020), suggesting a 98.9% probability that at least one of the 850 training genomes contains contamination.

Organism	**Type**	**Best Match**			**Total Matching (bp)**
		Match Length (bp)	**Identity (%)**	**E-value**	
Plasmodium vivax	Protozoan	396	100	$< 1 \times 10^{-180}$	2,232
Rhodoblastus acidophilus	Bacterium	396	99.24	$< 1 \times 10^{-180}$	866
Rhabdonatronobacter sediminivivens	Bacterium	53	100	3.82×10^{-17}	53
Dankookia rubra	Bacterium	50	100	1.55×10^{-15}	50
Mariniblastus fucicola	Bacterium	50	100	2.14×10^{-15}	50

Table 1: **CMV Enhancer Matches in Training Data.** BLAST Match Statistics from querying all CMV enhancer sequences in our testset against the Nucleotide Transformer's training dataset, finding five organisms with significant matches.

Our discovery of training data contamination demonstrates three critical points. First, interpretability tools such as SAEs can serve as auditing mechanisms, allowing researchers to identify and remove hidden sequences that may compromise model behaviour and downstream analyses. Second, this finding highlights a biosecurity concern: even when viral genomes are explicitly excluded, viral information can leak back into training corpora through contaminated reference sequences, undermining safeguards that rely on dataset curation alone. Third, contamination complicates scientific claims about generalisation, since models may appear to succeed on supposedly novel test organisms that were, in fact, inadvertently represented during training. Together, these points emphasise both the promise of interpretability for ensuring data integrity and the need for more rigorous standards in constructing biological training datasets.

The identified CMV enhancer contamination represents only 500 of 29 billion training tokens—insufficient to fully account for the model's strong CMV enhancer representation (78% F1-score). The remaining signal likely stems from either additional undetected contamination or transfer learning from similar mammalian enhancer sequences. To better understand the nature of this learned feature, we characterised the sequence properties that the CMV enhancer probe relies on. To do this, we ablated known regulatory motifs and compared the decrease in probe probability to random ablation. Our analysis revealed that the probe's performance does not primarily rely on known regulatory motifs (see Supp. Material C). To test if the probe instead relied on local k-mer statistics, we used u-shuffle (Jiang et al., 2008) to create decoy sequences, preserving k-mer frequencies while disrupting their order. We found that preserving 5-mer frequencies was sufficient to recover 85% of the original probe probability, suggesting that the model represents the enhancer through k-mer sequence patterns rather than biologically known motifs (see Supplementary Material C for the full ablation and decoy analysis).

2.3 Examining Conceptual Hierarchies with Meta-SAEs

Building on SAEs, Meta-SAEs (Leask et al., 2025) are trained on the decoder-column weights of another SAE, which are intended to encode input-features. The meta-SAE encoder, thereby, is designed to decompose features into more abstract, constituent features. To apply Meta-SAEs

to our single-annotation features, we train Meta-SAEs on our SAE for the large Nucleotide Transformer (500m), training exclusively on decoder-columns of SAE-latents with positive F1 scores, sweeping across latent dimensionalities and sparsity coefficients. After comparing multiple methods, we find that JumpReLU ($d_{meta-sae} = 128$) achieves the best trade-off between L0 sparsity (3-6) and explained variance (30–40%). See MethodsB for details. To interpret the Meta-SAE latents, we inspect the top-20 most activating "child-latents" for each meta-latent.

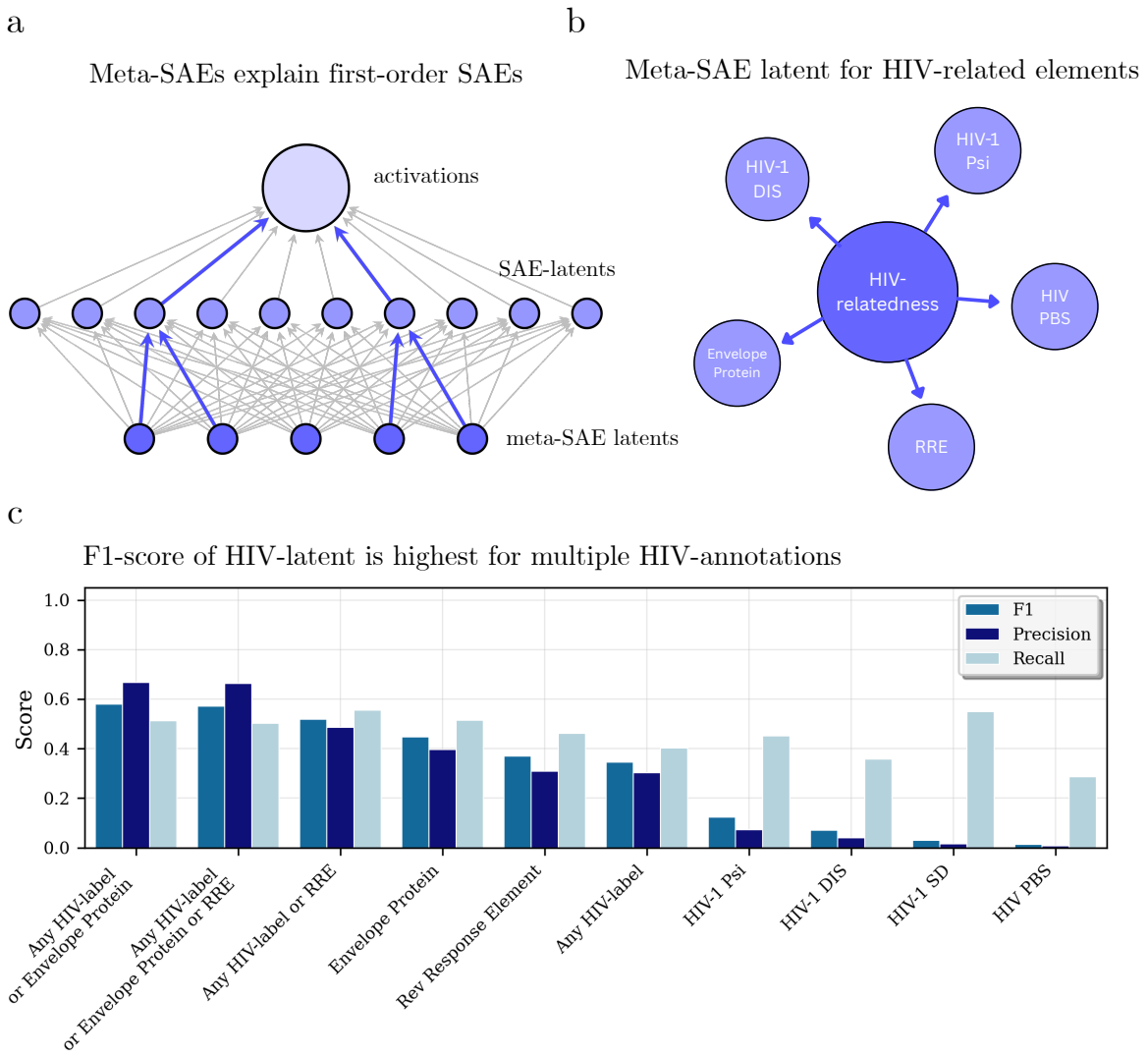

Figure 3: **Meta-SAEs identify an abstract concept related to HIV.** a) While SAEs are trained to reconstruct model activations, meta-SAEs are trained to reconstruct the decoder weights of first-order SAE-latents. b) We find a meta-SAE latent that activates on a cluster of HIV-related first-order SAE-latents. c) We assess this "HIV-latent" by treating its encoder-weights as a probe and measuring the probe F1-score across simple HIV-related annotations and their combinations, finding that the probe best separates a collection of many annotations.

We find that several Meta-SAE latents activate on clusters of semantically related child-latents. One striking example is latent #107, which activates most strongly on child-latents associated with HIV-related annotations: the Rev Response Element (RRE), HIV-1 dimerisation initiation

sequence (HIV-1 DIS), HIV-1 psi packaging element (HIV-1 psi), HIV primer binding site (HIV PBS), and envelope proteins.

To measure the association between latent #107 and this set of annotations, we extracted the corresponding encoder row and evaluated it as a linear classifier. To do this, we first determined an optimal bias on a validation set of annotated sequences and then measured the F1-score on a hold-out test set of 500 annotated plasmid sequences. As a control, we also measured the F1-score for every subset of annotations, to assess if the latent corresponds to a more specific set of features. As shown in Fig. 3, we find that the latent's F1-score is relatively high for all the annotations (57.2%) and substantially higher than for any individual annotation ($\leq 44.8\%$).

These results indicate that Meta-SAEs recover hierarchical structure within the representations of genomic language models. Whereas the base SAE isolates interpretable, single-annotation features, the Meta-SAE aggregates them into higher-order modules that correspond to coordinated biological processes. For example, latent #107 encapsulates multiple HIV-related sequence elements, RRE, DIS, Psi, PBS, and Env, reflecting the model's ability to link dispersed regulatory regions. This hierarchical organisation suggests that model features are not independent but form groupings reflecting shared regulatory or structural roles. By quantifying these relationships, Meta-SAEs provide a means to map dependencies between features. In practical terms, this enables more systematic interpretation of genomic language models, revealing how they encode multi-scale biological structure rather than isolated annotations.

2.4 Steering toward the A1408G substitution in 16S rRNA with SAE-latents

Having observed that SAE-latents can closely capture known biological annotations, we asked whether individual SAE-latents can causally modulate sequence predictions toward biologically meaningful variants. *Steering* refers to a causal intervention in the model's activation space, which results in a desired change in the model's output sequence. Steering involves reconstructing the model's activations using our SAE, while clamping an SAE-latent to a high value, and then continuing the forward pass with that reconstruction (see Methods 1).

Among features with confident annotations, latent #5066 exhibited a strong association with kanamycin resistance ($F1 > 0.6$, precision ≈ 0.9). We focused on the well-characterised A1408G substitution in 16S rRNA (Hobbie et al., 2006; Lynch & Puglisi, 2001; Nessar et al., 2011) and evaluated how manipulating #5066 alters the model's probability assigned to the corresponding "mutation token". We explore clamping to values that lie in and out of distribution. We define in-distribution steering as setting a latent to values within the empirical range observed on the steered sequence, while out-of-distribution refers to values beyond this range. We perform the steering on layer 10 of the 500M multi-species model, on which the SAE was trained.

At in-distribution steering levels, steering using #5066 at the masked position and also the immediately preceding position increased the A1408G logit and probability, promoting it to Top-1 among candidate tokens. Steering at a distant position (100 tokens before) had no effect, as did steering only on a masked or nearby token (token preceding the mask).

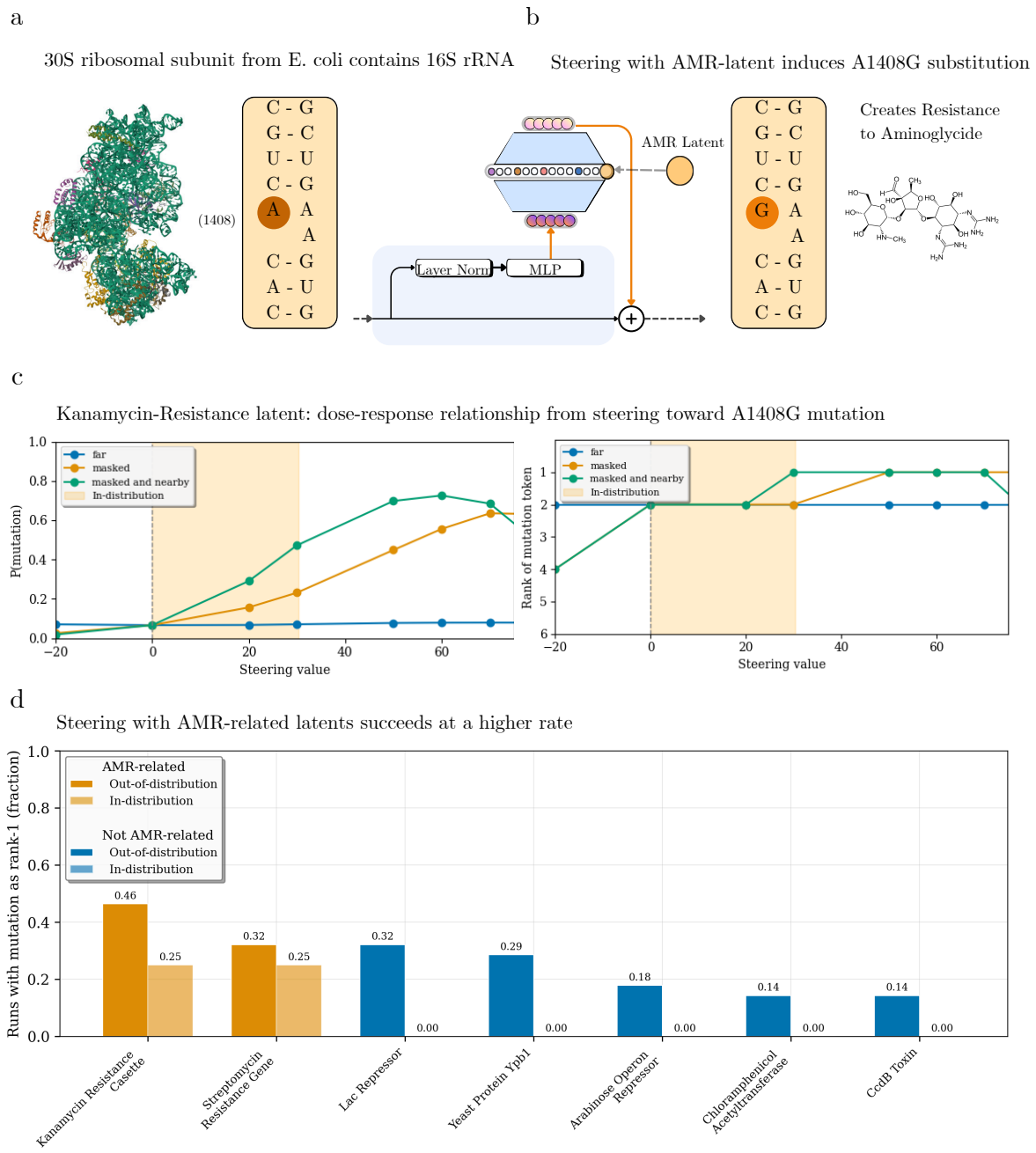

Figure 4: **Biologically meaningful steering with a Kanamycin resistance SAE-latent.** a) The 30S ribosomal subunit in E. coli (left) contains the 16S rRNA folded onto itself (centre), highlighting nucleotide position 1408, where the A to G substitution confers aminoglycide resistance. Structure rendered from RCSB Protein Data Bank (PDB ID: 7OE1; (Maksimova et al., 2021)). b) Steering (left) involves modifying the forward pass by reconstructing the MLP outputs using our SAE, with a latent for antimicrobial resistance, clamped to a high value. This induces the A1408G substitution (right), which confers aminoglycide resistance. Kanamycin A structure (right; image by SSamadi15, Wikimedia Commons, CC BY-SA 4.0). c) Shows that the model shifts probability mass toward the A1408G mutation as we increase the strength of SAE-steering (left) and how the rank of the mutation-token changes (right). Steering on the masked token and nearby tokens leads to a faster increase in probability mass (left) and is necessary for rank changes on in-distribution steering levels (right). d) We find that steering towards the mutation-containing token succeeds at a higher rate for Kanamycin Resistance latent compared to other interpretable SAE-latents. This is especially true if we only steer at in-distribution levels, where the success rate for all other latents becomes zero.

The rank of the mutation token in the non-steered forward pass was already very high at rank 2. Therefore, the change to rank 1 due to some random perturbation may not be an unlikely event. We address this concern by screening other well-annotated latents (F1 ≥ 0.5), which showed that this in-distribution effect was selective: only #5066 and #4420 (annotated as SmR, a streptomycin resistance gene) induced a Top-k rank shift, whereas the non-AMR latents did not (Fig. 4b).

We next characterised the dose–response relationship by scaling the steering magnitude beyond in-distribution ranges. The effect size on A1408G increased monotonically up to a breaking point, and sign-reversal (negative steering) suppressed it (Fig. 4a), and for many of the high steering values, steering at either mask or nearby token is sufficient for the rank change. At these higher doses, several non-AMR latents also produced rank shifts (Fig. 4b), indicating that specificity degrades outside the model's activation distribution. In fact, many of the latents show a dose-dependent effect on the token's probability similar to that of the antibiotic-resistant latents #5066 and #4420 or an inverted dose-dependent effect where the negative values steer towards greater likelihood.

Together, these results suggest that AMR-annotated features contain causal directions that can steer predictions toward resistance mutations at realistic activation levels. We note that the SAE has been trained to approximate the activation space of the nucleotide model. Thus, steering with out-of-distribution values can move our reconstruction out of the space the SAE learned to approximate well and so lead to unexpected effects on the predicted sequence. Furthermore, it has been shown that in artificial neural networks, semantic directions in activation space are not scaling invariant, and so scaling a direction that correlates with some semantic meaning can change the semantic meaning of the vector, especially when moving out of distribution (Black et al., 2022). Together, this might mean that steering works reliably only if we clamp latents to values within the distribution of values present in the sequence, which would be consistent with our observations. Our results also showcase the fragility of steering tasks with respect to the steering method used, especially in biological models where vocabulary size is limited and the underlying meaning of tokens and wider context windows is opaque.

3 Conclusion

As genomic language models become capable of composing novel functional genomes, it becomes increasingly plausible that their hidden representations encode a rich set of biological properties and structures that may at some point even surpass our own understanding of genomics. This could enable important scientific breakthroughs. It also becomes increasingly plausible that these models can be misused to design novel biohazards. These possibilities make it critical to build tools to interpret and steer genomic language models, both to extract scientific insight and to build effective guardrails that maximise the benefits of these tools.

Our central hypothesis was that genomic language model activations encode a large vocabulary of biological properties in superposition, and that sparse autoencoders can reveal this vocabulary by disentangling their representations. We tested this by training sparse autoencoders on the Nucleotide Transformer and showed that SAE-latents correspond to a diverse array of

interpretable biological concepts, uncovering over 100 distinct functional annotations encoded in the model activations. This included viral regulatory elements that we traced to training data contamination, demonstrating that interpretability methods can help diagnose quality issues in large-scale genomic datasets. We then showed that the Nucleotide Transformer encodes concepts at different levels of generality. Using meta-SAEs, trained on the weights of a base SAE, we showed that the model encodes a more general concept related to HIV. Finally, we showed that the concepts extracted by our SAE can modulate the Nucleotide Transformer's outputs in biologically plausible ways. In particular, we show that steering with an SAE-latent related to Kanamycin or Streptomycin Resistance causes the model to predict the A1408G substitution in the 16s rRNA, which confers aminoglycide resistance.

Together, these findings establish SAEs as an effective method for analysing representations of genomic language models, with applications in feature discovery, dataset auditing, and tracking representational shifts during finetuning. Future work should scale SAEs to larger genomic models to explore whether they encode a more extensive and fine-grained set of biological concepts. Additional research should apply SAEs to less-annotated regions and develop SAE-steering strategies tailored to complex sequence generation. These advances are realistic, given their success in other domains (Templeton et al., 2024; Simon & Zou, 2025), and would position SAEs as a general tool for understanding, auditing, and ultimately designing biological systems.

Acknowledgements

We would like to thank James Black, Sam Brown and Laura Dillon for helpful discussion and feedback (ordered alphabetically). We would also like to thank funders: AM acknowledges funding by Open Philanthropy, OMC acknowledges funding from a New College Todd-Bird Junior Research Fellowship and MRC Fellowship MR/Y010078/1. He acknowledges consulting fees to Pelago Biosciences, Faculty.ai and MarketCast and is on the scientific advisory board of Evolvere Biosciences. No funder had a role in the research or decision to publish. PJ is supported by funding from the Biotechnology and Biological Sciences Research Council UKRI-BBSRC grant (BB/T008784/1). PJ cofounded Evolvere Biosciences, but the company had no role in this study. FD's contribution to this work was financially supported by the Hector Fellow Academy. GMM thanks the EPSRC for support via EP/S024093/1.

4 Code and Reproducibility

All code developed for SAE training, evaluation, and analysis, along with the trained model weights and scripts to reproduce our main figures, are made publicly available at COMING SOON.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes, November 2018. URL http://arxiv.org/abs/1610.01644. arXiv:1610.01644 [stat].

Benegas, G., Batra, S. S., and Song, Y. S. DNA language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120(44): e2311219120, October 2023. doi: 10.1073/pnas.2311219120. URL https://www.pnas.org/doi/10.1073/pnas.2311219120. Publisher: Proceedings of the National Academy of Sciences.

Benegas, G., Ye, C., Albors, C., Li, J. C., and Song, Y. S. Genomic language models: opportunities and challenges. *Trends in Genetics*, January 2025. ISSN 0168-9525. doi: 10.1016/j.tig.2024.11.013. URL https://www.sciencedirect.com/science/article/pii/S0168952524002956.

Black, S., Sharkey, L., Grinsztajn, L., Winsor, E., Braun, D., Merizian, J., Parker, K., Guevara, C. R., Millidge, B., Alfour, G., and Leahy, C. Interpreting Neural Networks through the Polytope Lens, November 2022. URL http://arxiv.org/abs/2211.12312. arXiv:2211.12312 [cs].

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*, 2023.

Brixi, G., Durrant, M. G., Ku, J., Poli, M., Brockman, G., Chang, D., Gonzalez, G. A., King, S. H., Li, D. B., Merchant, A. T., Naghipourfar, M., Nguyen, E., Ricci-Tam, C., Romero, D. W., Sun, G., Taghibakshi, A., Vorontsov, A., Yang, B., Deng, M., Gorton, L., Nguyen, N., Wang, N. K., Adams, E., Baccus, S. A., Dillmann, S., Ermon, S., Guo, D., Ilango, R., Janik, K., Lu, A. X., Mehta, R., Mofrad, M. R. K., Ng, M. Y., Pannu, J., Ré, C., Schmok, J. C., John, J. S., Sullivan, J., Zhu, K., Zynda, G., Balsam, D., Collison, P., Costa, A. B., Hernandez-Boussard, T., Ho, E., Liu, M.-Y., McGrath, T., Powell, K., Burke, D. P., Goodarzi, H., Hsu, P. D., and Hie, B. L. Genome modeling and design across all domains of life with Evo 2, February 2025. URL https://www.biorxiv.org/content/10.1101/2025.02.18.638918v1. Pages: 2025.02.18.638918 Section: New Results.

Bussmann, B., Leask, P., and Nanda, N. BatchTopK Sparse Autoencoders, December 2024. URL http://arxiv.org/abs/2412.06410. arXiv:2412.06410 [cs].

Chen, K., Zhou, Y., Ding, M., Wang, Y., Ren, Z., and Yang, Y. Self-supervised learning on millions of primary RNA sequences from 72 vertebrates improves sequence-based RNA splicing prediction. *Briefings in Bioinformatics*, 25(3):bbae163, May 2024. ISSN 1477-4054. doi: 10.1093/bib/bbae163. URL https://doi.org/10.1093/bib/bbae163.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse Autoencoders Find Highly Interpretable Features in Language Models, October 2023. URL http://arxiv.org/abs/2309.08600. arXiv:2309.08600.

Dalla-Torre, H., Gonzalez, L., Mendoza-Revilla, J., Lopez Carranza, N., Grzywaczewski, A. H., Oteri, F., Dallago, C., Trop, E., de Almeida, B. P., Sirelkhatim, H., Richard, G., Skwark, M., Beguir, K., Lopez, M., and Pierrot, T. Nucleotide Transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, February 2025. ISSN 1548-7105. doi: 10.1038/s41592-024-02523-z. URL https://www.nature.com/articles/s41592-024-02523-z. Publisher: Nature Publishing Group.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy Models of Superposition, September 2022. URL http://arxiv.org/abs/2209.10652. arXiv:2209.10652 [cs].

Farrell, E., Lau, Y.-T., and Conmy, A. Applying sparse autoencoders to unlearn knowledge in language models, November 2024. URL http://arxiv.org/abs/2410.19278. arXiv:2410.19278 [cs].

Ferruz, N., Schmidt, S., and Höcker, B. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348, July 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-32007-7. URL https://www.nature.com/articles/s41467-022-32007-7.

Gao, L., Tour, T. D. l., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders, June 2024. URL http://arxiv.org/abs/2406.04093. arXiv:2406.04093.

Gujral, O., Bafna, M., Alm, E., and Berger, B. Sparse autoencoders uncover biologically interpretable features in protein language model representations. *Proceedings of the National Academy of Sciences*, 122(34):e2506316122, August 2025. doi: 10.1073/pnas.2506316122. URL https://www.pnas.org/doi/10.1073/pnas.2506316122. Publisher: Proceedings of the National Academy of Sciences.

Hayes, T., Rao, R., Akin, H., Sofroniew, N. J., Oktay, D., Lin, Z., Verkuil, R., Tran, V. Q., Deaton, J., Wiggert, M., Badkundri, R., Shafkat, I., Gong, J., Derry, A., Molina, R. S., Thomas, N., Khan, Y. A., Mishra, C., Kim, C., Bartie, L. J., Nemeth, M., Hsu, P. D., Sercu, T., Candido, S., and Rives, A. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, February 2025. doi: 10.1126/science.ads0018. URL https://www.science.org/doi/10.1126/science.ads0018. Publisher: American Association for the Advancement of Science.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, Santiago, Chile, December 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123. URL http://ieeexplore.ieee.org/document/7410480/.

Hobbie, S. N., Pfister, P., Bruell, C., Sander, P., François, B., Westhof, E., and Böttger, E. C. Binding of neomycin-class aminoglycoside antibiotics to mutant ribosomes with alterations in the A site of 16S rRNA. *Antimicrobial Agents and Chemotherapy*, 50(4):1489–1496, April 2006. ISSN 0066-4804. doi: 10.1128/AAC.50.4.1489-1496.2006.

Hwang, Y., Cornman, A. L., Kellogg, E. H., Ovchinnikov, S., and Girguis, P. R. Genomic language model predicts protein co-regulation and function. *Nature Communications*, 15(1):2880, April 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-46947-9. URL https://www.nature.com/articles/s41467-024-46947-9. Publisher: Nature Publishing Group.

Immer, A., Torroba Hennigen, L., Fortuin, V., and Cotterell, R. Probing as Quantifying Inductive Bias. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1839–1851, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.129. URL https://aclanthology.org/2022.acl-long.129/.

Jiang, M., Anderson, J., Gillespie, J., and Mayne, M. uShuffle: A useful tool for shuffling biological sequences while preserving the k-let counts. *BMC Bioinformatics*, 9(1):192, April 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-192. URL https://doi.org/10.1186/1471-2105-9-192.

Kamens, J. The Addgene repository: an international nonprofit plasmid and data resource. *Nucleic Acids Research*, 43(D1):D1152–D1157, January 2015. ISSN 0305-1048. doi: 10.1093/nar/gku893. URL https://doi.org/10.1093/nar/gku893.

King, S. H., Driscoll, C. L., Li, D. B., Guo, D., Merchant, A. T., Brixi, G., Wilkinson, M. E., and Hie, B. L. Generative design of novel bacteriophages with genome language models, September 2025. URL https://www.biorxiv.org/content/10.1101/2025.09.12.675911v1. ISSN: 2692-8205 Pages: 2025.09.12.675911 Section: New Results.

Leask, P., Bussmann, B., Pearce, M., Bloom, J., Tigges, C., Moubayed, N. A., Sharkey, L., and Nanda, N. Sparse Autoencoders Do Not Find Canonical Units of Analysis, February 2025. URL http://arxiv.org/abs/2502.04878. arXiv:2502.04878 [cs].

Li, N., Pan, A., Gopal, A., Yue, S., Berrios, D., Gatti, A., Li, J. D., Dombrowski, A.-K., Goel, S., Phan, L., Mukobi, G., Helm-Burger, N., Lababidi, R., Justen, L., Liu, A. B., Chen, M., Barrass, I., Zhang, O., Zhu, X., Tamirisa, R., Bharathi, B., Khoja, A., Zhao, Z., Herbert-Voss, A., Breuer, C. B., Marks, S., Patel, O., Zou, A., Mazeika, M., Wang, Z., Oswal, P., Lin, W., Hunt, A. A., Tienken-Harder, J., Shih, K. Y., Talley, K., Guan, J., Kaplan, R., Steneker, I., Campbell, D., Jokubaitis, B., Levinson, A., Wang, J., Qian, W., Karmakar, K. K., Basart, S., Fitz, S., Levine, M., Kumaraguru, P., Tupakula, U., Varadharajan, V., Wang, R., Shoshitaishvili, Y., Ba, J., Esvelt, K. M., Wang, A., and Hendrycks, D. The WMDP Benchmark: Measuring and Reducing Malicious Use With Unlearning, May 2024. URL http://arxiv.org/abs/2403.03218. arXiv:2403.03218 [cs].

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/10.1126/science.ade2574. Publisher: American Association for the Advancement of Science.

Lynch, S. R. and Puglisi, J. D. Structural origins of aminoglycoside specificity for prokaryotic ribosomes. *Journal of Molecular Biology*, 306(5):1037–1058, March 2001. ISSN 0022-2836. doi: 10.1006/jmbi.2000.4420.

Madani, A., Krause, B., Greene, E. R., Subramanian, S., Mohr, B. P., Holton, J. M., Olmos, J. L., Xiong, C., Sun, Z. Z., Socher, R., Fraser, J. S., and Naik, N. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8): 1099–1106, August 2023. ISSN 1546-1696. doi: 10.1038/s41587-022-01618-2. URL https://www.nature.com/articles/s41587-022-01618-2. Publisher: Nature Publishing Group.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 689–696, New York, NY, USA, June 2009. Association for Computing Machinery. ISBN

978-1-60558-516-1. doi: 10.1145/1553374.1553463. URL https://doi.org/10.1145/1553374.1553463.

Maksimova, E. M., Korepanov, A. P., Kravchenko, O. V., Baymukhametov, T. N., Myasnikov, A. G., Vassilenko, K. S., Afonina, Z. A., and Stolboushkina, E. A. RbfA Is Involved in Two Important Stages of 30S Subunit Assembly: Formation of the Central Pseudoknot and Docking of Helix 44 to the Decoding Center. *International Journal of Molecular Sciences*, 22(11):6140, January 2021. ISSN 1422-0067. doi: 10.3390/ijms22116140. URL https://www.mdpi.com/1422-0067/22/11/6140. Publisher: Multidisciplinary Digital Publishing Institute.

McGuffie, M. J. and Barrick, J. E. pLannotate: engineered plasmid annotation. *Nucleic Acids Research*, 49(W1):W516–W522, July 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab374. URL https://doi.org/10.1093/nar/gkab374.

Nessar, R., Reyrat, J. M., Murray, A., and Gicquel, B. Genetic analysis of new 16S rRNA mutations conferring aminoglycoside resistance in Mycobacterium abscessus. *The Journal of Antimicrobial Chemotherapy*, 66(8):1719–1724, August 2011. ISSN 1460-2091. doi: 10.1093/jac/dkr209.

Nguyen, E., Poli, M., Faizi, M., Thomas, A., Birch-Sykes, C., Wornow, M., Patel, A., Rabideau, C., Massaroli, S., Bengio, Y., Ermon, S., Baccus, S. A., and Ré, C. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution, November 2023. URL http://arxiv.org/abs/2306.15794. arXiv:2306.15794.

Nijkamp, E., Ruffolo, J. A., Weinstein, E. N., Naik, N., and Madani, A. ProGen2: Exploring the boundaries of protein language models. *Cell Systems*, 14(11):968–978.e3, November 2023. ISSN 2405-4712. doi: 10.1016/j.cels.2023.10.002. URL https://www.sciencedirect.com/science/article/pii/S2405471223002727.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving Dictionary Learning with Gated Sparse Autoencoders, April 2024. URL http://arxiv.org/abs/2404.16014. arXiv:2404.16014.

Simon, E. and Zou, J. InterPLM: discovering interpretable features in protein language models via sparse autoencoders. *Nature Methods*, pp. 1–11, September 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02836-7. URL https://www.nature.com/articles/s41592-025-02836-7. Publisher: Nature Publishing Group.

Steinegger, M. and Salzberg, S. L. Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. *Genome Biology*, 21(1):115, May 2020. ISSN 1474-760X. doi: 10.1186/s13059-020-02023-1. URL https://doi.org/10.1186/s13059-020-02023-1.

Tang, Z., Somia, N., Yu, Y., and Koo, P. K. Evaluating the representational power of pre-trained DNA language models for regulatory genomics. *bioRxiv*, pp. 2024.02.29.582810, September 2024. ISSN 2692-8205. doi: 10.1101/2024.02.29.582810. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10925287/.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Wu, Z., Arora, A., Geiger, A., Wang, Z., Huang, J., Jurafsky, D., Manning, C. D., and Potts, C. AxBench: Steering LLMs? Even Simple Baselines Outperform Sparse Autoencoders, January 2025. URL http://arxiv.org/abs/2501.17148. arXiv:2501.17148 [cs].

Yun, Z., Chen, Y., Olshausen, B. A., and LeCun, Y. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors, April 2023. URL http://arxiv.org/abs/2103.15949. arXiv:2103.15949.

A Related Work

A central challenge in interpretability research is that neural networks rarely represent input features at the level of individual neurons [CITE]. Instead, features are encoded as directions in high-dimensional activation space, which do not align with its standard basis directions (neurons), a phenomenon described by the *superposition hypothesis* (Elhage et al., 2022). This allows models to encode more features than neurons, but makes individual activations polysemantic and difficult to interpret directly. One approach to recovering these underlying feature directions is *sparse dictionary learning*, which approximates an activation vector $\mathbf{x}$ as a sparse, weighted sum of basis directions:

$$\mathbf{x} \approx \sum_i w_i \mathbf{d}_i, \quad \text{s.t. } \|\{w_i \mid w_i > 0\}\| \text{ is small,} \tag{1}$$

where $\mathbf{d}_i$ are dictionary atoms and w_i are sparse coefficients.

Sparse autoencoders (SAEs) are one implementation of this idea. Formally, SAEs consist of an encoder $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ and a decoder $g_\phi : \mathbb{R}^m \to \mathbb{R}^n$ trained to minimize the reconstruction error

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{x} - g_\phi(f_\theta(\mathbf{x}))\|^2 \right] + \lambda\, \Omega(f_\theta(\mathbf{x})), \tag{2}$$

where Ω is a sparsity-inducing regularizer and λ controls its strength. By learning an overcomplete dictionary under a sparsity constraint, SAEs are intended to disentangle superimposed features and yield more interpretable latent units.

We compare SAEs to three alternative approaches to interpreting the gLMs' latent representations. First, researchers have applied *unsupervised clustering* of sequence embeddings with UMAP (Benegas et al., 2023; Chen et al., 2024) or t-SNE (Nguyen et al., 2023). In contrast to SAEs, the primary purpose of these methods was likely more to visualise the high-dimensional model representations in 2D, *not* to extract features from the model. Even when used for feature discovery, they would provide clusters of similar embeddings but would not reveal explicit feature directions in the model's latent space, which SAEs do. Second, *attention weight analysis*,

such as in Chen et al. (2024), provides insights into long-range dependencies between sequence elements but, unlike SAEs, does not directly isolate disentangled features in the model's representation space. Third, many papers have trained classifiers (*probes*) on hidden representations of the model to test if the model representations encode various functional genetic elements (Benegas et al., 2023; Dalla-Torre et al., 2025; Tang et al., 2024; Hwang et al., 2024). Unlike SAEs, probing classifiers require supervised labels, making them well-suited for testing predefined hypotheses about what features a model encodes. However, this reliance on labeled data limits their usefulness for discovering novel, unexpected representations - a key advantage of SAEs.

By identifying explicit directions in the representation space, SAEs enable interventions such as *model steering or unlearning* specific encoded features. However, recent work in the domain of natural language suggests that alternatives outperform SAE-based methods for both purposes: Wu et al. (2025) suggest that alternatives like prompting techniques are more effective for steering and Farrell et al. (2024) suggests that Representation Misdirection for Unlearning (Li et al., 2024) provides several advantages for unlearning.

We focused this section on gLM interpretability techniques as applied to model representations. This does not cover other interpretability techniques, e.g. methods for characterizing the relative importance of various input parts to the model's output. For a more comprehensive review of interpretability methods for gLMs, see Benegas et al. (2025).

Most models previously interpreted with SAEs were Transformers trained on large corpora of text (Yun et al., 2023; Cunningham et al., 2023). Recently, this approach has been extended to protein Language Models, ESM-2 (Simon & Zou, 2025), and even the genomic language model EVO 2 (Brixi et al., 2025). In contrast to (Brixi et al., 2025), we apply this method to a standard Transformer-based gLM that is pretrained with a masked language modelling objective: the Nucleotide Transformer from Dalla-Torre et al. (2025).

A key challenge in understanding the representations learned by Sparse Autoencoders (SAEs) is interpreting the latent activations in a meaningful way. Previous work primarily relied on manual inspection of the most activating inputs to identify unifying concepts behind each latent dimension or used large language models (LLMs) to automate this process (Bricken et al., 2023; Gao et al., 2024). While this approach has been effective in domains such as natural language and vision, it is less suitable for biological sequence data, where meaningful features often correspond to functional or regulatory elements that are not trivially observable from raw sequence patterns. To address this limitation, we follow Simon & Zou (2025) in using automatically annotated biological features as an intermediate step before interpreting SAE-latents. These annotations allow us to quantitatively measure the association of an SAE latent and a concept.

In designing the SAE architecture, the choice of activation function plays a crucial role in the effectiveness of SAEs at extracting interpretable features. Early SAE implementations primarily used ReLU activations (Cunningham et al., 2023; Bricken et al., 2023), which enforce sparsity indirectly by thresholding negative activations at zero. To promote sparsity while avoiding "activation shrinkage", the Jump-ReLU activation function was proposed by Rajamanoharan et al. (2024). An alternative line of work has explored activation functions that fix the number

of active latents for a set of inputs. Gao et al. (2024) introduced Top-K activation functions, setting all but the K largest activations for a particular input to zero. More recent work by Bussmann et al. (2024) has modified this approach by allowing the top $k \times n$ latents to be active across a sample of n inputs, enabling more flexibility in the number of dictionary elements used to reconstruct a single input. We choose the BatchTopK activation function as it is widely regarded as being more effective than TopK and easier to train than JumpReLU SAEs.

As introduced by Bussmann et al. (2024), BatchTopK Sparse Autoencoder (BatchTopK SAE) is a variant of the TopK sparse autoencoder that enforces sparsity at the *batch level* rather than per sample. Given an input batch $X \in \mathbb{R}^{n \times d}$ consisting of n samples, the encoder produces activations

$$Z = W_{\text{enc}} X + b_{\text{enc}},$$

where $W_{\text{enc}} \in \mathbb{R}^{d \times m}$ and $b_{\text{enc}} \in \mathbb{R}^m$. The BatchTopK operator retains the top $n \times k$ activation values across the entire batch, setting all others to zero:

$$\text{BatchTopK}(Z)_{i,j} = \begin{cases} Z_{i,j}, & \text{if } Z_{i,j} \text{ is among the top } n \times k \text{ activations in the batch,} \\ 0, & \text{otherwise.} \end{cases}$$

The decoder reconstructs the batch as

$$\hat{X} = (\text{BatchTopK}(Z)) W_{\text{dec}} + b_{\text{dec}},$$

where $W_{\text{dec}} \in \mathbb{R}^{m \times d}$ and $b_{\text{dec}} \in \mathbb{R}^d$. The training objective minimizes the reconstruction loss with an auxiliary regularization term to prevent dead latents:

$$L(X) = \|X - \hat{X}\|_2^2 + \alpha L_{\text{aux}}.$$

During training, the encoder computes activations Z, after which the BatchTopK operator selects the globally top activations across all samples in the batch. This allows individual samples to activate a variable number of latents, improving reconstruction quality while maintaining a fixed *average sparsity*. During inference, the batch dependency is removed by estimating a global activation threshold

$$\theta = \mathbb{E}_X \left[\min\{z_{i,j}(X) \mid z_{i,j}(X) > 0\} \right],$$

and replacing the BatchTopK operator with a JumpReLU-style activation:

$$\text{JumpReLU}_\theta(z) = z\, H(z - \theta),$$

where $H(\cdot)$ is the Heaviside step function. This yields a fixed thresholding rule that preserves the sparsity structure learned during training.

A *Meta-SAE* (Leask et al., 2025) extends the sparse autoencoder framework by learning higher-level structure among features discovered by a base SAE. Instead of being trained on raw activations, it is trained on the decoder weights of a previously trained SAE. Let $W_{\text{dec}}^{(1)} \in \mathbb{R}^{m_1 \times d}$ be the decoder of the base SAE, where each row $x_i^{(2)}$ represents a learned feature in the original activation space $\mathbb{R}^d$. Stacking these feature vectors gives the dataset for the Meta-SAE:

$$X^{(2)} = W_{\text{dec}}^{(1)} \in \mathbb{R}^{m_1 \times d}.$$

The Meta-SAE defines an encoder-decoder pair:

$$Z^{(2)} = W_{\text{enc}}^{(2)} X^{(2)} + b_{\text{enc}}^{(2)}, \quad \hat{X}^{(2)} = \phi(Z^{(2)}) W_{\text{dec}}^{(2)} + b_{\text{dec}}^{(2)},$$

where $W_{\text{enc}}^{(2)} \in \mathbb{R}^{d \times m_2}$, $W_{\text{dec}}^{(2)} \in \mathbb{R}^{m_2 \times d}$, and $\phi(\cdot)$ is a sparsifying activation such as JumpReLU or TopK. The training objective minimises the reconstruction error of the base decoder weights:

$$L(X^{(2)}) = \|X^{(2)} - \hat{X}^{(2)}\|_2^2 + \alpha L_{\text{aux}}.$$

Each base feature (row of $W_{\text{dec}}^{(1)}$) is thus reconstructed as a sparse linear combination of meta-features (rows of $W_{\text{dec}}^{(2)}$):

$$x_i^{(2)} \approx \sum_{r=1}^{m_2} h_{ir}^{(2)} \, W_{\text{dec},r}^{(2)}.$$

where $h_{ir}^{(2)}$ are the sparse activations produced by the Meta-SAE encoder.

B Methods

B.1 Experimental Setup

Choice of Nucleotide Transformer We chose to focus on the Nucleotide Transformer (Dalla-Torre et al., 2025). One advantage of this model family is that it spans multiple scales, including a relatively small 50m parameter model, with relatively low-dimensional hidden states. This is advantageous because the computational cost of training SAEs grows linearly with the hidden-state size of the model (all else equal). A second reason for the Nucleotide Transformer family is that it has demonstrated strong benchmark performance, indicating that it has successfully encoded a variety of input properties, making it particularly promising for interpretability research.

Choice of Dataset The most obvious dataset for training a sparse autoencoder on the Nucleotide Transformer is the model's pretraining data, a dataset of 850 genomes, spanning multiple domains of life. However, functional annotations for these genomes are scarce, which results in most SAE-latents activating on regions without known function, such as hypothetical proteins. To avoid this problem, we chose to focus on sequences that can be densely and confidently annotated: plasmids. We downloaded a dataset of plasmids from AddGene, an online repository for genetically engineered plasmids (Kamens, 2015). Our dataset consisted of 93,306 plasmids for training and about 15,551 for validation and testing.

Creating Annotated Evaluation Sets To functionally annotate plasmid sequences, we used pLannotate (McGuffie & Barrick, 2021), a specialised tool for automated annotation of plasmid features. Our annotation pipeline consisted of the following steps:

1. Dataset preparation: We randomly selected three non-overlapping sets of 1,000 plasmid sequences each from a larger validation dataset of Addgene plasmids. To ensure reproducibility, we used a fixed random seed (42) for the selection process.

2. Annotation procedure: Each plasmid sequence was processed using pLannotate's annotate function with parameters set to include detailed annotations and treating sequences as linear. This procedure identified functional genetic elements including promoters, resistance markers, replication origins, and other standard plasmid features.

3. Annotation output: For each plasmid, the annotation process generated a comprehensive table of hits containing feature coordinates, feature types, and confidence scores (e-values)

4. Data organization: We compiled annotation results for each set into separate CSV files, preserving sequence identifiers for cross-referencing. These annotation files served as the foundation for subsequent analyses of feature representation in the Nucleotide Transformer activations.

All annotation processing was performed using Python 3.8 with pLannotate v1.2.0, pandas v1.3.5, and BioPython v1.79. We performed almost all our training in Google Colabs, using either a single A100 (for SAE training) or a single L4 GPU (for linear probe training).

B.2 Training Sparse Autoencoders

For each plasmid sequence, we extracted MLP outputs at layer 1, 4, 7, 10 and 12 (final) of the 50m Nucleotide Transformer and at layer 10 of the 500m Nucleotide Transformer. We downloaded the models from HuggingFace.

As mentioned before, we used BatchTopK SAEs, with an expansion factor of 8. We trained each SAE for 100,000 steps with a batch size of 4096 and a maximum sequence length of 512 tokens. We used the Adam optimiser with learning rates set to 3e-4 and linear warmup over the first 10% of steps and linear decay over the last 20%. We set K-values to 16 or 32. We normalised activations by $\sqrt{d_{mlp}}$. Our goal was to demonstrate that SAEs can extract biological concepts from the Nucleotide Transformer, rather than finding the "best" SAE for this model. We used the hyperparameters from Bussmann et al. (2024).

Following the implementation of (Bussmann et al., 2024), we initialised the encoder as the transpose of the decoder weights. We maintained unit-norm decoder columns throughout training, while also performing gradient-clipping and removing gradients parallel to decoder columns before each update.

B.3 Evaluating Sparse Autoencoders

To screen promising SAEs, we initially relied on three proxy metrics:(1) normalised MSE, measuring reconstruction fidelity, (2) L1-loss, measuring sparsity, and (3) the fraction of dead latents, activating less than once per 10 million tokens. We detail these below for each of our SAEs.

To determine if our SAE-latents had successfully captured known biological concepts, we had to correlate activation of individual SAE-latents with the annotations of activating input tokens. For our analysis, a token (a 6-mer) was considered annotated if it shared at least one nucleotide

Table 2: SAE Evaluation Metrics Across Layers

Model Size	Layer	NMSE	L1 Loss	Dead Latents (%)
50M	L1	0.0075	0.0293	5.74
50M	L4	0.0842	0.0168	43.43
50M	L7	0.0682	0.0396	36.13
50M	L10	0.0846	0.0528	62.72
50M	L12	0.0743	0.1045	50.10
500M	L10	0.1179	0.0213	77.72

with a region identified by pLannotate.

Inspection of Individual Latents: To understand the function of individual SAE-latents, we examined the tokens that elicited the strongest activations for each latent. For each highly activating token, we recorded its annotation (if any), its surrounding sequence context, the annotation of that context, and a confidence score (e-value) associated with the annotation. This allowed us to qualitatively assess whether a latent was responding to specific biological features.

Measuring Monosemanticity: A key aspect of evaluating the learned SAE-latents is to quantify their monosemanticity – the degree to which each latent corresponds to a single, well-defined concept. While perfectly monosemantic neurons, which activate exclusively for a specific property, are rarely observed in practice, we aim to measure how closely our latents approach this ideal. For instance, a highly monosemantic latent for "promoter" would ideally activate strongly and consistently for promoter sequences and remain inactive for other types of sequences. To quantify monosemanticity, we adopted the "Domain F1-score" metric, following the methodology of Simon & Zou (2025).

Domain Recall vs. Standard Recall: The primary difference between Domain F1-score and the standard F1-score lies in the calculation of recall. Standard recall measures the proportion of true positive tokens (tokens with a specific annotation) that a classifier correctly identifies. In contrast, Domain Recall focuses on "true domains" – contiguous regions of multiple tokens that are all annotated with the same functional element. Domain Recall calculates the fraction of these true domains in which a given latent activates on at least one constituent token. This modification significantly increases the measured recall, and consequently the F1-score, of our latents. The difference between these two recall metrics and their impact on the F1-score for our SAE-latents is visually illustrated in Figure 2. As expected, we generally observe lower values for the standard metrics. Notably, latents associated with puromycin resistance, envelope proteins, and streptomycin resistance maintain relatively high F1 values even under the standard metric.

F1-score with Activation Threshold: we swept thresholds over the observed activation range and selected the value that maximised validation F1. Importantly, the final F1-scores reported in our study are computed on two non-overlapping hold-out test sets to avoid overfitting.

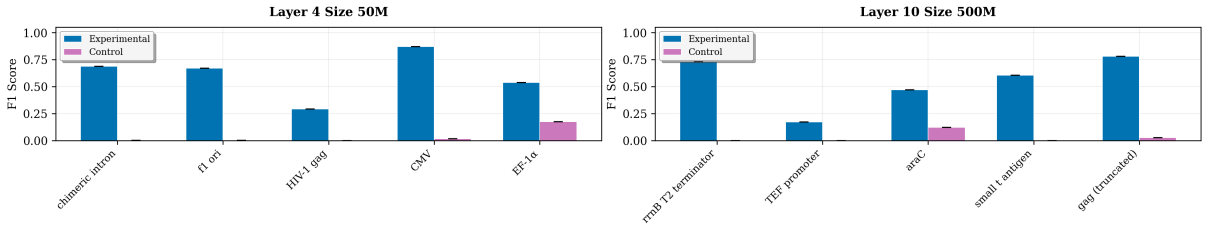

Figure 5: **Functional annotations are encoded via pretraining.** Comparing linear probe performance across 10 annotations in Layer 4 of the 50M model (left) and Layer 10 of the 500M model (right), consistently shows that the annotations are more linearly separable in the pretrained than in the base model.

B.4 Concept Validation Through Probing

For each concept, we train 30 logistic regressions on both the MLP outputs of the Nucleotide Transformer and a randomly initialised version. We train for 50 epochs, with early stopping after no improvement of at least 1e-5 in F1-score after 20 epochs. We use a batch size of 128. Detailed results are shown in Figure 5.

B.5 Training meta-SAEs

We train Meta-SAEs using three sparsity mechanisms: BatchTopK, JumpReLU, and classical Lasso-based dictionary learning (Mairal et al., 2009). All SAEs are trained with a batch size of 512, a learning rate of 2×10^{-4}, and for 5000 optimisation steps, using a linear warmup of the ℓ_1 penalty over the first 1000 steps. For JumpReLU models, we set the bandwidth to 0.01 and the initial threshold to 0.001. Encoders are initialised as the transpose of their corresponding decoders, and all weights use Kaiming Uniform initialisation (He et al., 2015). Across configurations, JumpReLU SAEs achieve slightly better reconstruction scores than BatchTopK or Lasso-based dictionary learning. Increasing the latent dimensionality generally improves reconstruction performance, with 128 latent units providing a good balance between compactness and accuracy.

B.6 SAE-based Steering

We assess whether sparse autoencoder (SAE) latents can be used to steer a masked language model (MLM) by directly manipulating latent activations at a chosen layer and reinjecting a residual-preserving reconstruction into the forward pass to obtain steered logits.

Model and layer. Steering is applied to the Nucleotide Transformer (NT) at a single unique layer during masked language modelling.

Overview of steering procedure. Given an input sequence s and a layer j:

1. **Capture baseline activations:** Run a forward pass and hook the MLP output at layer j to obtain per-position activations $\mathbf{a} \in \mathbb{R}^{N \times d}$, and record control logits $\mathbf{z}^{\text{ctrl}} \in \mathbb{R}^{N \times V}$.

2. **Decompose with SAE:** Pass $\mathbf{a}$ through SAE to obtain

$$(\tilde{\mathbf{a}},\, \mathbf{h}) = SAE(\mathbf{a}), \qquad \mathbf{r} = \mathbf{a} - \tilde{\mathbf{a}},$$

   where $\mathbf{h}$ are the hidden states of the SAE, $\tilde{\mathbf{a}}$ is the reconstruction of the activations and $\mathbf{r}$ is the reconstruction residual.

3. **Manipulate latents:** Let $I_{\max}$ be indices of latents to amplify and I_0 indices to set to zero. Optionally restrict to a set of token positions $P \subseteq \{1, \ldots, N\}$. For $i \in I_0$ set $h_{n,i} \leftarrow 0$; for $i \in I_{\max}$ set

$$h_{i,l} \leftarrow \begin{cases} v & \text{(fixed steering)} \\ \max\limits_{n',i'} h_{n',i'} & \text{(max-activation steering)} \end{cases} \qquad \text{for } n \in P \text{ (or all } n \text{ if } P \text{ is not specified).}$$

4. **Residual-preserving reconstruction:** Decode the edited latents and add back the residual:

$$\mathbf{a}^{\text{steer}} = SAE(\mathbf{h}) + \mathbf{r}.$$

5. **Steered forward pass:** Inject $\mathbf{a}^{\text{steer}}$ at layer j via a forward hook and complete the pass to obtain steered logits $\mathbf{z}^{\text{steer}}$.

Token- and vocabulary-level effects. We compute the logit difference

$$\Delta\mathbf{z} = \mathbf{z}^{\text{steer}} - \mathbf{z}^{\text{ctrl}} \in \mathbb{R}^{N \times V},$$

its mean over sequence positions $\bar{\Delta} \in \mathbb{R}^V$, and report the top-$k$ increased and decreased vocabulary items by entries of $\bar{\Delta}$.

Latent manipulation policies. Two steering-value policies are supported: (1) *fixed* ($v \in \mathbb{R}_{>0}$), and (2) *max-activation*, which sets selected latents to the maximum latent value observed in the current example. Steering can be applied globally or only at designated positions P (e.g., masked or motif-aligned tokens). We find that steering on the entire sequence leads to poor results.

Implementation details. Hooks intercept and substitute the output of the layer-j MLP projection (`output.dense`). Shapes are matched per batch/sequence; hooks are cleared after each pass. Returned diagnostics include $\mathbf{z}^{\text{ctrl}}$, $\mathbf{z}^{\text{steer}}$, lists of top increased/decreased tokens, and $\bar{\Delta}$.

Algorithm 1 SAE Latent Steering with Residual-Preserving Injection

Require: Model M, tokenizer T, SAE, layer index j
Require: Input sequence s; latent index sets $I_{\max}$ (to amplify) and I_0 (to zero); steering mode $m \in \{\text{FIXED}, \text{MAX-ACTIVATION}\}$; fixed value v if $m = \text{FIXED}$; optional positions P; top-K
Ensure: $\mathbf{z}^{\text{ctrl}}$, $\mathbf{z}^{\text{steer}}$, top-$K$ increased/decreased tokens, and $\bar{\Delta}$

$\mathbf{x} \leftarrow T(s)$ ▷ Tokenize
$(\mathbf{z}^{\text{ctrl}}, \mathbf{a}) \leftarrow \text{ForwardWithHook}(M, \mathbf{x}, \text{hook MLP output at layer } j)$
$(\tilde{\mathbf{a}}, \mathbf{h}) \leftarrow SAE(\mathbf{a})$ ▷ SAE decomposition
$\mathbf{r} \leftarrow \mathbf{a} - \tilde{\mathbf{a}}$ ▷ Reconstruction residual
$I \leftarrow P$ if specified else $\{1, \ldots, N\}$
for $n \in I$ **do**
  for $i \in I_0$ **do**
    $h_{n,i} \leftarrow 0$
  end for
  for $i \in I_{\max}$ **do**
    $h_{n,i} \leftarrow \begin{cases} v & \text{if } m = \text{FIXED} \\ \max\limits_{n',i'} h_{n',i'} & \text{if } m = \text{MAX-ACTIVATION} \end{cases}$
  end for
end for
$\hat{\mathbf{a}} \leftarrow SAE(\mathbf{h})$ ▷ Decode edited latents
$\mathbf{a}^{\text{steer}} \leftarrow \hat{\mathbf{a}} + \mathbf{r}$ ▷ Residual-preserving reconstruction
$\mathbf{z}^{\text{steer}} \leftarrow \text{ForwardWithInjection}(M, \mathbf{x}, \text{inject } \mathbf{a}^{\text{steer}} \text{ at layer } j)$
$\Delta\mathbf{z} \leftarrow \mathbf{z}^{\text{steer}} - \mathbf{z}^{\text{ctrl}}$; $\bar{\Delta} \leftarrow \text{MeanOverPositions}(\Delta\mathbf{z}; I)$
$\text{TopInc} \leftarrow \text{TopKIndices}(\bar{\Delta}, K)$; $\text{TopDec} \leftarrow \text{BottomKIndices}(\bar{\Delta}, K)$
return $\mathbf{z}^{\text{ctrl}}, \mathbf{z}^{\text{steer}}, \text{TopInc}, \text{TopDec}, \bar{\Delta}$

Outputs and reporting. We report (i) control vs. steered logits, (ii) $\bar{\Delta}$ over the vocabulary, and (iii) the top-K most increased/decreased tokens by $\bar{\Delta}$. When steering is position-restricted, summaries are computed over $i \in P$; otherwise over all positions.

C Supplementary Information

C.1 Characterising the Sequence Features Underlying Probe Performance

To understand what sequence features the CMV enhancer probe relies on, we tested two hypotheses: that it recognises specific regulatory motifs, or that it responds to simpler sequence statistics.

Motif ablation reveals minimal contribution of known binding sites. We systematically ablated four known motifs in CMV enhancers—E-box sequences, CREB/ATF sites, NF-κB binding sites, and CCAAT boxes—replacing each with random sequences of identical length. Only E-box ablation significantly reduced probe probability compared to both original and control sequences with random ablations ($p \leq 0.0001$, Wilcoxon signed rank test). The other

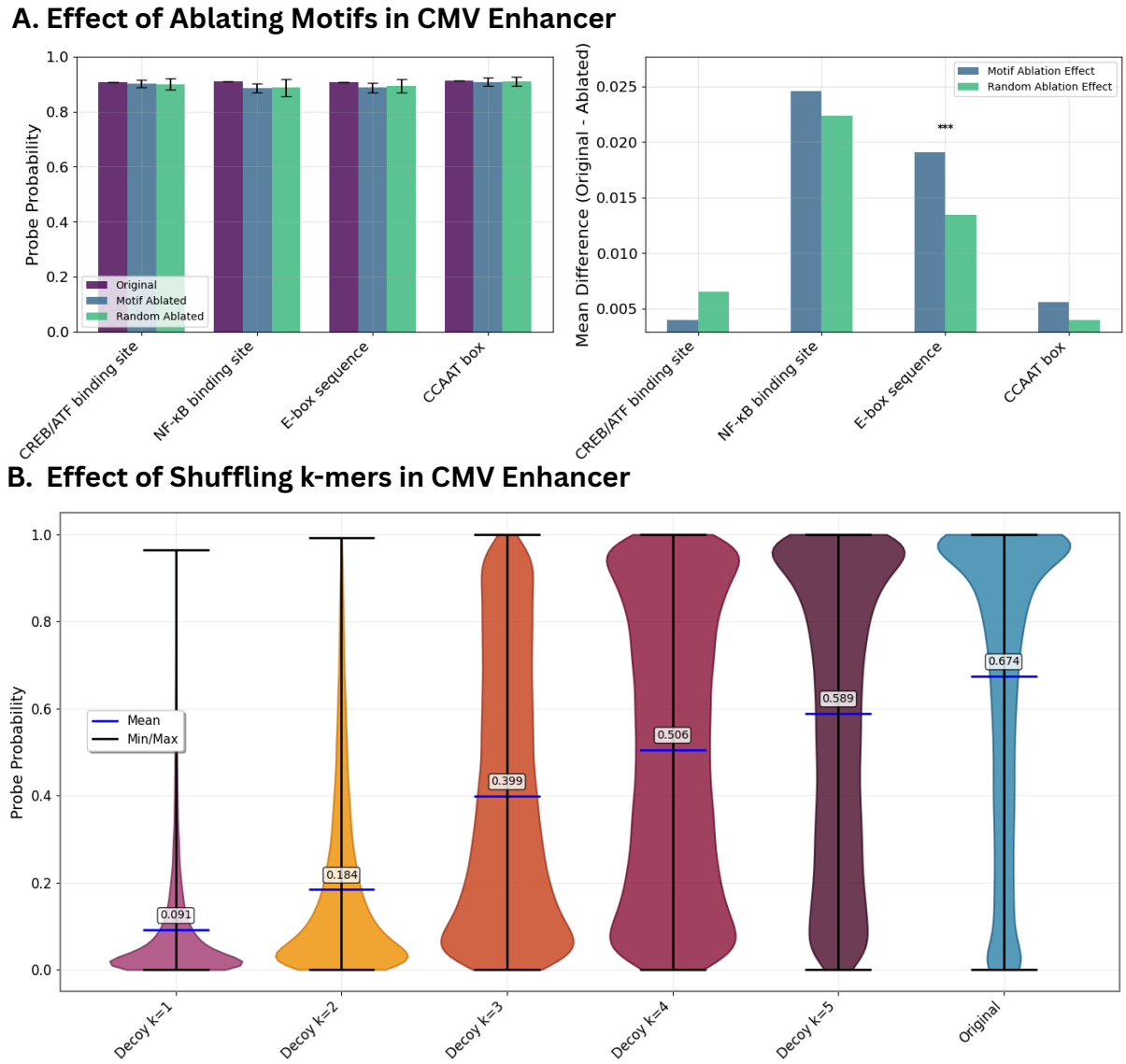

Figure 6: **The Effect of Disrupting the Sequences on CMV Enhancer Prediction.** A) Shows the effect of ablating motifs from CMV enhancer sequences on the probe probability assigned to these sequences. As baselines, we compare to the original sequence and a sequence with a random ablation of the same length. B) Compares the average probe probability for CMV enhancer sequences to decoys, shuffled sequences that preserve only the distribution of k-mers, not their order.

three motifs showed no significant effect, suggesting the probe does not primarily rely on these canonical regulatory elements.

K-mer statistics explain probe performance. We next tested whether simpler sequence features could account for probe behavior by generating decoy sequences that preserve k-mer frequencies while disrupting higher-order structure. For each k from 1 to 5, we shuffled sequences to maintain exact k-mer counts while randomizing their positions. Figure 6B shows that probe performance increased monotonically with k: preserving only nucleotide ($k = 1$) or dinucleotide

($k = 2$) frequencies yielded poor performance, while $k = 5$ recovered 85% of the original probe probability. This 5-mer dependency explains why longer motifs (6–12 bp) contribute minimally—the probe captures local sequence patterns rather than extended regulatory elements.