INTERPRETING AND STEERING PROTEIN LANGUAGE MODELS THROUGH SPARSE AUTOENCODERS

Edith N. Villegas Garcia & Alessio Ansuini
Data Engineering Laboratory
Area Science Park
Padriciano, TS 34149, Italy
{edith.villegas,alessio.ansuini}@areasciencepark.it

ABSTRACT

The rapid advancements in transformer-based language models have revolutionized natural language processing, yet understanding the internal mechanisms of these models remains a significant challenge. This paper explores the application of sparse autoencoders (SAE) to interpret the internal representations of protein language models, specifically focusing on the ESM-2 8M parameter model. By performing a statistical analysis on each latent component's relevance to distinct protein annotations, we identify potential interpretations linked to various protein characteristics, including transmembrane regions, binding sites, and specialized motifs. We then leverage these insights to guide sequence generation, shortlisting the relevant latent components that can steer the model toward desired targets such as zinc finger domains. This work contributes to the emerging field of mechanistic interpretability in biological sequence models, offering new perspectives on model steering for sequence design.

1 INTRODUCTION

Since the introduction of the transformer architecture (Vaswani, 2017), the capabilities of neural networks to model and generate natural language have increased dramatically. Yet, due to their black-box nature, we still lack a clear understanding of how these models achieve such capabilities (Rai et al., 2024). Recently, the mechanistic interpretability approach has been proposed, where researchers try to reverse engineer neural networks in a way similar to reverse engineering computer programs (Olah, 2022; Rai et al., 2024). This involves understanding which features the network is learning from our input data, and then how it performs operations with this set of features.

It has been observed that neural networks tend to encode high-level features as linear directions in their representation space—such as the gender direction in word embeddings (Park et al., 2023). Additionally, these models can store more facts and features than their parameter counts would suggest, a phenomenon known as superposition (Elhage et al., 2022). This phenomenon represents a core problem for interpretability: as a single neuron activation can be polysemantic and represent multiple features simultaneously.

Recently, sparse autoencoders (SAEs) have been proposed as a method to disentangle internal representations in language models, extracting features from superposition in an unsupervised manner (Templeton et al., 2024; Bricken et al., 2023; Cunningham et al., 2023). Notably, these features appear to be actionable: artificially activating them during inference can steer a model's output (Templeton et al., 2024; Makelov, 2024). Such methods have been successfully applied to language (Templeton et al., 2024; Cunningham et al., 2023; Gao et al., 2024), as well as vision and multimodal models (Gorton, 2024; Surkov et al., 2024), but biological and protein sequence models remain relatively unexplored (Simon & Zou, 2024; **?**).

Protein language models have been shown to encode structural, functional, and evolutionary information in their internal representations (Rives et al., 2019; Lin et al., 2023; Hayes et al., 2024). Interpretability methods for these models could reveal biological mechanisms, and support model debugging and editing for safety considerations. Additionally, model steering can be incorporated into sequence design pipelines.

The main contributions of this paper are:

• A trained sparse autoencoder (SAE) for the ESM-2 8M parameter model, along with potential interpretations of its latent components (sections 2.2, 3.1 and 3.3).

• A methodology for generating protein sequences by intervening on specific latents, demonstrating successful steering towards non-trivial features, such as zinc finger domains (section 3.4).

• A heuristic for selecting the model layer from which to extract representations using an intrinsic dimension estimator (section 3.2).

2 BACKGROUND

2.1 PROTEIN LANGUAGE MODELS

Many advancements in Natural Language Processing have been successfully applied to biological sequence modeling. Transformer-based neural networks can be trained on protein sequences using the Masked Language Modeling (MLM) task, where each amino acid is treated as a token that can be randomly masked. The model learns to predict the masked tokens by minimizing the following loss function (Rives et al., 2019):

$$\mathcal{L}_{\text{MLM}} = \mathbb{E}_{x\sim X}\mathbb{E}_M \sum_{i\in M} -\log p(x_i|x_{/M}) \tag{1}$$

where x is a protein sequence, M is a set of masked indices and $p(x_i|x_{/M})$ is the probability assigned to the ground truth amino acid x_i given its sequence context.

Training on the Masked Language Modeling (MLM) task forces the network to learn dependencies between masked amino acids and their sequence context while simultaneously capturing various biological features present in the data. Embeddings extracted from these models have been shown to encode information about secondary structure, tertiary contacts (residue-residue interactions), function, remote evolutionary relationships, and factors relevant to predicting mutational effects (Rives et al., 2019; Elnaggar et al., 2021; Meier et al., 2021; Lin et al., 2023; Hayes et al., 2024). On the other hand, the attention mechanism appears to prioritize binding sites, with attention maps capturing information about residue-residue interactions. (Vig et al., 2020).

2.2 SPARSE AUTOENCODERS

The sparse autoencoders used for interpretability are simple, single-layer models trained on the activations of a larger language model. To disentangle network features, the hidden layer is made significantly larger than the original embeddings, creating an overcomplete basis. A sparsity constraint is then applied to ensure that only a few latent neurons are active at a time, making the SAE's hidden representation far more interpretable than standard language model components (Templeton et al., 2024; Bricken et al., 2023; Cunningham et al., 2023).

2.2.1 ARCHITECTURE

The autoencoder is composed of an encoding and a decoding function, given by:

$$z = f_{enc}(x) = \text{ReLU}(W_{enc}(x - b_{dec}) + b_{enc}) \tag{2}$$

$$\check{x} = f_{dec}(z) = (W_{dec} \cdot z + b_{dec}) \tag{3}$$

Here f_{enc} is the encoder, that takes an embedded amino acid token $x \in \mathbb{R}^d$ from a given layer in the model and returns a latent $z \in \mathbb{R}^n_{\geq 0}$ with a hidden dimension n that is m times bigger that of the original vector (expansion factor). The decoder f_{dec} approximately reconstructs x given z, through the decoding matrix $W_{dec} \in \mathbb{R}^{n\times d}$ and the bias weight $b_{dec} \in \mathbb{R}^d$.

The loss function used for the training is a combination of the reconstruction error of the autoencoder $\mathcal{L}_{MSE}$ plus a sparsity constraint $\mathcal{L}_{L_1}$:

$$\mathcal{L}(x) = \mathcal{L}_{MSE} + \mathcal{L}_{L_1} = \sum_d (x_d - \check{x}_d)^2 + \lambda \sum_n z_n \tag{4}$$

While training, we renormalize the W_{dec} matrix to have unit norm after each backward pass. This is necessary to prevent that autoencoder latents become arbitrarily small and satisfy the L_1 constraint without actually being sparse.

3 METHODS

3.1 TRAINING DETAILS

We use the ESM-2 family of models (Lin et al., 2023) as our base, extracting activations from the final output of the transformer block. We train on approximately 15k non-redundant protein sequences from SCOPe 2.08 (Fox et al., 2014). Further details on the architecture, training procedures, and hyperparameter selection can be found in section A.2 of the appendix.

3.2 LAYER SELECTION

We adopt a principled strategy to select the layer from which we extract representations for the sparse autoencoder. The initial intuition, in line with earlier studies (Templeton et al., 2024; Gao et al., 2024), is to choose a mid-to-late layer, where the model is assumed to have developed abstract features but is not yet focused on the output reconstruction task. However, unlike these prior works, we move beyond mere intuition by incorporating a quantitative measure based on intrinsic dimension.

Specifically, we compute the intrinsic dimension of each layer's representations using the estimator proposed by (Facco et al., 2017), and then identify where this value plateaus. Previous research has shown that layers corresponding to local minima or plateaus in intrinsic dimension are where abstract information is most clearly encoded (Valeriani et al., 2024). Selecting a layer within this plateau increases the likelihood of capturing meaningful representations, providing a stronger foundation for interpretability and model steering.

3.3 INTERPRETING AUTOENCODER LATENTS

We extract protein annotations from the UniProt database (uni, 2025) and convert them into binary labels for each amino acid in the sequence. We then compute the precision π and recall ρ of each latent component k in detecting a given feature ϕ.

Let A be the set of all amino acids and A_{ϕ^+} the set of amino acids that have been annotated with the feature ϕ. Considering a latent k to be active (k^+) for a given amino acid when its value z_k exceeds a certain threshold τ_z, we have:

$$\pi = P(\phi^+|k^+) = \frac{\left|\left\{a \in A_{\phi^+} : z_k > \tau_z\right\}\right|}{\left|\{a \in A : z_k > \tau_z\}\right|} \tag{5}$$

$$\rho = P(k^+|\phi^+) = \frac{\left|\left\{a \in A_{\phi^+} : z_k > \tau_z\right\}\right|}{\left|A_{\phi^+}\right|} \tag{6}$$

This gives us a value of precision and recall for each pair of k, ϕ. We consider a latent component to be associated with a specific feature if its precision or recall exceed a predefined threshold that we set to 0.80.

3.4 GENERATING STEERED SEQUENCES

Once a latent corresponding to a specific feature is identified, we can steer the model during inference to increase the likelihood of generating protein sequences that contain that feature. This

approach, previously demonstrated in natural language models by (Templeton et al., 2024), is outlined in figure 1.

We begin with a randomly generated amino acid sequence of fixed length. After a forward pass through the model and the encoder layer of the SAE, we modify the target latent z_k by scaling and shifting its value to increase its magnitude (equation 7). We then pass the modified value z_k^* through the decoder layer f_{dec} and add back the original reconstruction error of the embedding x before passing it through the rest of the model (equation 8).

$$z_k^* = a \cdot z_k + b \tag{7}$$

$$x^* = f_{dec}(z_k^*) + x_{err} \tag{8}$$

For each position in the sequence, we randomly sample an amino acid according to the probability distribution predicted by the model under the intervention. We repeat this process starting from the predicted sequence and perform 100 iterations of inference-prediction to refine the sequence. We select the sequence at the iteration where the value of the activation z_k is maximum.

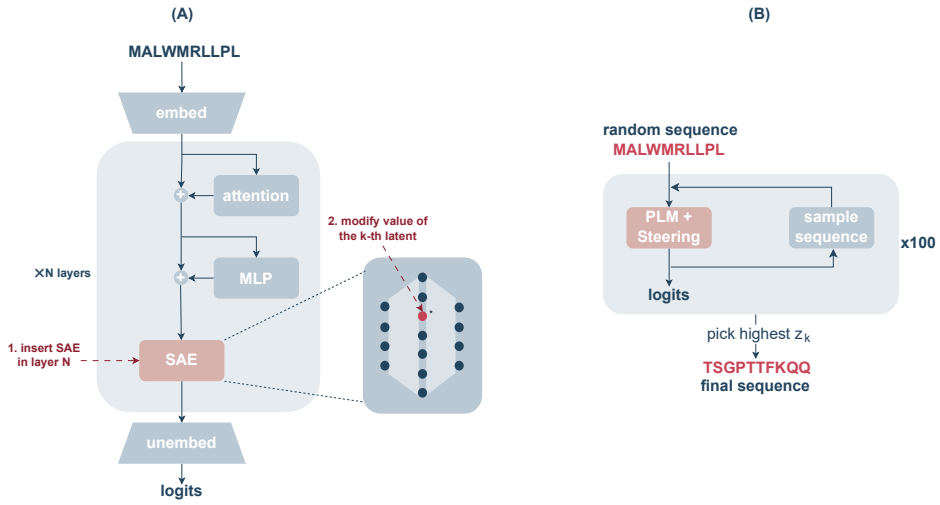

Figure 1: Sequence generation procedure. (A) To steer the model outputs, the base Protein Language Model is modified through the insertion of a sparse autoencoder in the residual stream, at a particular layer. During inference, the value of one of the latents in the autoencoder is modified. (B) Starting from a random sequence, we perform inference with the modified and intervened model, and sample a new sequence from the output logits. We repeat this procedure iteratively a certain number of times (i.e. 100), and at the end we retain the sequence which gives the highest value for the activation of the target latent z_k.

4 Results

4.1 Interpreting Latents

For the interpretability analysis, we focus on the autoencoder that provides the best trade-off between sparsity and reconstruction quality, as described in section A.4.2 of the appendix. We compute recall and precision for all [k, ϕ] pairs, following the methodology outlined in section 3.3, using three increasing thresholds of latent activation. This allows us to assess the robustness of the identified features.

We find 395 putative [k, ϕ] associations, detailed in table 1. Among these, there are latent components associated to different binding sites, cellular regions and motifs like zinc fingers. The complete set of latent - feature associations is available in the supplementary data (see section 5).

Table 1: Number of latent-feature annotation pairs with a minimum precision/recall of 0.8 for different values of the activation threshold τ_k

τ_k	**# Pairs (Precision $>$ 0.8)**	**# Pairs (Recall $>$ 0.8)**	**Total**
0.01	4	262	266
0.10	8	234	242
1.00	133	61	194
Total (unique)	133	262	395

We also identify many potential associations with a lower confidence (lower values of precision or recall). To get an idea of how many putative association are found for each value of precision/recall (as well as a combined F1-score) we plot their cumulative distributions in figure 2.

Intuitively, a latent component that perfectly matches an annotation type should exhibit both high precision and recall, resulting in a high F1-score. However, since the model is trained to optimize a masked language modeling loss, the features it learns may not directly align with those in the manually curated dataset. For instance, a latent k might encode a more specific subcategory of a dataset label ϕ, such as identifying the starting amino acid of a helix rather than the entire helix structure (as seen by **?** on some features). In such cases, the association between k and ϕ would likely have high precision but low recall.

Similarly, a high recall but low precision may indicate that the model has learned a more coarse-grained feature than those defined in the dataset. This is evident in cases such as latent $k = 610$, which activates across various types of zinc fingers, and latent $k = 555$, which responds to alpha-keto acids. To prevent selecting latents with high recall due to trivial reasons – such as activating indiscriminately on all amino acids – we also assess the proportion of times a latent is active on amino acids lacking a given label, denoted as $P(k^+|\phi^-)$, before confirming an association. In section A.4.3, we present examples of the distributions of $P(k^+|\phi^+)$ and $P(k^+|\phi^-)$ for a zinc finger region.

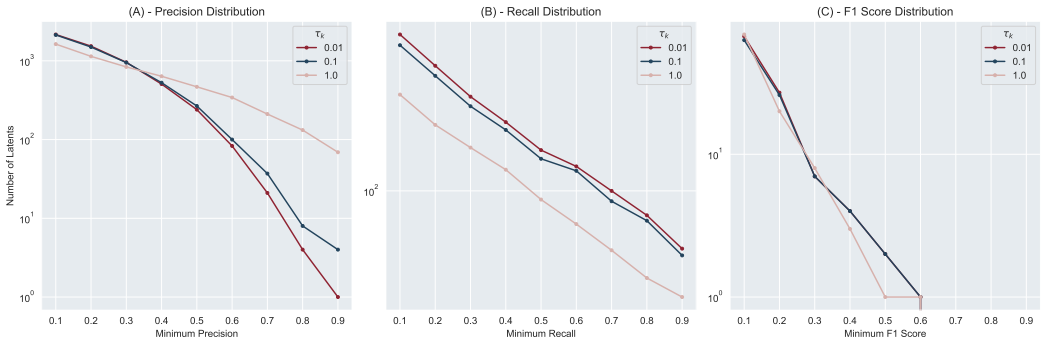

Figure 2: Distribution of the number of latent SAE components that detect a feature with a minimum value of precision, recall and F1-score. Setting a higher value for the activation threshold τ_k significantly increases the precision with which latents detect features, but it decreases the recall.

4.2 Sequence Generation

We test the sequence generation procedure with shortlisted latents with a clear association to the zinc finger region annotations (high recall in this specific case). We probe different values of sequence length ($[22, 27, 30, 35, 40, 60]$), of the scaling a ($[2, 5, 10, 20, 30]$) and shift b ($[0.1, 1, 10, 50, 100, 200]$), for a total of 180 combinations, obtaining one sequence for each.

Using an online tool for automatic annotation of zinc finger regions (Sathyaseelan et al., 2023), we look for known zinc finger motifs and Pfam family matches in our generated sequences. We find 24 matching regions (out of 180 sequences) when we simultaneously intervene on the two most prominent latents (highest recall) displaying a zinc finger association. In contrast, intervening on

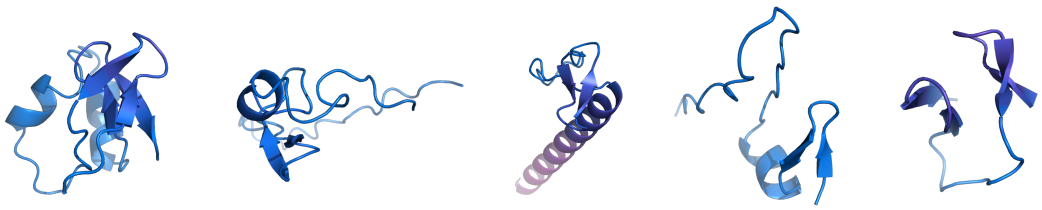

Figure 3: Examples of generated sequences subsequently folded with ESMFold (Lin et al., 2023). The sequences were generated while intervening on the model by increasing the value of latent components associated to the zinc finger motif. With the intervention, the model has a tendency to generate pairs of beta sheets in the vicinity of a helix, as in a typical zinc finger structure.

the most prominent latent produces only 3 matches, while generating sequences from the baseline model or intervening on a random latent or a random pair of latents produces no matches in any case. Among the matched sequences, the highest percent similarity was 48%, with an average sequence similarity of 31%, indicating a good level of diversity among the generated sequences.

While this sequence generation pipeline requires parameter fine-tuning to improve the success rate, the process can be automated by introducing appropriate heuristics to search the parameter space efficiently. To the best of our knowledge, this is the first application of steering with sparse autoencoder features to generate complex protein sequences, extending beyond trivial features like specific amino acids or simple amino acid repeats.

5 Discussion and conclusions

In this study, we demonstrate the potential of sparse autoencoders (SAEs) for interpreting and manipulating the internal representations of protein language models. By training a SAE on the ESM-2 8M parameter model, we identified and interpreted latent features associated with various protein annotations, including transmembrane regions, binding sites, and zinc finger motifs.

We have also demonstrated, for the first time, that these latent components can be leveraged to steer the model towards generating protein sequences with non-trivial structural features, like zinc finger motifs. The results highlight the utility of SAEs in disentangling the complex, polysemantic representations within protein language models, paving the way for more interpretable and controllable sequence generation. This approach not only deepens our understanding of how these models encode biological features but also opens up new possibilities for protein design and engineering. Future work could extend these methods to larger models and a wider range of protein features, further bridging the gap between interpretability and practical applications in computational biology.

Code & Data Availability

The weights for the trained sparse autoencoder model are available from huggingface at: https://huggingface.co/evillegasgarcia/sae_esm2_6_l3. The code is available from github at: https://github.com/edithvillegas/plm-sae. Supplementary data is available from zenodo at https://doi.org/10.5281/zenodo.14837817

Acknowledgments and disclosure of funding

A.A. was supported by the project “Supporto alla diagnosi di malattie rare tramite l’intelligenza artificiale”- CUP: F53C22001770002, and by the European Union – NextGenerationEU within the project PNRR ”PRP@CERIC” IR0000028 - Mission 4 Component 2 Investment 3.1 Action 3.1.1. E.N.V.G. was supported by the project PON “BIO Open Lab (BOL) - Rafforzamento del capitale umano”—CUP: J72F20000940007. We acknowledge the computational resources provided by the ORFEO supercomputing platform at AREA Science Park.

REFERENCES

Uniprot: the universal protein knowledgebase in 2025. *Nucleic Acids Research*, 53(D1):D609–D617, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html [Accessed: 2024].

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

Naomi K Fox, Steven E Brenner, and John-Marc Chandonia. Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures. *Nucleic acids research*, 42(D1):D304–D309, 2014.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. *arXiv preprint arXiv:2406.03662*, 2024.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *bioRxiv*, pp. 2024–07, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Aleksandar Makelov. Sparse autoencoders match supervised features for model steering on the ioi task. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in neural information processing systems*, 34:29287–29303, 2021.

Chris Olah. Mechanistic interpretability, variables, and the importance of interpretable bases, 2022. https://transformer-circuits.pub/2022/mech-interp-essay/index.html.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019. doi: 10.1101/622803. URL https://www.biorxiv.org/content/10.1101/622803v4.

Chakkarai Sathyaseelan, L Ponoop Prasad Patro, and Thenmalarchelvi Rathinavelan. Sequence patterns and hmm profiles to predict proteome wide zinc finger motifs. *Pattern Recognition*, 135: 109134, 2023.

Elana Simon and James Zou. Interplm: Discovering interpretable features in protein language models via sparse autoencoders. *bioRxiv*, pp. 2024–11, 2024.

Viacheslav Surkov, Chris Wendler, Mikhail Terekhov, Justin Deschenaux, Robert West, and Caglar Gulcehre. Unpacking sdxl turbo: Interpreting text-to-image models with sparse autoencoders. *arXiv preprint arXiv:2410.22366*, 2024.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet, 2024. https://transformer-circuits.pub/2024/scaling-monosemanticity/ [Accessed: 2024].

Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: Interpreting attention in protein language models. *arXiv preprint arXiv:2006.15222*, 2020.

A Appendix

A.1 Background

A.1.1 Protein Language Models

Many advancements in Natural Language Processing have been successfully applied to biological sequence modeling. Transformer-based neural networks can be trained on protein sequences using the Masked Language Modeling (MLM) task, where each amino acid is treated as a token that can be randomly masked. The model learns to predict the masked tokens by minimizing the following loss function (Rives et al., 2019):

$$\mathcal{L}_{\text{MLM}} = \mathbb{E}_{x \sim X} \mathbb{E}_M \sum_{i \in M} -\log p(x_i | x_{/M}) \tag{9}$$

where x is a protein sequence, M is a set of masking indices, and $p(x_i|x_{/M})$ is the probability assigned to the ground truth amino acid x_i given its sequence context.

Training on the Masked Language Modeling (MLM) task forces the network to learn dependencies between masked amino acids and their sequence context while simultaneously capturing various biological features present in the data. Embeddings extracted from these models have been shown to encode information about protein secondary structure, tertiary contacts (residue-residue interactions), function, remote evolutionary relationships, and factors relevant to predicting mutational effects (Rives et al., 2019; Elnaggar et al., 2021; Meier et al., 2021; Lin et al., 2023; Hayes et al., 2024).

On the other hand, the attention mechanism appears to prioritize binding sites, with attention maps capturing information about residue-residue interactions. (Vig et al., 2020).

A.2 SPARSE AUTOENCODER TRAINING

A.2.1 SPARSE AUTOENCODER ARCHITECTURE

The autoencoder is composed of an encoding and a decoding function, given by:

$$z = f_{enc}(x) = \text{ReLU}(W_{enc}(x - b_{dec}) + b_{enc}) \tag{10}$$

$$\check{x} = f_{dec}(z) = (W_{dec} \cdot z + b_{dec}) \tag{11}$$

Here f_{enc} is the encoder, that takes an embedded amino acid token $x \in \mathbb{R}^d$ from a given layer in the model and returns a latent $z \in \mathbb{R}^n_{\geq 0}$ with a hidden dimension n that is m times bigger that of the original vector (expansion factor). The decoder f_{dec} approximately reconstructs x given z, through the decoding matrix $W_{dec} \in \mathbb{R}^{n \times d}$ and the bias weight $b_{dec} \in \mathbb{R}^d$.

The loss function used for the training is a combination of the reconstruction error of the autoencoder $\mathcal{L}_{MSE}$ plus a sparsity constraint $\mathcal{L}_{L_1}$:

$$\mathcal{L}(x) = \mathcal{L}_{MSE} + \mathcal{L}_{L_1} = \sum_d (x_d - \check{x}_d)^2 + \lambda \sum_n z_n \tag{12}$$

While training, we renormalize the W_{dec} matrix to have unit norm after each backward pass. This is necessary to prevent that autoencoder latents become arbitrarily small and satisfy the L_1 constraint without actually being sparse.

A.2.2 TRAINING DATASET

We train our model using the Astral SCOPe 2.08 dataset, filtered to 40% sequence identity, which includes approximately 15k highly non-redundant protein sequences (Fox et al., 2014). This dataset provides a manageable number of tokens, enabling faster iteration over different hyperparameters while maintaining a diverse range of protein sequences and structural domains. This diversity allows the autoencoder to learn a broad spectrum of features.

A.2.3 HANDLING OF DEAD LATENTS

We check how frequently each of the latents activates over a subsample of tokens (50 batches of 4096 tokens) at regular intervals during training (every 500 batches). When this frequency is close to zero ($< 10^{-5}$), we consider that the latent is "dead" and we re-initialize its weights to "revive" it.

A.2.4 EVALUATION METRICS

To decide which hyperparameters (learning rate, sparsity penalty λ, SAE hidden size n) produce the best sparse autoencoder, we use the following metrics:

- L_0: The average number of non-zero components in the latent vector z for a given amino acid token. This is our measure of the sparsity level of the autoencoder.
- Number of dead latents: The number of components in the latent space that are never non-zero over a large number of sample tokens ($\sim 10^5$). This is a general metric for sparse autoencoders quality.
- Cross Entropy (CE) Increase: Difference between the average cross entropy loss of the original model and the cross entropy loss of the model when we substitute the activations in a given layer by the corresponding activations reconstructed by the autoencoder. This indicates how much of the model's performance the sparse autoencoder fails to reconstruct.

A.2.5 HYPERPARAMETER SELECTION

We perform a hyperparameter sweep with the following values: learning rate ($5e^{-4}$, $1e^{-4}$, $1e^{-3}$); L_1 penalty ($0.0003, 0.001, 0.005$) and dictionary size multiplier ($32, 10, 5$).

We use the evaluation metrics detailed in section A.2.4 to decide on the best combination of hyperparameters. Specifically, we aim to find a model that balances reconstruction error and sparsity by focusing on the "elbow" part of the CE increase against L_0 plot, where increasing the density of active latents in the latent space does not significantly reduce the CE increase, indicating an optimal trade-off.

A.3 Interpretability

A.3.1 Feature Annotation Data

As ground truth features for the interpretability analysis, we use the following protein annotations from Uniprot version 2024_1 :

- Transmembrane region
- Topological domain
- Binding site
- Zinc finger region
- Region of interest
- Intramembrane region
- Active site
- Disulfide bond
- Glycosylation site
- Helix
- Turn
- Strand

A.4 Additional Results

A.4.1 Intrinsic Dimension Analysis

We estimate the intrinsic dimension across all layers of ESM-2 8M following the methodology of (Valeriani et al., 2024). The resulting curve, shown in Figure 4, guides our decision to extract embeddings from layer 3. Larger models in the ESM-2 family exhibit a similar intrinsic dimension profile across different model sizes, with a more pronounced plateau region of relatively low intrinsic dimension in the same relative layer position.

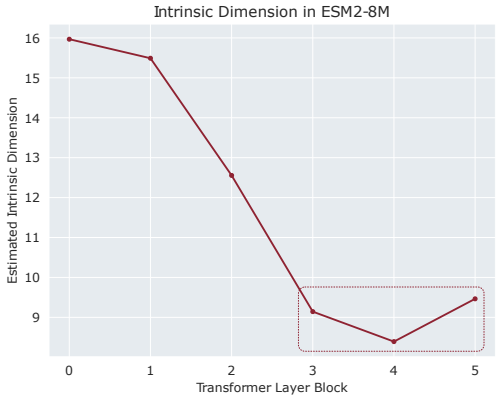

Figure 4: Evolution of the intrinsic dimension estimate through the layers of the ESM-2 8M model. We highlight the layers in the plateau/final ascent region.

A.4.2 SPARSE AUTOENCODER SELECTION

We evaluate all versions of the trained autoencoder primarily on two metrics: cross-entropy increase and sparsity (measured by L_0). These two goals are in conflict with each other, so we select what we think is a good compromise at the bend of the Pareto frontier (figure 5). The selected autoencoder has an average L_0 per amino acid of 18, and a cross-entropy increase of 0.10, with a hidden size that is 10 times larger than the original hidden size of the ESM-2 model. The number of dead latents in this SAE is 573. Decreasing the cross-entropy even more would entail a significant increase in activation density, which we want to avoid.

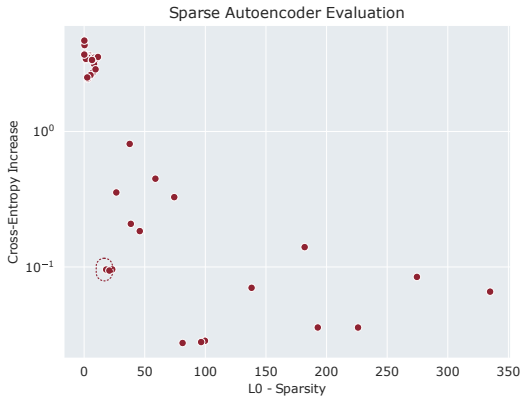

Figure 5: Cross-entropy increase vs sparsity trade-off for all the vanilla sparse autoencoders trained on layer 3 embeddings from ESM-2 8M. The selected autoencoder is indicated by a dashed circle.

A.4.3 INTERPRETING LATENTS - ACTIVATION PLOTS

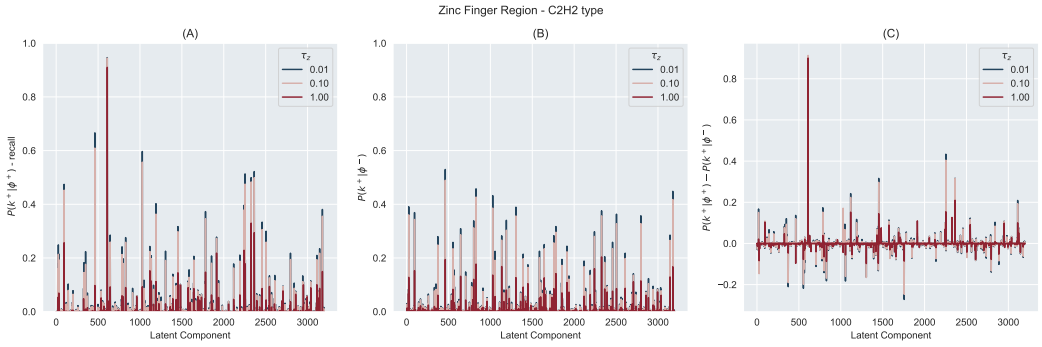

Figure 6: (A) $P(k^+|\phi^+)$ - Percentage of tokens for which each latent component is active when there is a C2H2 zinc finger type label, (B) $P(k^+|\phi^-)$ - percentage when there is no C2H2 zinc finger label, and (C) difference between these two values for three different activation thresholds τ_k. From the last plot, we see that there is a latent component that is prominently associated with the C2H2 zinc finger label.