Automated Neuron Labelling Enables Generative Steering and Interpretability in Protein Language Models

Arjun Banerjee [1] **David Martinez** [1] **Camille Dang** [1] **Ethan Tam** [1]

Abstract

Protein language models (PLMs) encode rich biological information, yet their internal neuron representations are poorly understood. We introduce the first automated framework for labeling every neuron in a PLM with biologically grounded natural language descriptions. Unlike prior approaches relying on sparse autoencoders or manual annotation, our method scales to hundreds of thousands of neurons, raveling individual neurons are selectively sensitive to diverse biochemical and structural properties. We then develop a novel neuron activation-guided steering method to generate proteins with desired traits, enabling convergence to target biochemical properties like molecular weight and instability index as well as secondary and tertiary structural motifs—including alpha helices and canonical Zinc Fingers. We finally show that analysis of labeled neurons in different model sizes reveals PLM scaling laws and a structured neuron space distribution.

1. Introduction

Protein language models (PLMs) have transformed biological sequence modeling, enabling breakthroughs in protein structure prediction, function annotation, and design (Rao et al., 2021; Lin et al., 2022; Ruffolo & Madani, 2024). Despite the empirical successes of models like ESM-2 (Lin et al., 2022) and ProtTrans (Elnaggar et al., 2022), the internal representations of these models remain opaque, making it difficult to understand how specific features of protein sequences are represented. This lack of interpretability poses barriers to rigorous analysis of model knowledge and hinders the ability for researchers to generate proteins with specific features, which has long been a goal of De Novo Protein design.

Early studies in PLM interpretability relied largely on heuristic probing, which showed that PLM attention heads encoded structural information about proteins (Vig et al., 2020; Rao et al., 2021) and are able to identify functional sites like allosteric residues (Kannan et al., 2024; Dong et al., 2024). Going beyond heuristic probing, recent work have leveraged sparse autoencoders (SAEs) to show that sparse latents in PLMs can capture binding sites, structural motifs, and functional domains that can be used for steering (Simon & Zou, 2024; Adams et al., 2025; Parsan et al., 2025). However, SAEs require training a new model on top of the PLM, which can introduce optimization instability, architecture-specific biases, and sensitivity to initialization, which all hinder effectiveness (Kantamneni et al., 2025; Chaudhary & Geiger, 2024; Farrell et al., 2024).

In parallel, neuron-level labeling has emerged as a promising approach for interpreting model internals, with early methods showing that neurons in vision models capture human-aligned features (Bau et al., 2017). Recent work has extended labelling by using natural language descriptors to GPT-2 (Bills et al., 2023) and introducing high-fidelty frameworks that combines exemplar mining, simulator-based scoring, and distillation of high-quality neuron explanations across an entire model (Choi et al., 2024).

1.1. Contributions

We apply hidden-unit labeling to PLMs, which enables 2 novel contributions:

1. **Neuron Understanding:** We identify individual neurons that encode broad physicochemical properties—such as charge and hydrophobicity— and distinct structural motifs like zinc fingers or supercoils. We then explore scaling laws and neuron representations.

2. **Text2Protein Generation:** We leverage LLM-guided neuron queries to generate protein sequences with highly specific features. We show convergence of characteristics (GRAVY, weight, etc.) and both secondary and tertiary structure.

[*]Equal contribution [1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, California. Correspondence to: Arjun Banerjee <abaner@berkeley.edu>.

Proceedings of the Workshop on Generative AI for Biology at the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

2. Methods

At a high level, our problem is to label each neuron in a protein language model with the biophysical feature they are associated with. Then, we seek to use these labels to steer proteins towards features we desire.[1]

2.1. Problem Setup

More formally, let $\phi : \mathcal{X} \to \mathbb{R}$ denote a scalar-valued neuron feature defined over protein sequences $x \in \mathcal{X}$, extracted from a fixed layer of a pretrained protein language model (PLM). Each sequence x_i is associated with a feature vector $f_i \in \mathcal{F}$ representing structured biological annotations.

Our goal is to produce a natural language description $h \in \mathcal{H}$ that accurately summarizes the conditions under which $\phi(x)$ exhibits strong activation.

Once we obtain a natural language description $h \in \mathcal{H}$ for each neuron feature ϕ, we leverage these interpretable neuron-level labels to guide protein design and analysis. Specifically, given a target biological property $\tau \in \mathcal{F}$, we identify neurons ϕ whose descriptions h indicate sensitivity to τ, and modulate their activations within the PLM to bias outputs toward sequences with enhanced expression of τ. This enables gradient-free, concept-level interventions on protein sequences, using the neuron basis as a control interface for steering model behavior in alignment with biophysical objectives.

2.2. Model

To achieve this, we create the following pipeline as shown in Figure 1:

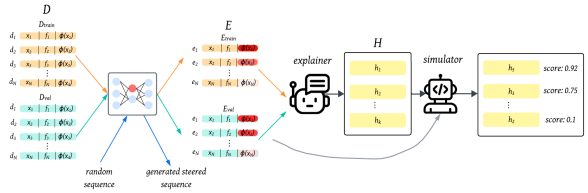

Figure 1: Pipeline for Automatic Neuron Labelling

1. **Dataset Construction:** We create a dataset $\mathcal{D} = \{(x_i, f_i, \phi(x_i))\}_{i=1}^{n}$, $\phi(x_i) \in \mathcal{R}^+$ is the normalized activation of neuron ϕ on a protein sequence x_i. .

2. **Explainer Model:** We create an explainer function $E : \mathcal{D}_{\text{top}} \subset \mathcal{D} \to \mathcal{H}$ which maps a subset of top-activating examples to a hypothesis h in natural language.

3. **Simulator Model:** We create a simulator $S : \mathcal{H} \times \mathcal{X} \times \mathcal{F} \to [0, 10]$ which predicts a discretized activation score $\hat{\phi}(x)$ given a description h, a sequence x, and its features f. The goal of this simulator is to find the hypothesis that maximizes the alignment of the hypothesis with the observed activation: $r(h) = \text{Corr}\left(S(h, x_j, f_j), \phi(x_j)\right), \quad \text{for } (x_j, f_j) \in \mathcal{D}_{\text{val}}$, where $\text{Corr}(\cdot, \cdot)$ denotes the Pearson correlation coefficient.

4. **Generator:** We aim to generate protein sequences $x \in \mathcal{X}$ that maximize the activation of a subset of neurons $\{z_k\}$ in a fixed layer ℓ of a pretrained protein language model f. These neurons are selected based on their known association with a desired biological property. Starting from a randomly initialized sequence, we iteratively apply masked inpainting: at each step, we randomly mask a fraction of residues in x, run the masked sequence through f, and apply an affine intervention $z_k^* = a \cdot z_k(x) + b$ to the selected neuron activations via forward hooks. The modified activations are then propagated through the model to update output logits, and the masked tokens are resampled from the softmax distribution.

Dataset Construction: We initially randomly sample $500,000$ protein sequences under 1024 amino acids from the UniProt Dataset (Consortium, 2023), which includes manually labeled qualitative and quantitative descriptors (such as sequence features and functional annotations). We use these descriptors as the features that describe a given protein; we enrich these textual descriptors with additional quantitative data using the `BioPython` (Cock et al., 2009) and `Modlamp` (Müller et al., 2017) libraries [2]. A complete list of features and sequence dataset analysis can be found in Appendix A.

We then place forward hooks at each linear layer of the ESM model and record how much each full protein sequence activates each neuron. We store the sequences that cause the most and least relative activation per neuron.

Explainer Model: To generate concise natural language explanations for each neuron, we use a prompt-based method that asks a language model to analyze the biological patterns shared by the sequences that most strongly activate that neuron. For each neuron, we collect a list of the k top-activating protein sequences along with their associated quantitative and qualitative features. These are formatted into a structured prompt that instructs the model to holistically generalize the key shared biological traits across the examples.

We use OpenAI's GPT-4.1-nano as our explainer model and sample m multiple candidate explanations per neuron

[1] All code is available at: https://github.com/arjun-banerjee/PLMNeuron

[2] Dataset at: https://huggingface.co/datasets/protolyze/plminterp

using a high-temperature ($T = 0.90$) setting to encourages diverse hypotheses and allow the model to explore multiple plausible generalizations. Each prompt explicitly instructs the model to identify only stable, non-varying features, to avoid redundancy, and to generate a single, concise sentence without introductory phrases. The prompt can be found in Appendix B

Simulator Model: To evaluate which hypotheses best explain a neuron's behavior, we train a LLM that performs in-context prediction of activation levels. The simulator takes as input a hypothesis, a protein sequence, and its associated biological features, and predicts a discrete activation value from 0 to 10. The hypothesis that best predicts the real activation is considered to be the most accurate.

We fine-tune Longformer to serve as the simulator due to its ability to handle long context lengths (Beltagy et al., 2020). Each input example is formatted as a structured natural language prompt that includes the hypothesis, followed by the amino acid sequence and a dictionary of feature-value pairs. The target is the neuron's normalized activation value, bucketed into one of 11 classes – which was empirically found to be optimal for analyzing activations.

We then perform gradient descent on the cross-entropy loss between the model's predicted class distribution and the ground truth activation bucket. This directly trains the model to use the hypothesis in-context to predict how strongly a neuron will activate on a given input. At evaluation time, we apply the simulator to held-out sequences and compute the Pearson correlation between its predicted scores and the true activations. This correlation score is used to rank hypotheses and select the best explanation for each neuron. The prompts for this can be found in .

Generator: To understand which neurons should be active for a given feature, we query an LLM to assess whether a given label indicates association with the feature. The prompt for this is in .

We then follow a similar method to Garcia 2025 (Garcia & Ansuini, 2025), but we operate on the neuron level and using the ESM tokenizer rather an an encoder/decoder. Beginning with a randomly initialized sequence, we mask a random set of amino acids and forward it through the model to extract hidden representations and identify the activation of the target neuron. An affine transformation ($Ax + B$) is then applied to increase the neuron's activation magnitude, after which the modified hidden state is propagated through the remainder of the model to produce updated output logits. A new set of sampled tokens are sampled from the resulting distribution and decoded via the tokenizer to produce a sequence. This procedure is iterated for a fixed number of steps, and the sequence yielding the highest observed activation is retained. In doing so, we perform a form of activation-guided sequence optimization directly in neuron space.

3. Generation Results

We explore our generation mechanism on both characteristic and structural guidance schemes with 2 differently sized models: `esm2_t6_8M` and `esm2_t12_35M`.

3.1. Characteristic Guidance

3.1.1. Single-Characteristic Steering

To assess characteristic guidance, we activate neurons associated with specific biochemical properties via the previously discussed generation loop. We focus on single-feature steering, in which we select neurons associated with a target descriptor τ and iteratively steer a randomly initialized sequence to enhance or suppress the corresponding property.

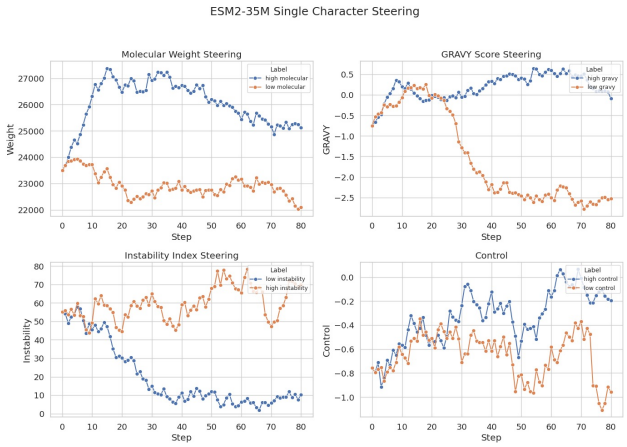

Figure 2: Steering with `esm2_t12_35M`, A = 10 and B = 3

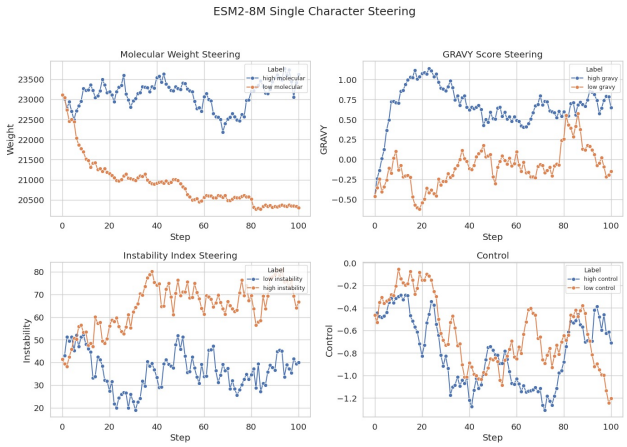

Figure 3: Steering with `esm2_t6_8M`, A = 200 and B = 10

We evaluate the effectiveness of this intervention across three biochemical metrics: molecular weight, GRAVY score, and instability index. For each property, we conduct two experiments—steering toward a "high" and "low" variant of the feature—using independently annotated neurons identified via semantic search. As seen in Figure 2 and Figure 3, we observe consistent and monotonic divergence between the "high" and "low" trajectories. For instance, sequences steered toward higher molecular weight exhibit a clear upward trend, while those steered toward lower weight decline generally steadily. Similar patterns emerge for GRAVY score and instability index.

To validate that these changes are due to semantically meaningful interventions, we include a control condition where neurons not associated with any biological property are randomly selected. As expected, the control experiments show no random change in the target metrics in an unsteered manner, confirming that the observed effects are driven by the neuron labels and not by random perturbations.

3.1.2. Probing Characteristic Representation Space

The existence of neurons labeled with semantically opposing concepts (e.g., 'High τ' vs. 'Low τ') lead to the possible hypothesis that they may encode inverse directions in the model's representational space. To understand the semantic relationships between such oppositely labeled neurons, we hypothesize that multiplying a 'Low τ' neuron by a negative value should invert its functional effect—producing a result akin to amplifying a 'High τ' neuron. To test this, we introduce *negative steering*, where we multiply the selected neurons by negative A and B values, as shown in Figure 4

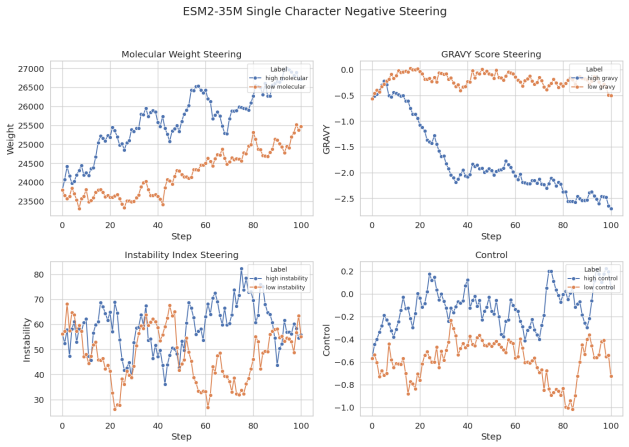

Figure 4: Steering with `esm2_t12_35M`, A = -10 and B = -5

We find that the negative steering relationship is highly feature dependent; for instance, in the case of GRAVY score, negative steering led to the complete inverse result of positive steering, while in the case of instability index it led to no conclusive direction changes. Interestingly, in the case of molecular weight, negative steering of low molecular weight and negative steering of high molecular weight *both* increased the weight.

3.2. Structural Guidance

3.2.1. Secondary Structure Guidance

We start by testing the ability of our generator to develop generalized secondary structure by steering towards alpha helices (semantically searching "alpha") and beta strands (semantically searching "beta") starting from a neutral sequence of 75 D-configuration amino acids, respectively. The results of these generations can be seen in Figure 5

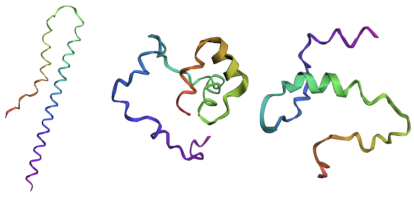

Figure 5: (Left): Visualization of the starting protein; (Middle): Visualization of the alpha helix steered protein; (Right): Visualization of the beta helix steered protein

To label the regions of the generated steers, the sequences were fed into PsiPred (McGuffin et al., 2000) and their resultant domains were annotated with C (to denote coils), S (to denote beta strands), or H (to denote alpha helices). As depicted in Figure 6, the alpha steered model contains 5 distinct alpha helix domains spanning 34% of the protein, while the beta steered model contains 4 distinct beta strand domains. Both proteins also contain the opposing secondary structure motifs (there are 3 beta strand domains in the alpha protein and 1 alpha helix in the beta domain); this result is not surprising, as there are many neurons that are labelled as controlling both alpha and beta secondary structures.

While these results demonstrate the generators ability to converge on the desired secondary structures, it is important to note that these structures are likely not actualize in the laboratory. The alpha and beta steered proteins have a ΔG value of 99.64 and 113.05 kCal/mol, respectively, which is far above the typical 5-15 kCal/mol range (Ahmad, 2022). A more complete analysis of the generated proteins can be found in Appendix C, and a discussion of how to improve viability can be found in the Future Works section.

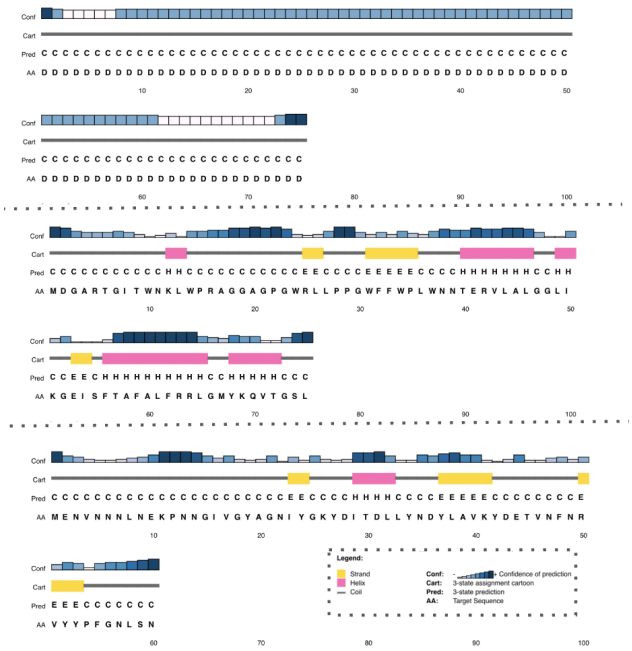

Figure 6: PsiPred outputs of the neutral sequence (top), the alpha steered sequence (middle), and the beta steered seqeunce (bottom)

3.2.2. Tertiary Structure Guidance

To test the ability of our generator to develop tertiary structure, we generate proteins steered towards neurons containing "zinc finger." We were able to observe generations with this characteristic: in Figure 7, we display a generated protein with a canonical C2H2 zinc finger motif, characterized by the pattern Cys-X(2–4)-Cys-X(12)-His-X(3–5)-His.

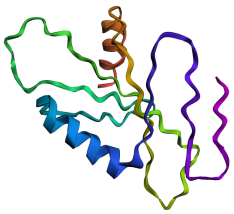

Figure 7: Visualization of a Generated Zinc Finger Motif

Of the 100 sequences we generated, this was the only fully correct sequences we observed; however, many other sequences contained near hits to Zinc Finger characteristics. One protein generated a "HCCCCACACF" subsequence, which is very similar to the RING finger domain. Another sequence contained a C-X3-C-X20-C-X4-C domain, which resembles a C4 motif.

We predict that using this approach on a model with more hidden units, like the 15 billion variant, would yield more rich structural information which would enable more accurate generations. However, it is important to note that this protein is likely not able to viably fold given it's ΔG value of 172.49; more about this is discussed in Appendix C.

4. Interpretability Results

We label all the neurons of 3 differently sized models: `esm2_t36_8B_UR50D`, `esm2_t12_8B_UR50D`, and `esm2_t36_8B_UR50D`. A random selection of neuron labels across the 12 layers of `esm2_t12_8B_UR50D` can be seen in Table 1. [3]

Neuron	Description
(0, 160)	Strongly activates for secreted proteins with low to negative GRAVY scores.
(1, 323)	Strongly activates for flagellin proteins involved in bacterial flagellum structure.
(4, 204)	Strongly activates for proteins with high charge at pH 7 and a significant fraction of beta-sheet structure.
(7, 467)	Strongly activates for proteins with tryptophan synthase activity and negative gravy scores.
(9, 437)	Strongly activates for proteins with a specific role in DNA replication initiation and regulation.
(11, 473)	Strongly activates for chloroplastic proteins involved in RNA binding and processing.

Table 1: Neuron-level activation descriptions. Each neuron is identified by an ordered pair (ℓ, n), where ℓ is the layer index and n is the neuron index within that layer.

4.0.1. Toward a PLM Scaling Law

Preliminary analysis of the labels suggested that bigger models are able to capture more fine-grained protein features; for instance, Figure 8 demonstrates that the 3B model is able to capture structural details *hands, supercoiling,* and *fingers*, whilst the 8M model is only able to capture *fingers*. As such, we hypothesize that bigger PLMs are able to capture more niche elements of a proteins composition rather than just adding more parameters that represent generic features.

Another set of phenomena we observed is the late-specialization of functional features in smaller models, possibly attributed to information bottlenecks. In other words, early feed-forward layers in smaller models function as

[3] All labels available at https://huggingface.co/protolyze/datasets

universal encoders, while the last layers are largely discriminative on the set of learned embeddings from earlier layers. In layers S1–S5, ESM-8M exhibits only a trickle of function-related signal; instead, these layers are likely devoted to extracting generalizable sequence patterns—basic amino-acid correlations, local motifs, and low-level statistical features—that will be broadly useful downstream. We observe the functionality shift from encoding to discriminating features in an incremental manner by the roughly linear increase in neuron-specific attribution. In addition, observe how increasing the parameter size leads to greater frequency of feature-attributed neurons, suggesting that the model shifts from an encoder-discriminator model to a hierarchal model which expends more layers to learn richer and finer-grain features, which is supported by the emergence of hand features in the ESM-3B model in Figure 8. In smaller models, this is not feasible because the tight information bottleneck in early layers collapsed diverse feature information into a low-dimensional code, leaving insufficient representational capacity to carve out the fine-grained detectors necessary for richer features like hands.

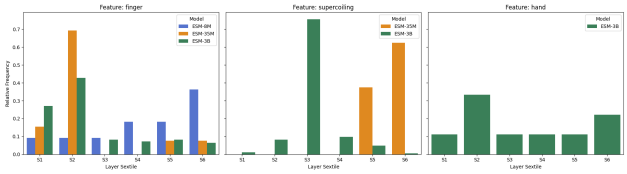

Figure 8: Relative occurance of structural features for each ESM variant labeled by layer sextile. Certain structural features fail to be captured by smaller models. Sextiles were used to ensure accurate relative comparisons between models, as the 8M variant is 6 layers and all other model's layers are divisible by 6.

We speculate that this behavior stems from the limited parameter budget, which forces the model to spread feature attribution broadly across neurons and inhibits clear specialization. Consequently, early layers operate as a general-purpose encoder: their activations simultaneously support the original PLM training objective of masked-amino-acid prediction while enabling implicit downstream functional protein characteristics to form. Unraveling how information becomes entangled across neuron populations—and how this entanglement influences specialization—remains an important avenue for future investigation.

4.0.2. Feature Locations

Table 1 suggests that lower-layer neurons capture local biochemical features like charge and GRAVY score (e.g., neuron (0, 160)). Middle layers begin to detect more complex patterns such as secondary structure and domain-like segments (e.g., neurons (4, 204) and (7, 467)), while higher layers abstract to functional roles like DNA replication or RNA processing (e.g., neurons (9, 437) and (11, 473)). This progression seems to reflect the typical hierarchical abstraction in protein language models.

To confirm this hypothesis, we sample 3 characteristics associated with functional roles (*repair*, *recombination*, and *replication*), 3 characteristics associated with structural roles (*sheet*, *alpha*, and *beta*) and 3 characteristics associated with sequence-derived properties (*charge*, *hydrophobicity*, *instability*). We semantically search all the neurons associated with this keyword, and then plot their relative distributions in Figure 9 for each of the model variants.

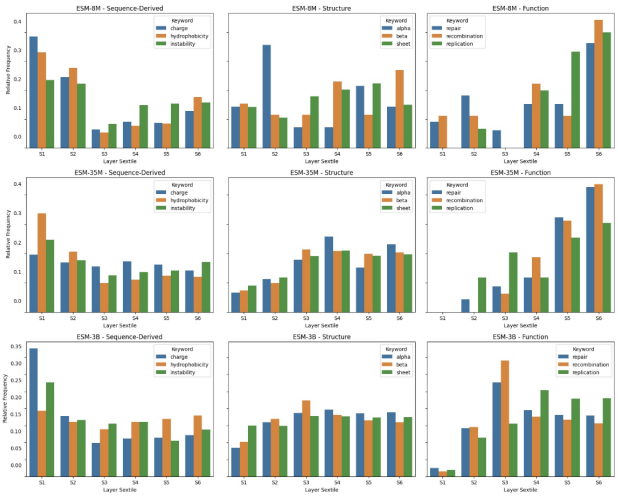

Figure 9: Relative occurance of each property (functional, structural, sequence derived) for each ESM variant labeled by layer sextile. Sextiles were used to ensure accurate relative comparisons between models, as the 8M variant is 6 layers and all other model's layers are divisible by 6.

The figure above demonstrates how different categories of protein features, from sequence-derived to structural to functional features, are encoded in layers of the PLM. All model sizes demonstrate some degree of hierarchical encoding. Starting with local biochemical properties / sequence-derived features (charge, hydrophobicity, instability), we can see that its representation peaks in lower layers of the models, and drops in higher sextiles, which is consistent with low-level local feature encoding in earlier layers. This aligns with the intuition that local-level features such as biochemical properties can often be inferred with just

Structural properties, as shown in the middle column, peak slightly in the middle layers, but persist through the later layers, which reflect an increasing context window that is required to infer secondary and tertiary structural patterns, which often span multiple residues but not the full protein. And on the rightmost column, we can see that functional

features have the strongest concentration in later sextiles, particularly in the ESM-8M and ESM-35M models, showing that functional properties require global context and tend to emerge in deeper layers of the network. This progression reflects the pattern typical of language models where shallow layers tend to encode local features, and as layer depth increases,

In ESM-8M, we can see that the model's layers appear to have a more distinct specialization, such as the sequence-derived features sharply peaking at the first sextile and then quickly falling off after, or functional-related neurons concentrated in the final sextile, which shows that the model delays global / semantic processing until the end.

Meanwhile, as the model size increases, the distributions become more distributed, For example, functional features show up in earlier sextiles in ESM-3B as compared to the other two smaller models, indicating greater distribution across the model.

5. Looking Forwards

5.1. Limitations and Future Work

LLM Oversimplification: Despite providing comprehensive prompts and rich feature sets, the explainer model (GPT-4.1-nano) may still generate oversimplified or overly generic explanations, reflecting limitations in the model's ability to synthesize complex biological patterns. Additionally, the model might also hallucinate features due to high temperatures, despite the attempts of the simulator to mitigate this. To assess neuron description quality in the future, it would be useful to generate an evaluation rubric and benchmark for descriptions and then have humans assess the quality of said labels.

Structural Viability: While our generator can produce proteins exhibiting desired characteristics, it does not guarantee that the resulting sequences will fold into viable, stable structures. To address this, we aim to incorporate a reinforcement learning module into our loop where a structure prediction model (e.g., AlphaFold2 or ESMFold) acts as an oracle to evaluate the foldability of generated sequences. The feedback from this model serves as a reward signal, encouraging the generator to produce sequences that are not only functionally relevant but also structurally plausible. Future work could compare generated sequences to their nearest real neighbors and incorporate Δ G scores for a more robust assessment.

Expand Labels: Currently, we only label 3 different models and use one LLM due to compute limitations. We plan to extend neuron labeling beyond the three models used in this work to the full suite of ESM models, covering a broader range of architectures and sizes. Additionally, we aim to use multiple large language models and diverse prompting strategies to generate more robust and varied natural language labels. This will help reduce bias from any single LLM and better capture the functional diversity of neurons across models.

Model Pruning: We seek to explore pruning neurons based on their assigned labels, focusing on those with low specificity, high redundancy, or minimal activation. By removing such neurons, we aim to reduce model size and inference cost while retaining the core functional capacity of the model. This approach can support interpretable model compression and shed light on which neurons are essential for biological representation. We hope this will generate a smaller condense variant of ESM with the same performance.

5.2. Conclusion

We introduce a comprehensive framework for interpreting individual neurons in protein language models through natural language explanations, simulation-based validation, and generative steering. By leveraging both qualitative annotations from UniProtKB and computed biochemical features, we provide a scalable and biologically grounded method for neuron labeling in ESM2. Our explainer model generates concise hypotheses that capture the biological patterns associated with each neuron, while our simulator quantitatively evaluates the quality of these hypotheses through in-context prediction. We demonstrate that these labeled neurons can be used to steer sequence generation toward targeted biophysical properties and structural motifs, enabling new modes of interaction with pretrained protein models. We lastly analyze the labels, suggesting elementary scaling laws and analyze feature locations.

Together, these contributions establish a foundation for mechanistically understanding and manipulating PLMs at the neuron level. Future work can further integrate structural validation, extend to other protein models such as AlphaFold and RoseTTAFold, and explore higher-order compositional representations.

Additionally, our generative steering framework lays the groundwork for simultaneously modulating multiple biological properties within a single protein sequence. This multi-objective control could be especially valuable in therapeutic protein design, where optimizing for multiple criteria such as stability and efficacy is essential. By enabling fine-grained, neuron-level manipulation of pre-trained models, we move closer to controllable and interpretable protein generation pipelines that align with practical needs in drug discovery and synthetic biology.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

The authors thank Anant Sahai, Naman Jain, and Qiyang Li for feedback in the early stages of the project, and for insightfully teaching the class that made this possible. We are also grateful to Kiran Suresh, Amos You, and Eshaan Moorjani for introducing us to neuron labeling work and helpful discussions regarding scaling.

Code and Data Availability Statement

All code can be found publicly on Github at https://github.com/arjun-banerjee/PLMNeuron. All datasets used and labels generated can be found at https://huggingface.co/protolyze/datasets.

References

Adams, E., Bai, L., Lee, M., Yu, Y., and AlQuraishi, M. From mechanistic interpretability to mechanistic biology: Training, evaluating, and interpreting sparse autoencoders on protein language models. *bioRxiv*, 2025. doi: 10.1101/2025.02.06.636901. URL https://www.biorxiv.org/content/early/2025/02/08/2025.02.06.636901.

Ahmad, F. Protein stability determination problems. *Frontiers in Molecular Biosciences*, 9:880358, Aug 2022. doi: 10.3389/fmolb.2022.880358.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations, 2017. URL https://arxiv.org/abs/1704.05796. First two authors contributed equally. Oral presentation at CVPR 2017; Subjects: Computer Vision and Pattern Recognition (cs.CV); Artificial Intelligence (cs.AI); ACM classes: I.2.10.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. URL https://doi.org/10.48550/arXiv.2004.05150. Version 2 introduces the Longformer-Encoder-Decoder (LED) model.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

Chaudhary, M. and Geiger, A. Evaluating open-source sparse autoencoders on disentangling factual knowledge in gpt-2 small, 2024. URL https://arxiv.org/abs/2409.04478. Subjects: Machine Learning (cs.LG); Artificial Intelligence (cs.AI); Neural and Evolutionary Computing (cs.NE). arXiv preprint arXiv:2409.04478.

Choi, D., Huang, V., Meng, K., Johnson, D. D., Steinhardt, J., and Schwettmann, S. Scaling automatic neuron description. *Transluce AI*, 2024.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

Consortium, T. U. Uniprot: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 2023. doi: 10.1093/nar/gkac1052. URL https://academic.oup.com/nar/article/51/D1/D523/6835362.

Dong, T., Kan, C., Devkota, K., and Singh, R. Allo-allo: Data-efficient prediction of allosteric sites. *bioRxiv*, 2024. doi: 10.1101/2024.09.28.615583.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., and Rost, B. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, Oct 2022. doi: 10.1109/TPAMI.2021.3095381. Epub 2022 Sep 14. PMID: 34232869.

Farrell, E., Lau, Y.-T., and Conmy, A. Applying sparse autoencoders to unlearn knowledge in language models, 2024. URL https://arxiv.org/abs/2410.19278. arXiv preprint arXiv:2410.19278 (v2, updated Nov 2, 2024); Subjects: Machine Learning (cs.LG); Artificial Intelligence (cs.AI).

Garcia, E. N. V. and Ansuini, A. Interpreting and steering protein language models through sparse autoencoders, 2025. URL https://arxiv.org/abs/2502.09135.

Guerois, R., Nielsen, J. E., and Serrano, L. Predicting changes in the stability of proteins and protein complexes: A study of more than 1000 mutations. *Journal of Molecular Biology*, 320(2):369–387, 2002. ISSN 0022-2836. doi: https://doi.org/10.1016/S0022-2836(02)00442-4.

URL https://www.sciencedirect.com/science/article/pii/S0022283602004424.

Kannan, G. R., Hie, B. L., and Kim, P. S. Single-sequence, structure-free allosteric residue prediction with protein language models. *bioRxiv*, 2024. doi: 10.1101/2024.10.03.616547. URL https://www.biorxiv.org/content/10.1101/2024.10.03.616547v1.

Kantamneni, S., Engels, J., Rajamanoharan, S., Tegmark, M., and Nanda, N. Are sparse autoencoders useful? a case study in sparse probing, 2025. URL https://arxiv.org/abs/2502.16681. arXiv preprint arXiv:2502.16681.

Lin, Z., Akin, H., Rao, R., and et al. Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv*, 2022. doi: 10.1101/2022.07.20.500902. URL https://www.biorxiv.org/content/early/2022/10/31/2022.07.20.500902.

McGuffin, L. J., Bryson, K., and Jones, D. T. The psipred protein structure prediction server. *Bioinformatics*, 16(4): 404–405, April 2000. doi: 10.1093/bioinformatics/16.4.404.

Müller, A. T., Gabernet, G., Hiss, J. A., and Schneider, G. modlAMP: Python for antimicrobial peptides. *Bioinformatics*, 33(17):2753–2755, September 2017. doi: 10.1093/bioinformatics/btx285. URL https://academic.oup.com/bioinformatics/article/33/17/2753/3796392.

Parsan, N., Yang, D. J., and Yang, J. J. Towards interpretable protein structure prediction with sparse autoencoders. *arXiv preprint arXiv:2503.08764*, 2025. URL https://arxiv.org/abs/2503.08764.

Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., and Rives, A. Transformer protein language models are unsupervised structure learners. *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=fylclEqgvgd.

Ruffolo, J. A. and Madani, A. Designing proteins with language models. *Nature Biotechnology*, 42(2):200–202, February 2024. doi: 10.1038/s41587-024-02123-4. URL https://www.nature.com/articles/s41587-024-02123-4.

Simon, E. and Zou, J. Interplm: Discovering interpretable features in protein language models via sparse autoencoders. *bioRxiv*, 2024. doi: 10.1101/2024.11.14.623632. URL https://www.biorxiv.org/content/10.1101/2024.11.14.623632v1.

Vig, J., Madani, A., Varshney, L. R., Xiong, C., Socher, R., and Rajani, N. F. Bertology meets biology: Interpreting attention in protein language models, 2020. URL https://arxiv.org/abs/2006.15222.

A. Dataset

We used 500,000 sequences along with qualitative annotations from the UniProtKB dataset (Consortium, 2023). We used the following features: subcellular location, Gene Ontology (biological process and molecular function), catalytic activity, pathway, enzyme commission (EC) number, disruption phenotype, induction, domain, and functional descriptions. In addition, we computed quantitative biochemical features using the `BioPython` package, including sequence length, molecular weight, isoelectric point, aromaticity, instability index, GRAVY score, and predicted secondary structure fractions (helix, turn, sheet). We also used the `modlAMP` package to calculate charge at pH 7, Boman index, aliphatic index, and hydrophobic moment. To understand the distribution of the dataset we sampled from, refer to Figure 10.

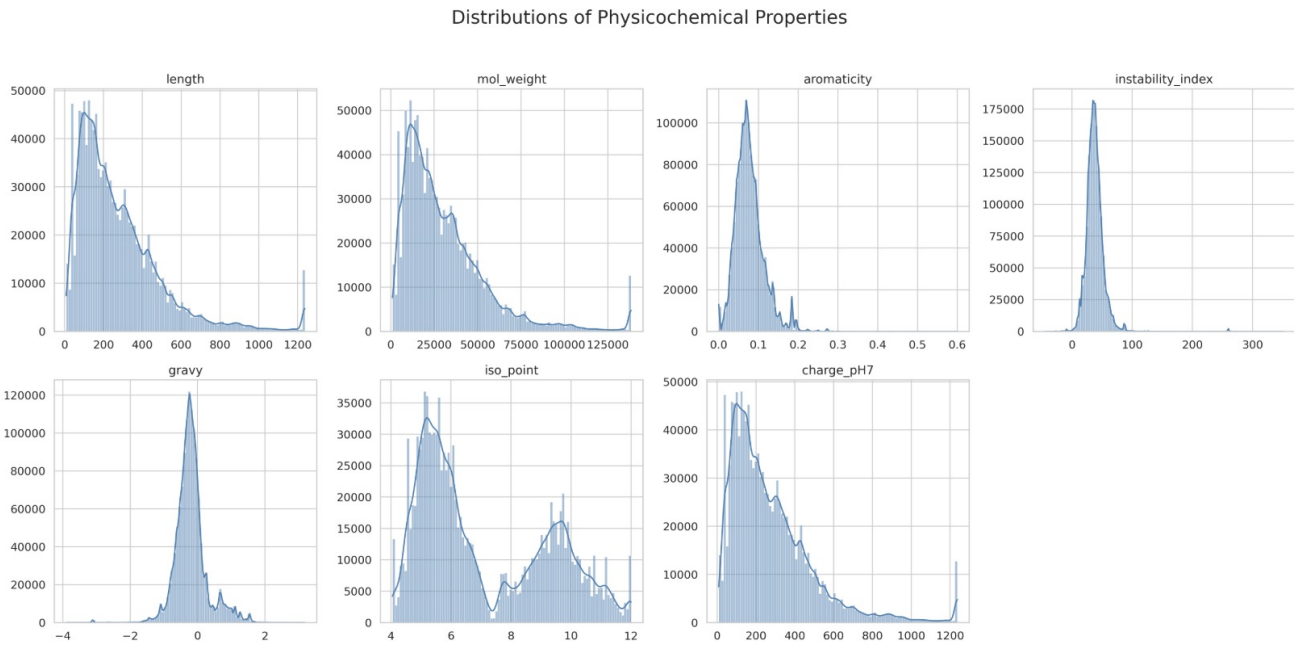

Figure 10: Distribution of features on label dataset

B. Prompts

We experimented with a variety of explainer system prompts with varying levels of directions and examples. The system prompt that generated the best results was:

```
You are an AI researcher investigating a specific neuron inside a protein language model.
    Your task is to describe the biological features of protein sequences that cause the
    neuron to strongly activate. Your goal is to generalize the neuron label by finding
    patterns in the features across all examples provided.

You will be given the following information:

<protein_sequences>
{{PROTEIN_SEQUENCES}}
</protein_sequences>

<biological_features>
{{BIOLOGICAL_FEATURES}}
</biological_features>

<activation_values>
{{ACTIVATION_VALUES}}
</activation_values>

Analyze the data provided:
```

```
1. Examine the protein sequences and their corresponding biological features.
2. Pay attention to the activation values for each sequence.
3. Look for common patterns or characteristics among sequences with high activation values
   .
4. Consider your knowledge of biology to interpret the significance of these patterns.

Formulate your description:
1. Identify the most important 1-2 features that consistently appear in sequences with
   high activation.
2. Focus on features that are common across most or all high-activation sequences.
3. Disregard features that vary significantly among the examples.
4. Create a concise, one-sentence description that captures the essence of what causes the
    neuron to strongly activate.

Output your final description inside <neuron_description> tags. Ensure your description:
- Is limited to one sentence
- Uses as few words as possible
- Directly states the relevant features without introductory phrases
- Describes only consistent patterns across the provided examples

Example high-quality responses:
"Strongly activates for sequences of membrane proteins involved in transmembrane transport
    processes."
"Strongly activates for proteins with negative gravy scores"
"Strongly activates for glycoproteins involved in cellular structural functions"
```

Then, the input prompt was:

```
You will be given a list of DNA or protein sequences and their associated biological
    features where a neuron strongly activates. Your task is to summarize the shared
    biological features among these sequences in one concise sentence.

Here is the list of sequences and their associated features:

<sequences_and_features>
{{SEQUENCES_AND_FEATURES}}
</sequences_and_features>

To complete this task, follow these steps:

1. Carefully read through all the sequences and their associated biological features.
2. Identify common themes or patterns in the biological features across the sequences.
3. Focus on the most prominent and frequently occurring features.
4. Synthesize these common features into a single, concise statement.

Your summary should capture the essence of the shared biological features using the fewest
    words possible while still conveying the key information.

Provide your summary in the following format:
<summary>
[Your one-sentence summary of shared biological features]
</summary>

Remember, brevity is crucial. Aim to use no more words than absolutely necessary to
    accurately convey the shared biological features.
```

The simulator model was trained using the following prompt:

```
    Task: Predict activation 0  10  . ONLY ANSWER WITH A NUMBER
    Neuron: {row['neuron_id']}
    Description: {hypo}
    Sequence: {seq}
    Features: {comp}
    ONLY ANSWER WITH A NUMBER BETWEEN 0 AND 10.
```

Where the appropriate values replace the elements in the brackets. Inference was then done with this prompt as well.

Then, the prompt for neuron selection was simply:

```
Answer only with a True or False. A neuron described as {neuron} be useful in trying
    to generate a protein with the following characteristic: {characteristic}?

For example:
Prompt: "Answer only with a True or False. A neuron described as "Encodes information
    about Zinc Fingers" be useful in trying to generate a protein with the following
    characteristic: "Alpha-Sheet"?", Answer: False
Prompt: "Answer only with a True or False. A neuron described as "Associated with high
     hydrophobicity" be useful in trying to generate a protein with the following
    characteristic: "Increasing hydrophobicity"?", Answer: True
```

C. Analysis of Generated Proteins

To understand how likely our steering loop is to generate proteins not in the distribution of proteins the neurons were labeled with, we plot the characteristics of the generated proteins against the distribution of the dataset we used to generate labels.

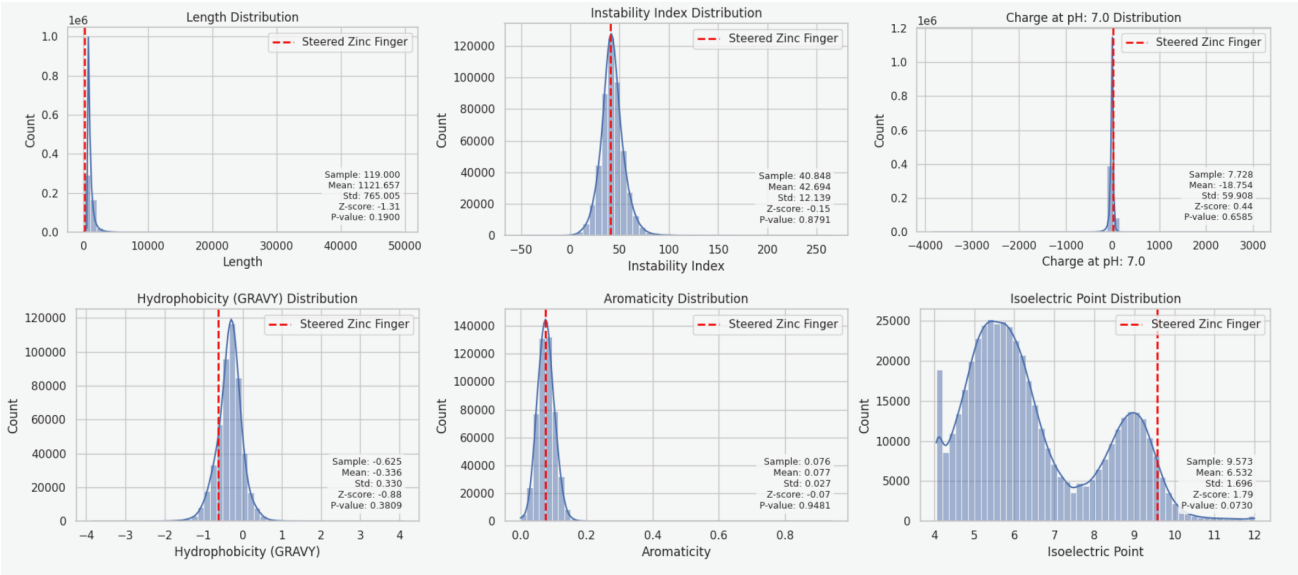

Figure 11: Relative Position of the Generated Zinc Finger Protein

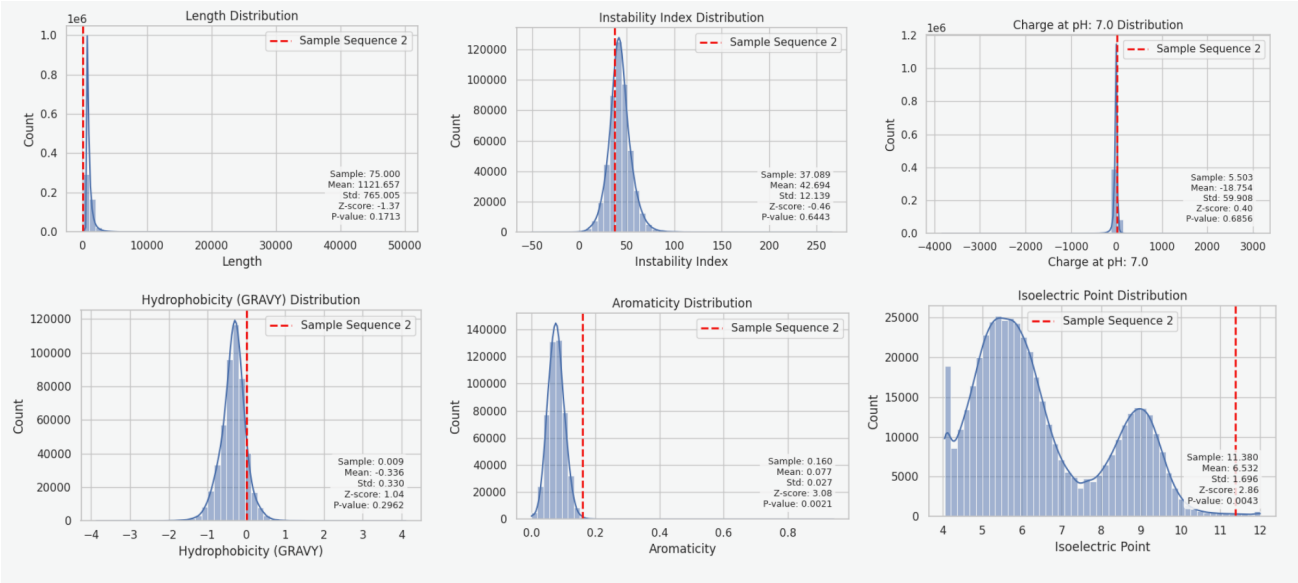

Figure 12: Relative Position of the Generated Alpha Helix Protein

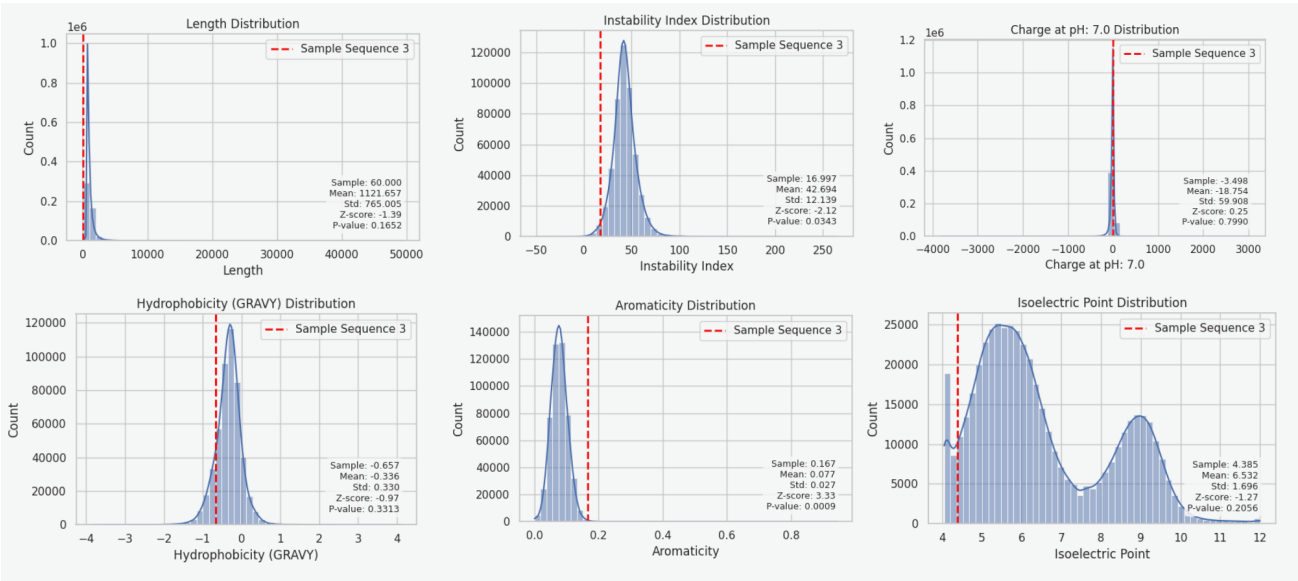

Figure 13: Relative Position of the Generated Beta Strand Protein

In general, we find that the characteristics of generated proteins roughly lie in the dataset distribution.

We then assess folding potential by calculating the ΔG of the various proteins generated. We find that essentially no protein generated would be able to viably fold. The procedure for calculating ΔG can be found below and is based on existing literature (Guerois et al., 2002):

```
Alpha Protein
BackHbond       =               -20.54
SideHbond       =               -8.28
Energy_VdW      =               -41.05
Electro         =               -2.47
Energy_SolvP    =               64.27
Energy_SolvH    =               -52.90
```

```
Energy_vdwclash =               6.10
energy_torsion  =               7.56
backbone_vdwclash=              60.68
Entropy_sidec   =               19.47
Entropy_mainc   =               127.51
water bonds     =               0.00
helix dipole    =               -0.05
loop_entropy    =               0.00
cis_bond        =               0.00
disulfide       =               0.00
kn electrostatic=               0.00
partial covalent interactions = 0.00
Energy_Ionisation =             0.00
Entropy Complex =               0.00
-----------------------------------------------------------
Total          =                                   99.64
```

```
Beta Protein

BackHbond       =               -13.22
SideHbond       =               -10.70
Energy_VdW      =               -23.19
Electro         =               -0.83
Energy_SolvP    =               42.47
Energy_SolvH    =               -26.60
Energy_vdwclash =               2.45
energy_torsion  =               11.96
backbone_vdwclash=              39.40
Entropy_sidec   =               14.64
Entropy_mainc   =               116.13
water bonds     =               0.00
helix dipole    =               -0.06
loop_entropy    =               0.00
cis_bond        =               0.00
disulfide       =               0.00
kn electrostatic=               0.00
partial covalent interactions = 0.00
Energy_Ionisation =             0.00
Entropy Complex =               0.00
-----------------------------------------------------------
Total          =                                   113.05
```

```
Zinc Finger
BackHbond       =               -37.56
SideHbond       =               -21.78
Energy_VdW      =               -74.44
Electro         =               -1.94
Energy_SolvP    =               111.03
Energy_SolvH    =               -96.64
Energy_vdwclash =               5.35
energy_torsion  =               15.02
backbone_vdwclash=              80.42
Entropy_sidec   =               43.27
Entropy_mainc   =               230.07
water bonds     =               0.00
helix dipole    =               -0.04
loop_entropy    =               0.00
cis_bond        =               0.00
disulfide       =               0.00
kn electrostatic=               0.00
partial covalent interactions = 0.00
Energy_Ionisation =             0.15
Entropy Complex =               0.00
-----------------------------------------------------------
```

```
Total          =                                    172.49
```