The First Proteinbender: A Novel "Structure-based Protein Search Engine"

Stanford CS224N Custom Project

Ethan Zhang (ethanz)
Dept. of Computer Science
Stanford University

Saahil Sundaresan (saahilss)
Dept. of Computer Science
Stanford University

Zane Chan (zanechan)
Dept. of Mathematics
Stanford University

Abstract

A protein's structure is arguably its most obvious and defining feature. However, there is no way to search protein databases using purely natural language structure descriptions - existing methods rely on information such as protein name or sequence and can perform only rudimentary keyword searches (eg. "DNA-binding domain," "beta sheet", etc.). Thus, it is difficult for users who have a desired structure in mind but lack information for further query inputs to search precisely for a target protein. We attempt to solve this critical bottleneck by implementing a natural language-to-protein pipeline with two capabilities: 1) structures retrieval given textual descriptions, and 2) modification of a structure given input structure and a textual description of an alteration: these synthetic structures can then be used to query databases for similar existing proteins. To this end, we report the curation of a novel multi-source dataset of 135,361 pairs of protein text and structure embeddings, as well as the training of a contrastive learning (CL) model to map text and structural information about a protein onto a low-dimensional joint embedding space, a joint space-to-structure embedding model, and several architectures of structure embedding-to-3D structure decoder/visualization models to recover 3D protein structures from the joint embedding space. Collectively, these models represent a significant step towards the realization of a "protein search engine" capable of fetching and editing protein structures given natural language input. We also propose a novel method for objective evaluation of text-based protein editing models based on known protein mutations.

1 Key Information to include

Mentor: Shijia Yang. No external collaborators or project sharing.
Contributions: Dataset curation was performed by EZ. SS wrote the linear baseline. SS and EZ worked on CL and the linear decoder. ZC worked on lookup, editing and coordination prediction. All authors contributed equally to ideation and this writeup.

2 Introduction

The most widely-used protein databases, such as the RCSB Protein Data Bank (PDB), offer several options for users to look up structures of interest. However, these options require users to already have precise knowledge about their structure of interest - for example, a user would need to know a protein's name, accession code, sequence, or other similar information in order to find it easily. Although these search options are reliable, they might pose difficulties for novice users or even for advanced researchers who lack specific query information.

We thus aimed to create a natural language input protein lookup tool which would allow users to more freely and intuitively query RCSB PDB's data, akin to a search engine. We envision not only the ability for users to utilize natural language as input, but also the capability to iteratively supply

further text input on top of previous queries, providing users ultimate flexibility in repeatedly refining their searches.

To our knowledge, our goal of creating a "protein search engine" is novel and has never been described in the literature previously. We faced two major challenges in our efforts: 1) there is no pre-existing database containing protein structures and respective well-formatted structural descriptions. Moreover, building such a dataset is a nontrivial task that requires both creativity as well as inter-domain expertise in order to curate the high-quality descriptions necessary for training models. 2) Text and protein structures are two highly distinct modes of data that require careful thought and implementation to jointly train a model on.

3 Related Work

Similar work has been carried out for small molecules. Liu et al. (2023b) utilize a contrastive learning framework to learn a joint space from small molecule and text embeddings, allowing for text-based editing of molecular structures. However, rather than a text-to-molecule lookup function, they instead implement functionality to match a given molecule to a pre-existing text caption.

Related work has been conducted towards the end of text-to-protein generation. Chroma, as described in Ingraham et al. (2023), uses a diffusion-based architecture to generate proteins and a diffusion-conditioner framework to allow users several modes of input to impose constraints on the protein generation process, including but not limited to natural language input. However, their text-structure database is comprised of only 45 thousand samples. Most critically, Chroma is a large model that requires minutes to generate a single structure, even when using a high-end GPU (eg. V100), making their generation process unwieldy for rapid lookup purposes. Liu et al. (2023a) produce a text-to-protein-sequence model with additional text-based editing functionality by employing a contrastive learning approach to align embeddings for protein sequences and text captions. However, sequence output adds an extra layer of prediction complexity, and since protein structure folding is highly dependent on sequence specifics, runs significant risks with regard to accuracy.

Overall, we aim to address two potential gaps that we identify in previously published works: 1) Current works focus on outputting full structures, which are inefficient for lookup purposes; 2) rather than focusing entirely on textual descriptions of protein structure, current text-to-protein models aim to generalize across a broad range of textual inputs (e.g. descriptions of chemical properties, binding properties, etc.) – and their training corpora are thus sparse in textual structure description, potentially weakening models' structure-specific capabilities.

4 Approach

We envisioned the following pipeline for a structure-based protein search and edit engine:
Edit tasks: Given a user-specified protein and a natural language description of desired modifications, we map both the base protein structure and the textual edit description to a joint representation space. Structural edits can be performed mathematically in the vector space, bypassing the need to edit the protein's primary structure (i.e. its actual amino acid sequence). The joint embedding space would then be mapped back out to the 3D structure space via a decoder to generate the edited protein as an output.
Search Tasks: The above edit-oriented pipeline is already compatible with text-based lookup. Given the natural language description of a protein, the description's embedding is mapped onto the joint representation space and then run through the decoder to generate an output 3D structure. Given a database of 3D protein structures, we identify the closest structure in the database to the generated structure, and output that as the protein in question, avoiding computationally intensive and unstable predictions with regards to the protein's primary residue sequence as an intermediate step. Implementation of the above pipeline requires the following: (1) embedding schemes for the text and structure input data, (2) a contrastive learning model to learn a joint representation space between the text and structure embeddings, (3) a decoder mechanism from the joint representation space back to a 3D structure space, and (4) lookup methods to find proteins nearby to the generated structure, as well as metrics to evaluate protein editing.

For (1), we embed protein structures as 401-dimensional descriptor vectors using GraSR as described in Xia et al. (2022). We choose this encoding method in particular due to the descriptor representing

the protein's geometric information (as opposed to chemical or information not closely related to structure), its ability to discriminate between protein classes, and its ability to use the low-dimensional descriptor to rapidly query databases for structurally similar proteins. Additionally, as GraSR is already designed specifically for embedding protein structures, it does not need to be further optimized for our task. We embed natural language input of structural and edit descriptions as 768-vectors using SciBERT, a pretrained BERT language model optimized for scientific data.

All models described below were trained in PyTorch with a fixed 80-10-10 train-test-validation split. All models used the Adam optimizer, described by Kingma and Ba (2017). All model weights were initialized with the Xavier uniform distribution, and all biases were initialized to either 0.0 or 0.1.

Linear Baseline: As a baseline, we train a simple feed-forward neural network to map from fine-tuned SciBERT text embeddings $t \in \mathbb{R}^{768}$ to GraSR structure embeddings $s \in \mathbb{R}^{401}$. The model consists of three linear layers interspersed with ReLU and dropout for nonlinearities, trained using MSE loss.

Contrastive Learning: We adapt and modify code from Liu et al. (2023b) to train two projectors E_T and E_S to align text ($\mathbb{R}^{768}$) and structure ($\mathbb{R}^{401}$) embeddings, respectively, in an $\mathbb{R}^k$ joint embedding space. Both E_T and E_S consist of three fully-connected layers. We use the InfoNCE loss function

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{2}\mathbb{E}_{x_s,x_t}\Big[\log \frac{\exp(E(x_s,x_t))}{\exp(E(x_s,x_t)) + \sum_{x_{t'}} \exp(E(x_s,x_{t'}))} + \log \frac{\exp(E(x_s,x_t))}{\exp(E(x_s,x_t)) + \sum_{x_{s'}} \exp(E(x_{s'},x_t))}\Big],$$

where x_s and x_t are pairs of structure and text joint space embeddings (eg. $E_S(s), E_T(t)$), $x_{s'}$ and $x_{t'}$ are negative samples (i.e. a non-pair text or protein embedding), and $E(x, y)$ computes the inner product between x and y. Our choice of this loss function is motivated by the unsupervised nature of the task, where we have positive pairs (i.e. text and structure embeddings describing the same source protein), but negative pairs can be randomly sampled; optimizing InfoNCE loss should thus learn a robust and generalizable joint embedding space where edits can be performed by combining structural embeddings of existing proteins with the text embedding of requested edits.

Joint Embedding Decoder: We then train a decoder to map joint embedding space text embeddings back to the GraSR embedding space. The decoder consists of four linear layers with interspersed ReLU and dropout layers for nonlinearity, trained on MSE loss. The motivation for having a separate decoder (as opposed to just directly mapping from text to structure as in the linear baseline) is that performing structure edits under our paradigm requires combinations of disjoint data types (structure and text) in a joint embedding space, necessitating a separate decoder from this space back to the structure level.

Coordinate Prediction: From the post-decoding GrasR embedding, we train a visualizer to predict 3-D prediction of individual alpha carbon coordinates per protein. We trained three models for this task: a three-layer fully connected model, a decoder LSTM, and a convolution model feeding into a 6-layer transformer. The hidden dimension size was 512, with a dropout rate of 0.5 and a learning rate of 0.01 with a scheduled decay of 0.1 at the fifth and tenth epoch. Each model masked predicted outputs after the designated protein length, which was included in the final, 401 dimension of the GrasR inputs. The final loss function is

$$\frac{1}{n}\sum_{i=1}^{n}(l_i - \hat{l}_i)^2 + \mu\mathcal{L}_{kabsch}(x, \hat{x})$$

where the first term is the MSE loss over true and predicted bond lengths l_i and $\hat{l}_i$, μ is 0.1, and the second term is the kabsch rmsd. Kabsch-rmsd was used to learn high-level protein structural information, as GrasR is a rotation-invariant embedding.

5 Experiments

5.1 Data

We construct a novel dataset of 135,361 text-protein structure pairs, over 3 times larger than the most extensive previously-reported datasets (Ingraham et al., 2023). Our dataset draws upon three sources - text extracted from academic articles' full texts, academic articles' abstracts, and select samples

from SwissProtClap, a dataset of over 400,000 text-protein pairs created and described by Liu et al. (2023a). A summary of our dataset and its constituent sources can be found in Table 1 below.

Most structures published in the Protein Data Bank have an associated publication describing the new protein's structure solving process - we use PyPaperBot to scrape SciHub for these corresponding publications and extract raw text from the downloaded PDFs using PyMuPDF's Fitz library. We then used Gemini Pro 1.5's API to automatically extract text that described the associated protein's structure and discard the rest.

We also retrieve each structure's associated PubMed abstract and use a regex search to keep sentences concerning protein structure, discarding the rest.

Finally, we refine a previously-described dataset, SwissProtClap. The original dataset's text portion is obtained from SwissProt - however, SwissProt's protein annotations are often primarily concerned with function, localization, and non-structural properties. As such, we refine the text data by performing the same regex search as for our abstract data. Moreover, because SwissProtClap was used to train a text-to-protein sequence model, we were required to obtain corresponding structures for each caption. Inconveniently, multiple SwissProt entries can correspond to a single PDB entry - this is because SwissProt entries can describe small segments of proteins (domains) rather than entire structures. As a result, we instead obtain structures from the AlphaFold Protein Structure Database, described by Varadi et al. (2023), due to its one-to-one mapping with SwissProt entries.

Finally, we use GraSR and SciBERT to embed our text and structure data, giving us a final dataset of 135,361 $\mathbb{R}^{401} : \mathbb{R}^{768}$ embedding pairs, respectively.

	# of Samples	Methods
Papers	8,258	PyPaperBot, Fitz, Gemini Pro 1.5
Abstracts	32,622 5	Regex
MinSPC	94,481	Regex

Table 1: Summary of protein-text pair dataset

5.2 Evaluation method

Contrastive Learning: Beyond the InfoNCE loss function used during training, we evaluate the contrastive learning model on our curated dataset using mean RMSD error in order to capture the average "distance" between structure- and text-based representations of the same protein in the joint embedding space. In particular, for a pair of learned projection schemes $E_T : \mathbb{R}^{768} \to \mathbb{R}^k$ (for text embeddings) and $E_S : \mathbb{R}^{401} \to \mathbb{R}^k$ (for structure embeddings) onto a k-dimensional joint representation space, we compute mean $RMSD$ distance over N examples

$$Eval_{E_T,E_S} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{\sum_{j=1}^{k}(E_S(s_i)_j - E_T(t_i)_j)^2}{k}}$$

where $s_i \in \mathbb{R}^{401}$ and $t_i \in \mathbb{R}^{768}$ are the GraSR-based structural and SciBERT-based textual description embeddings respectively for protein i. A lower mean RMSD score indicates a joint representation space where text and structure embeddings of proteins are perfectly aligned, and vice versa.

Joint Embedding Decoder: We again evaluate using an RMSD-based method, comparing the mean RMSD distance between ground-truth GraSR embeddings $s_i \in \mathbb{R}^{401}$ from the structure portion of our curated dataset with the predicted GraSR embeddings $\hat{s}_i \in \mathbb{R}^{401}$ for those same proteins computed by running their SciBERT text description embeddings ($t_i \in \mathbb{R}^{768}$) through learned projector $E_T : \mathbb{R}^{768} \to \mathbb{R}^k$. It is of note that the quality of RMSD values for the decoder model is thus partially dependent on the quality of the joint representation space learnt during contrastive learning. Therefore, given a k- dimensional joint embedding space with text projector E_T, we evaluate our decoder $Dec : \mathbb{R}^k \to \mathbb{R}^{401}$ over N examples

$$Eval_{Dec,E_T} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{\sum_{j=1}^{401}(s_{ij} - \hat{s}_{ij})^2}{k}}$$

$$\hat{s}_i = Dec(E_T(t_i))$$

Text-based Lookup: From our test set, we randomly select 100 samples to evaluate text-based lookup. Our metrics for look-up success were prediction accuracy, as well as TM-align similarity of the output and target proteins. TM-align is a structural alignment algorithm used to compare protein 3D structures by aligning their C-alpha backbones and measuring the similarity using the TM-score, which ranges from 0 to 1. A TM-score closer to 1 indicates a higher structural similarity, while a score above 0.5 typically signifies that the proteins share the same fold. Conversely, a TM-score below 0.2 generally suggests that the similarity is no better than that between random proteins. The TM-score is calculated by normalizing the root-mean-square deviation (RMSD) of the aligned residues, taking into account the size of the proteins. The equation for TM-score is:

$$\text{TM-score} = \max\left(\frac{1}{L_{\text{target}}}\sum_{i=1}^{L_{\text{aligned}}}\frac{1}{1+\left(\frac{d_i}{d_0(L_{\text{target}})}\right)^2}\right)$$

where L_{target} is the length of the target protein, L_{aligned} is the number of aligned residues, d_i is the distance between the i-th pair of residues, and $d_0(L_{\text{target}})$ is a scale factor that depends on L_{target}. TM-align employs dynamic programming to optimize the alignment, providing both an optimal superposition of the structures and a detailed residue-residue correspondence, making it a valuable tool for structural bioinformatics and comparative protein analysis.

Coordinate Prediction: We used a mixture of quantitative and qualitative methods. Quantitative methods included Kabsch RMSD, MSE of pairwise distance, MSE of bond length, and primarily, the BioPython Superimposer similarity function. Kabsch RMSD loss is the RMSD loss after computing the optimal rotation using an SVD decomposition of the correlation matrix. The BioPython Superimposer computes the rotation invariant similarity of two proteins. Qualitatively, we created 3-D plots of sampled output and target alpha-carbon structures, which provided visual confirmation of the model's ability to learn protein-like arrangements.

Editing: Due to the extreme sparsity of data containing unmutated structure, mutated structure, and textual description describing structural differences, we constructed three such datapoints. Because the editing task is far more difficult than lookup, we set our evaluation metric as whether or not the target structure was within the top 100 results outputted by our model following our editing computation.

5.3 Experimental details

Linear Baseline: We experimented with varying sizes of the hidden layer dimensions and dropout probabilities, as well as conducting an initial learning rate sweep for the Adam optimizer. As a result of these experiments, we settled on hidden layer dimensions `([768, 1024], [1024, 1024], [1024, 401])`, dropout probability $p = 0.2$, and initial Adam learning rate `1e-4`. We allowed the model to train for 50 epochs.

Contrastive Learning: We experimented with varying the sizes of hidden layers, the initial learning rates for both projectors, as well as batch size. The final text projector E_T and structure predictor E_S have layers of size `([768, 1536], [1536, 1536], [1536, 256])` and `([401, 802], [802, 802], [802, 256])`, respectively, while we found an optimal learning rate of `1e-5` for both projectors. Our final model was trained for 60 epochs with a batch size of 64. We also experimented with the final dimension size k of the joint representation space in conjunction with the decoder model (so as to have a representation space low-dimensional enough to allow learning of high-quality joint embeddings, but still complex enough to allow a reasonably faithful projection back into $\mathbb{R}^{401}$ GraSR structure space), and experimentally settled on an optimal value of $k = 256$.

Joint Embedding Decoder: We settled on linear layer dimensions `([256,768],[768, 1024], [1024, 768], [768, 401])` with dropout probability $p = 0.2$. We allowed the model to train for 50 epochs, and a after a learning rate sweep settled on initial learning rate `1e-5` and batch size 32.

Coordinate Prediction: We had three categories of variation: loss function, model configuration, and model hyperparameters.

Loss functions were chosen to balance high-level relative geometry and low-level structure. Trialed loss functions included combinations of L1, L2, and Kabsch RMSD loss, as well as MSE of pairwise alpha carbon distance and bond length. Combinations of loss functions were initially varied and tested over a simple linear model before being applied to the larger LSTM and transformer models.

Three model architectures were used to predict output coordinates: linear, LSTM, and transformer. We hypothesized that the LSTM would outperform the linear model due to protein sequentiality. Similarly, we hypothesized that the transformer would obtain better coordinate predictions since self-attention would enable it to track the surrounding protein structure. Variables included with and without dropout, learning rate schedulers, and encoding methods for the LSTM and transformer. Trialed encoding methods were linear models with with the ReLU activation function, then an LSTM encoder for the LSTM and a convolution layer for the transformer. We hypothesized that the convolution layer would learn sequential information from the 2D GrasR embedding space, which would better captilalize on transformer encoder attention.

Finally, we varied hyperparameters over each model, including learning rate, batch size, hidden dimension size, dropout probability, and scheduler milestones.

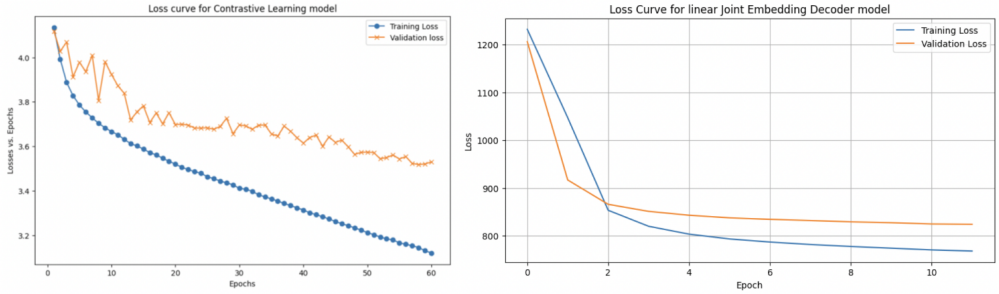


Figure 1: Loss curves for Contrastive Learning and linear Joint Embedding Decoder models using hyperparameters described in Section 5.3.

Editing: We try various values for α and β, adjusting to minimize the L2 norm of the combined vector and the target vector.

6 Results and Analysis

Intermediate Evaluation (CL): We used RMSD evaluation scores alongside validation loss as criteria for selecting the best-performing layer dimensions and hyperparameter values for our Contrastive Learning and linear Joint Embedding Decoder models. While RMSD scores are difficult to intepret as standalone results due to their being scale-dependent, we report here the highest-achieved RMSD scores for the Contrastive Learning task as an intermediate result, as well as what RMSD scores would look like for a "random projection" on the dataset as a benchmark. We also provide loss curves for both the CL and linear Joint Embedding Decoder models in Figure 1. For the Contrastive Learning task, which mapped inputs into a $\mathbb{R}^{256}$ joint representation space, we achieved test set score $Eval_{E_T,E_S} = 0.206$ against a baseline eval score (after shuffling the rows of the data to generate random pairs) of 0.249, with similar values on randomly sampled subsets. As an intermediate result, this indicates our model at least performs consistently better than random at mapping the joint embeddings.

However, we observed that it was quite hard to get contrastive validation loss values below 3.6, or joint embedding validation loss values below 800 regardless of changing the number of layers or changing hyperparameters. We attribute this in large part to the unstandardized nature of the text embeddings in our dataset; the text data we were able to compile even after this was highly variable in description length and scope, which likely contributed to lower contrastive learning and linear decoder performance than we had initially hoped for, affecting the final lookup outputs.

Text Lookup: For our baseline text-to-GraSR linear model, we achieve a TMalign score of 0.167. Using our contrastive learning approach, we improve our score to 0.198. Interestingly, we experienced an odd phenomenon where the top-scoring lookups returned by GraSR were consistently from a particular small (ca. 10) set of proteins. We initially hypothesized that this was due to our models outputting vectors in an unpopulated region of the GraSR embedding space, causing lookups to return the same outliers repeatedly. However, a PCA plot of the entire PDBs' GraSR embeddings as well as our test set's GraSR embeddings (CL method) reveal this to not be the case (Figure 2).

Hence, we also attempt a nearest-neighbor approach within the GraSR embedding space, where we compute our lookup result w as $\arg\min_{w \in W} \|w - q\|_2$, where q is the query $\mathbb{R}^{401}$ vector. With this approach, we manage to achieve a TMalign score of 0.242. Unsurprisingly, this below TMAlign's structural similarity significance threshold of 0.5, given the extremely difficult nature of the text-to-structure mapping task. Moreover, because textual descriptions are often short (ca. 1 paragraph), it is impossible for them to capture the full of a protein's structure - this would result in many reasonable matches for a text query, which likely was a key cause behind the low TMalign scores.

We further qualitatively evaluate our model's capabilities by querying it with free text prompts (Figure 3). We find that, although our model can interpret semantic information to a moderate degree and understand high-level structural ideas (Fig. 3a, b), it is unable to properly interpret more granular instructions (Fig. 3c). Our model also tends to fail on requests that are more abstract in nature/require reasoning. (Fig. 3d).

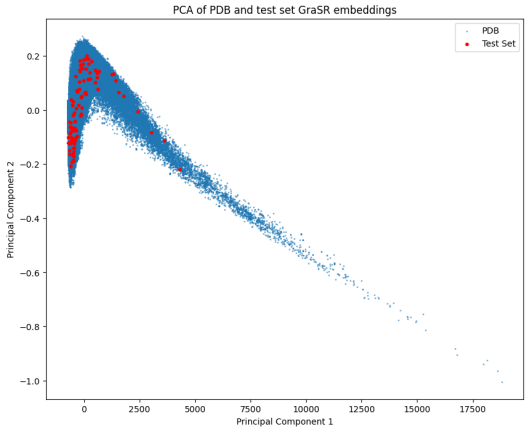


Figure 2: Plot of the first two principal components of GraSR embeddings of all PDB proteins and our test set proteins.

Coordinate Prediction:

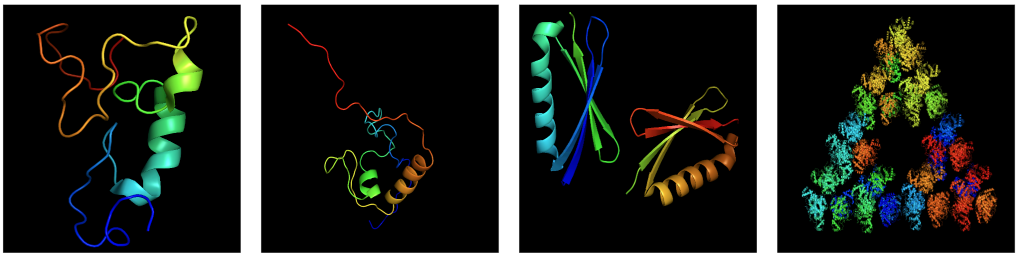
Loss Function	Average Biopython Similarity Score
L1	20.586
Pairwise Distance MSE	20.458
Bond Length	19.613
MSE	19.598
Chosen Loss Function	17.993
Kabsch RMSD	15.166

Table 2: Comparison of Loss Functions and Average Biopython Similarity Scores

For our loss function experimentation, we found that the Kabsch RMSD yielded the highest biopython similarity score, as expected. Surprisingly, it outperformed the other loss functions by a large margin. For our loss function experimentation, we found that the Kabsch RMSD yielded the highest biopython similarity score, as expected. However, qualitatively, we found that different loss functions optimized over different protein features. Losses like pairwise distance and Kabsch RMS appeared to learn high-level protein structure, but failed to learning physically realistic behavior. We found that including bond length in the loss qualitatively allowed for more realistic outputs.

Model	Average Biopython Similarity Score
Linear	12.545
LSTM	13.257
Transformer	13.659

Table 3: Comparison of Architectures and Average Biopython Similarity Scores



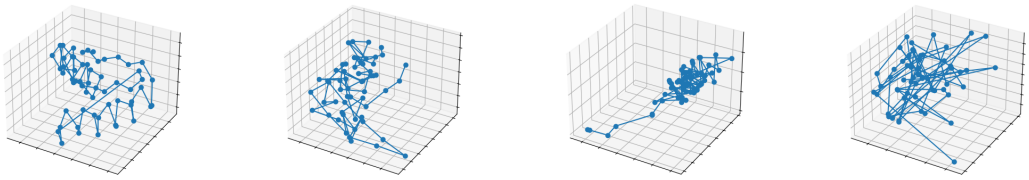
a) Prompt: "A very tiny protein with no beta sheets and one alpha helix." 1st result.

b) Prompt: "A protein with a disordered backbone and no defined tertiary structure." 2nd result.

c) Prompt: "A protein complex consisting of three homodimers joined along a threefold symmetry axis." 1st result.

d) Prompt: "A protein shaped like a 'Y'." 1st result.

Figure 3: Input text queries and the corresponding output proteins returned by our CL GraSR embedding space nearest neighbor model.



(a) True Protein (b) Linear (c) LSTM (d) Transformer

Figure 4: Side-by-side comparison of images

Interestingly, the linear baseline performed the best among the three main models. Additionally, while average biopython similarity scores were similar across model architectures, qualitative results were very different. The linear model qualitatively produced the most realistic output plots, learning reasonably accurate high level protein structure from the low dimensional embedding. Because the Kabsch RMSD loss is invariant across rotation, the outputted coordinates may be rotations of the ground truth. Indeed, see that the linear model generates a mirrored version of the ground truth image.

Editing: We choose parameters $\alpha = 0.95$ and $\beta = 0.05$, such that our final vector $v = \alpha E_S(s) + \beta E_T(t)$. In spite of our tuning efforts, none of the three desired structures were in the top 100 outputs for their respective inputs. However, an average TMalign score of 0.329 was reported for our three cases. Although our sample size is very small, this TMalign score indicates that there is great promise in our editing methodology, and that protein edits can be somewhat reliably expressed as combinations of textual and structural input in embedding space.

7 Conclusion

In this project, we aimed to develop a novel "structure-based protein search engine" capable of retrieving and modifying protein structures using natural language descriptions. Our approach involved curating a substantial dataset of 135,361 text-protein structure pairs, training a contrastive learning model to create a joint embedding space for text and structure representations, and developing various models to decode these embeddings into 3D structures.

First, we successfully created the largest known dataset for text-structure pairs in the protein domain, significantly enhancing the potential for learning robust models. One primary limitation of this is the scarcity of data containing explicit structural mutations and their textual descriptions limited the performance of our editing models. Most text descriptions emphasize chemical or biological function. As such, finetuning a model to better parse for tertiary structure and expanding our dataset with more

detailed structural descriptions and corresponding mutations could significantly improve the accuracy and reliability of our models.

Second, our contrastive learning model effectively mapped text and structure embeddings into a joint space, achieving a notable improvement over baseline methods in RMSD evaluation scores. That said, our models struggled with capturing detailed and specific structural features from short textual descriptions, leading to lower-than-expected TM-align scores. Developing more sophisticated evaluation metrics that can better capture the nuances of protein structure prediction and editing will be crucial for future improvements.

Third, our text-based lookup model demonstrated the ability to interpret high-level structural concepts from textual descriptions. Qualitative examination of the text look-up functionality showed promise for some examples. That said, given the nature of our input data, model predictions faced significant challenges with fine-grained accuracy.

Fourth, the model could capture high-level structural features effectively for the the coordinate prediction task. Surprisingly, the linear model performed the LSTM and transformer. Significant work could be done on improving the predictions for lower-level and more finte-grained structural information.

In conclusion, our work represents a first step towards enabling natural language-based protein structure search and modification. This would have exciting implication in lowering the barrier to entry for biologists to integrate computational methods into their work. While significant challenges remain, our findings represent a first step into a more lightweight and exciting intersection between natural language processing and biology.

8 Ethics Statement

While our project is largely technical and exploratory in nature, there are a few possibilities that may pose ethical and possible societal risks.

First is the potential for misrepresented output from our model. Incorrect outputs could misrepresent protein function or interactions, leading to incorrect scientific conclusions or flawed biomedical applications. This is especially the case if this project were to be used in an automated pipeline, with no human verification of results. Another possibility is that the model generates a potentially harmful protein.

To mitigate this risk, we have taken several steps. First, we use a diverse datasource, comprising 135,361 text-protein structure pairs from varied sources, including academic articles and SwissProtClap, to ensure a wide representation of protein structures and descriptions. Second, we have implemented robust evaluation metrics, such as RMSD and TM-align, to quantitatively assess the accuracy and reliability of our models. Third, we recommend that any designed structures produced by our model be subjected to thorough experimental validation in the laboratory before any scientific or biomedical application.

Second, the technology could be used for harmful purposes, such as designing pathogenic proteins. We have considered several mitigation strategies. First, we will avoid training our models on data clearly associated with potentially harmful applications. This includes omitting datasets known to be used for pathogenic or bio-weapon research. Second, implementing strict usage or access controls on the system can help prevent misuse. This would include user authentication, monitoring, and limiting access to only authorized and verified researchers. Third, developing and enforcing ethical use policies for our technology may be helpful. Users would have to agree to adhere to policies, which should explicitly prohibit the use of the technology for harmful purposes.

References

John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. 2023. Illuminating protein space with a programmable generative model. *Nature*, 623:1070–1078.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Shengchao Liu, Yanjing Li, Zhuoxinran Li, Anthony Gitter, Yutao Zhu, Jiarui Lu, Zhao Xu, Weili Nie, Arvind Ramanathan, Chaowei Xiao, Jian Tang, Hongyu Guo, and Anima Anandkumar. 2023a. A text-guided protein design framework.

Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Animashree Anandkumar. 2023b. Multi-modal molecule structure–text model for text-based retrieval and editing. *Nature Machine Intelligence*, 5:1447–1457.

Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingi Yeo, Oleg Kovalevskiy, Kathryn Tunyasuvunakool, Agata Laydon, Augustin Žídek, Hamish Tomlinson, Dhavanthi Hariharan, Josh Abrahamson, Tim Green, John Jumper, Ewan Birney, Martin Steinegger, Demis Hassabis, and Sameer Velankar. 2023. AlphaFold Protein Structure Database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic Acids Research*, 52(D1):D368–D375.

Chunqiu Xia, Shi-Hao Feng, Ying Xia, Xiaoyong Pan, and Hong-Bin Shen. 2022. Fast protein structure comparison through effective representation learning with contrastive graph neural networks. *PLOS Computational Biology*, 18(3):1–21.