# Final Report: Conditional Protein Sequence Generation and Improved Evaluation

**Ross DeVito**
UC San Diego
La Jolla, CA 92093
rdevito@ucsd.edu

## Abstract

Generative models are pushing the state of the art in image and text generation, but the landscape is less clear for protein generation. This project aims to benchmark two classes of generative models, autoregressive language models and denoising diffusion probabilistic models (DDPMs), on even footing on the task of conditional protein generation. Both will be implemented as transformers with the same architecture and trained on the same subset of UniRef50. As is done in the image and text domains, we incorporate several evaluation metrics based on embeddings from a pretrained model, ProtTrans. These aim to evaluate overall performance with respect to the real distribution and the efficacy of conditioning. The code is available at github.com/RossDeVito/conditional_protein_gen_proj.

## 1 Introduction

The goal in protein engineering is to generate a new proteins that will have some desired function. The wide range of protein uses from enzymes to vaccines and the limited number of functionally similar examples raises the question of how to condition a model in a way that will yield relevant candidates. Previous generative modeling work has used tags related to molecular or cellular function (15) while recent papers using diffusion models have focused on using structural constraints (3) or inpainting (13; 20).

**Problem Definition.** For this project we focus on generating discrete amino acid sequences conditioned on sets of tags, as is done in ProGen (15). The tags represent properties of proteins such as associated biological processes, chemical properties, and taxa of organisms in which they are found. This formulation has the benefit of being able to leverage large databases of millions of proteins annotated with these common tags.

**Problem Significance.** Protein engineering is an important for developing compounds and enzymes with medical and industrial applications. This includes proteins with pharmaceutical properties, proteins used as enzymes or guides in gene editing, and proteins desired with flavors or scents. Machine learning can help address a number of the challenges in protein design, namely the large state space and the difficulty of evaluation. Conditional generative models can lead to better exploration of the space of possibly relevant proteins and less wasted cost and effort computationally and physically evaluating poor candidates.

**Technical Challenge.** Part of the technical challenge in protein generation is the lack standard datasets, evaluation methods, and problem formulations. Hundreds of millions of annotated protein sequences are available, but papers typically focus on different ways on formulating the task, using the data, and evaluating performance that makes head to head comparison difficult. Recent publicly

Preprint. Under review.

available models are mainly language models only useful for protein generation not conditioned on desired properties (8; 16; 9).

Another part of the challenge is the difficulty of evaluating generated proteins without synthesizing and physical testing. Predicting how a protein will behave and interact is a difficult problem itself despite the computational and biological research efforts. This in turn makes it hard to know if proteins generated would display some desired property. Unlike for images and text, we cannot fall back on human evaluation of proteins from looking at examples. Actually synthesizing proteins and running experiments to verify properties is typically prohibitively costly or time consuming.

NLP metrics like perplexity give some information about how a given model fits a distribution of sequences, but they may not give much intuition into a proteins chemical or biological function. Another approach is to rely on sequence based bioinformatics approaches for evaluation. Multiple sequence alignment (MSA), used by tools like BLAST (2), can find similar real world proteins that may give insight into the potential function. Other approaches annotate structural features in generated protein sequences and compare them with those in known proteins with the desired properties.

Both of these bioinformatics approaches suffer from their insensitivity to the effects of small changes in a protein sequence that drastically change its properties. Evaluating the impact these small differences during evaluation requires some reliable model of the change's effects more complex than sequence similarity.

**State-of-the-Art.** ProGen (15), a 1.2 billion parameter conditional autoregressive language model, was trained to generate discrete amino acid protein sequences conditioned on a set of attribute tags. These tags covered attributes about the proteins, like their cellular location, molecular function, and associated biological processes, as well as taxonomic information on in which species the proteins are found. Conditioning is done by inserting vectors representing the tags to the beginning of the sequence, then adding one amino acid at a time to the end. ProGen displays performance similar to large language models in NLP in perplexity and was shown to generalize to unseen protein families, but the autoregressive generation limits the ability of the model to be used in directed modification of an existing protein through approaches like inpainting.

Recent advances in DDPMs have led to research on their use in protein engineering. This work has tended to focus on the 3D structure of proteins, which is important for their ultimate function. One approach to conditioning DDPMs to generate functionally relevant proteins is to view generation as an inpainting problem. In SMCDiff (20) generation is formulated as a motif-scaffolding problem, where the model must create a stable protein scaffold to surround a motif with the desired functional properties. This allowed for generation of longer and more structurally diverse 3D scaffolds, but the model cannot generate to motifs not in the training set.

In ProteinSGM (13), an unconditioned DDPM is trained to generate 6D coordinate representations of proteins. There is a limited ability to do conditional generation by using inpainting with the unconditioned model to replace unwanted parts of some protein, but there is no conditioning of what is generated. The indicated part of the protein is just replaced with a sequence the DDPM finds generally likely. ProteinSGM generates only secondary structure of proteins ($\alpha$-helices,$\beta$-sheets, etc.), not the underlying amino acid sequence primary structure.

Anand and Achim (3) used a DDPM with both a continuous component to predict 3D structure and a discrete DDPM to predict the corresponding amino acid sequence. Unlike the previous inpainting examples, conditioning was done using a coarse user defined topology of the desired protein. Each block in this conditioning scheme represented one helix, beta sheet, or loop, and during training conditioning would sometimes be under specified by dropping out some adjacency information between blocks. During generation a continuous DDPM first generates a 3D backbone conditioned on the constraints, then a discrete DDPM (5) generates the amino acid sequence based on the backbone.

## 2 Related Work

### 2.1 Conditional Generation

The most notable publicly available models for protein generation, ProGen2 and ProtGPT2 (16; 9), are autoregressive language models trained on protein sequences. These models were trained for

generation not conditioned on any desired protein properties. They instead try to generate protein sequences with properties that make them like the sequences in the real distribution. Autoregressive language models like these can be used in a forms of conditional generation where the task is formulated as motif-scaffolding problem. The major limitation of using motif scaffolding for conditioning is that there must be prior knowledge of what motifs will produce the desired effects if scaffolded in a valid way. Diversity is also limited, as only the scaffold around the motifs that determine function is changing. Models trained specifically for these motif-scaffolding or inpainting tasks, like SMCDiff (20), also struggle to generalize well outside the training motif set.

Focusing on models explicitly for generation conditioned on some desired properties, a major question in what exactly to condition on. ProGen (15) is an autoregressive language model like ProGen2, but it was trained on sequences with preprended tags that indicated chemical, cellular, and biological attributes as well as taxa the protein may be found in. Anand and Achim (3) instead condition on the desired topology on the generated protein. In cases where the topology that will lead to the desired properties can be inferred, this kind of conditioning seems like the more straightforward approach to generate candidates. In other cases where desired properties can be matched to functional tags in the Genome Ontology or narrowed down to some taxa, models can leverage the hundreds of millions of natural protein sequences annotated with these tags. These tag conditioned models would likely generate proteins with more diverse topologies than models explicitly conditioned on a topology, but it may come at the cost of generating more poor candidates if the tags underspecify what is desired.

### 2.2 Evaluation

In ProGen, the authors compared primary structure prediction using pairwise global alignment, secondary structure prediction using PSIPRED (11), and conformational energy analysis using Rosetta-RelaxBB (1). In the case of secondary structure prediction, PSIPRED simulates the rudimentary protein folding of an amino acid sequence as it forms the foundation of its final peptide. Lastly, conformational energy analysis calculates "fullatom models" that simulate the biochemical forces undergone during polypeptide chain formation, with the intuition that a peptide with a very high conformational energy cost is less likely to fold naturally.

We hope to expand on these metrics by using Precision and Recall for Distributions (PRD) (18) to help evaluate the trade off between quality of generated samples and the coverage of the modes of the real distribution of protein sequences. To generate meaningful embeddings of the protein sequences required to compute the PRD, we will use ProtTrans (8). ProtTrans is a pretrained protein language model that can be used to embed variable-length amino acid sequences as fixed-length vectors.

For baselines, the authors of ProGen used random amino-acid sampling and quasi-random BLOSUM62-informed sampling. Random sampling is simply that, each amino acid is equally likely to be chosen when sampling. BLOSUM62 (10), however, is an amino acid substitution matrix derived from experimental results; quasi-randomly sampling from BLOSUM62 will inherently yield more biologically relevant substitutions while still maintaining a general randomness. The authors found that primary structure analysis yielded comparable results between ProGen and these random baselines, but both secondary structure analysis and conformational energy analysis yielded favorable results for ProGen.

## 3 Methodology

**Problem Setting.** Here we focus on protein sequence generation conditioned on a set of tags $C$. This set of conditioning information can contain attributes of proteins used with UniRef (19) reference clusters. This includes the common taxa for the cluster and biological process, cellular component, and molecular function Gene Ontology (GO) terms (4; 7). For unconditioned generation, $C$ contains just the root taxa tag.

The generated protein sequence is $x$ where $x_i$ is a token representing the protein's $i^{th}$ amino acid. There are 21 tokens representing amino acids. The standard 20 are used as well as X to represent unknown or rare amino acids. This included asparagine, glutamine, selenocysteine, and pyrrolysine, which collectively only appeared 146 times in the training set.

We are applying autoregressive language models (ARLMs) and DDPMs to this problem, which formulate generation in different ways. Autoregressive models formulate generation as next token prediction additionally conditioned on the tag set $C$, where we learn the model given by equation 1.

$$P_\theta(x_i|x_{<i}, C) \tag{1}$$

Diffusion models instead generate by iteratively denoising sequences of pure noise into protein sequences. This is done by learning a function $p_\theta(x_{t-1}|x_t)$ to reverse as iterative noising function $q(x_t|x_{t-1})$, where $t$ is the number of noising steps and $x_t$ is $x$ after these $t$ steps. In practice noising functions that can be non-iteratively computed based on the desired number of steps, which can be written as $\tilde{q}(x_t|x_0, t)$, are used along with denoising models $\tilde{p}_\theta(x_0|x_t, t)$ trained to remove all noise and return to the original unnoised training example. A single step of denoising is then approximated with equation 2, which also includes the additional conditioning on $C$ we need for this task. Notably, autoregressive models are a subset of discrete diffusion models where the noise function $q$ deterministically masks tokens from right to left as $t$ increases (5).

$$p_\theta(x_{t-1}|x_t, C) \propto \tilde{q}(x_{t-1}|\tilde{p}_\theta(x_0|x_t, t, C), t-1) \tag{2}$$

**Idea Summary.** The goal of the project is to have an even comparison of autoregressive language model and diffusion model performance on the conditioned protein generation task. Both models use the same transformer architecture and are trained on the same subset of UniRef50 representative proteins and attribute tags. Generation is done with and without conditioning to try to measure its impact.

As is common in image and text domains, pretrained public models are used to aid evaluation. ESMFold (14) is used to generate 3D folded protein structures for qualitative evaluation. For qualitative evaluation we will use embeddings from ProtTrans (8), a large protein language model, in two new ways for protein generation.

**Description.** Both ARLMs and DDPMs are implemented as transformers where the encoder encodes conditioning information that the decoder uses to generate protein sequences. The encoder for both transformers takes in a shuffled set of conditioning tokens $C$ with no positional encoding. The input and output of the decoder differs between model types. For the ARLM, the input to the decoder for training is the true protein sequence $x$ with a start token prepended while the target decoder output is $x$ with a stop token appended. Learned positional encodings are added to each amino acid embedding in the input to the decoder. Causal attention masking is used to ensure that decoder output $j$ is equivalent to equation 1 where $i = j$. Top k sampling is used to generate proteins with the ARLM. Based on preliminary testing $k = 10$ and a sampling temperature of $1.0$ are used, but the high variance between runs makes it hard to confidently say these are great choices.

With the DDPM, the input to the decoder is the noised protein $x_t$ and the output is the predicted denoised protein $\hat{x}_0$. The model is a discrete DDPM (D3PM) which uses the absorbing states noise function from Austin et al. (5). With this noise function an additional mask token becomes a possible input to the decoder, and the input is moved towards being noise by setting elements of the protein sequence to this masked state. If the maximum amount of noising and denoising steps is $T$, for $\tilde{q}(x_t|x_0, t)$ the probability of any amino acid in $x_t$ being masked is $(T - t + 1)^{-1}$ for $t \geq 0$.

As is done in ProGen (15), during training the direction of protein strings is randomly flipped and some dropout probability is applied to the conditioning tags. Here a dropout rate of 0.2 was used instead of the 0.4 in ProGen because we had fewer tags per protein since we filtered to just those with 1,000 samples in our subset. If all conditioning tags are dropped out $C$ will be set to just include to token for the root taxa, which is equivalent to unconditional sampling as this is true for all proteins in our data set.

**Implementation.** The models are implemented in Pytorch using the Pytorch Lightning framework. Pytorch Lightning allows for easy training with mixed precision, gradient clipping, Stochastic Weight Averaging, and gradient accumulation. Both the ARLM and D3PM transformers have about 1.1 million trainable parameters. They both have 1 encoder layer and 5 decoder layers with an 8 attention heads, embedding size of 128, and a feed forward dimension of 192. A small version of each was also

trained with about 220,000 parameters by reducing the number of decoder layers to 3, the embedding dimension to 64, and the feed forward dimension to 128.

We used Dask to preprocess the 227 GB UniRef50 protein dataset. Dask parallelized loading the data in chunks and filtering the samples and tags. The Pytorch Lightning dataloaders used with the models handle flipping sequences and dropping out available attribute tags as is done in ProGen.

Embeddings for evaluation will be generated using the ProtTrans T5 XL half precision model. This model was trained with a a Bart-like MLM denoising objective and is faster through the use of float16 half precision. A protein's embedding is the mean of the per amino acid embeddings excluding padding positions.

## 4 Experiments

**Datasets and Tools.** We will be using UniRef50, a representative subset of the larger UniProt consortium dataset. UniRef50 is a subset of UniRef90, in which proteins were clustered into families based on sequence similarity. UniRef50 further reduces the size of the dataset by clustering the UniRef90 families into larger clusters using the representative protein sequences' similarity (19). A representative protein was selected for each cluster and these representative proteins are what we use in our dataset. This ensures that the proteins in the train, validation, and test folds are sufficiently distinct. Each representative protein is marked with its cluster's common taxon (organism of origin) and Gene Ontology (GO) tags, which code for cellular location, protein function, tissue of function, etc. These annotations are used as the conditioning information.

The full UniRef50 dataset contained around 10 million proteins. To reduce the data down to something easier to work with, we filtered to just tags with 1,000 or more proteins and proteins between 50 and 256 amino acids in length with at least one tag. This resulted in 866,384 samples, which were split 80:10:10 into train, validation, and test sets. Our reduced subset includes 172 Gene Ontology tags and 100 taxa for conditioning.

**Baselines.** The ARLM and D3PM models will be compared against random substitution of some percent of the bases in the original protein as is done in ProGen. Here, we compare against 10%, 25%, and 50% substitutions.

For the autoregressive model, we can also compare performance on of a trained model against one that guesses the next base with uniform probability or with the empirical probability from the dataset. Though they are trained on different datasets, we can also compare the perplexity of ProGen and our autoregressive model at least relatively compared to the perplexity for the random baselines.

**Evaluation Metrics.** Both of our main evaluation methods use embeddings from ProtTrans. The first is with the k-nearest neighbors based Improved Precision and Recall (IPR) (12) precision and recall metrics. These compare the generated and real distributions in the embedding space. Precision measures the fraction of generated images that are realistic with respect to the real distribution's approximated manifold, while recall reflects the fraction of that manifold covered by the generated distribution. This does not directly measure the accuracy of the conditioning. It does so only implicitly, in that conditioning should help improve a model's recall of subclasses associated with certain conditioning tags.

K-nearest neighbors (k-NN) in the ProtTrans embedding space is also be used to better evaluate the impact of conditioning using two mutually exclusive attributes. Proteins will be generated using the conditioning tags from the real set of proteins with these attributes and embedded. We will use k-NN with the real set of proteins' embeddings to evaluate how similar generated proteins are to real proteins with that attribute compared to those with the mutually exclusive tag. We can treat this as the problem of classifying the generated proteins with the nearest neighbor real proteins in embedding space and measure performance using the macro-F1 score and the macro average precision (AP).

For the autoregressive language model, we can compare perplexity of the trained ARLM with the perplexity with uniform and empirical probability next amino acid prediction.

**Quantitative Results.** Perplexity can be used to compare how well autoregressive models predict true samples. Perplexity values for the uniform and empirical baselines and the ARLM and ProGen

models on their respective datasets are in Table 1. For our dataset there was only a 0.34 improvement between the uniform and empirical baselines. For the ProGen paper this gap was close to 7, but the uniform and empirical evaluation was not done on an out of distribution test set as it is for all other values in Table 1. Our ARLM models improved over the empirical baseline by 6.51 and 7.23. ProGen improved over the baseline by 4.8 or 0.36 depending on whether the whole test set or a random 20% subset is used for evaluation, but its hard to fairly compare these perplexities against each other as they are based on different datasets.

Table 1: The values in the "this project" column are from the data and models in this paper, while the "From ProGen" column includes the values the ProGen authors reported. The perplexity (PPL) values in the left column are with respect to the whole test set. As this test set is composed of proteins from different clusters than any in the training dataset, this is equivalent to the perplexities for the out of distribution (OOD) test set in ProGen. The first PPL reported for ProGen is over their whole test set of 100k proteins, while the second is for a random subset of 20k.

| **This project** | | **From ProGen** | |
|---|---|---|---|
| **Model** | **PPL** | **Model** | **PPL** |
| Uniform Probability | 22.96 | Uniform Probability | 25.00 |
| Empirical Probability | 22.62 | Empirical Probability | 18.14 |
| ARLM small | 16.11 | ProGen | 13.34, 17.78 |
| ARLM | 15.39 | | |

Table 2: Mean IPR precision and recall of six generation repetitions. The F1 score is the harmonic mean of the mean precision and recall.

| **Generation Method** | **Precision** | **Recall** | **F1** |
|---|---|---|---|
| Rand 10% | 0.926 | 0.959 | 0.942 |
| Rand 25% | 0.387 | 0.373 | 0.380 |
| Rand 50% | 0.240 | 0.010 | 0.019 |
| ARLM uncond | 0.131 | 0.171 | 0.148 |
| ARLM | 0.135 | 0.273 | 0.181 |

For the IPR metrics and the k-nearest neighbors classification with mutually exclusive attributes for the ARLM and the random baselines, proteins were generated six times using each method for 30,000 test set samples. The results for the IPR metrics, in Table 2, show that the ARLM likely falls somewhere between randomizing 25% and 50% of the bases in the original protein. The ARLMs have lower precision on average, but much better recall than when 50% of the input was randomized. I was not able to use a DDPM to get scores much better than zero on the IPR metrics. The best precision was 0.00023 and the best recall was 0.0.

For the k-nearest neighbors classification evaluation, I chose one set of mutually exclusive cellular component tags and one set of molecular function tags. For cellular component we chose tags GO:0016021 and GO:0005737, which indicate the proteins are located and carry out their function in the cell membrane or cytoplasm respectively. For molecular function we used GO:0003677 and GO:0005524 for DNA binding and ATP binding. The 21 nearest neighbors are used for classification and the macro-F1 and macro average precision (AP) used to evaluate how similar the generated proteins are to those that really have the attribute tag. As with IPR, each model was sampled six times (except only once for the DDPM) and evaluation is done with members of the 30,000 sample test set with either tag. The ARLM was able to perform similarly to the 50% random replacement baseline and better than the best DDPM and the ARLM without conditioning information. A concerning outcome for this evaluation method is that proteins with 10% of their amino acids randomly reassigned score as well as or better than the original ground truth proteins. Otherwise, the ARLM performs similarly to randomizing 50% of amino acids.

**Qualitative Results.** For qualitative analysis, 3D protein structures were generated for the ground true, ARLM generated, DDPM, and 50% random substitution protein sequences. 3D structures were generated with ESMFold (14), which was chosen over alpha fold mainly for speed. Blue regions in the figures are where the model is more locally confident in the predicted 3D structure, while more red areas are less confident. The ARLM seemed better at generating larger structures like longer spirals that may make their topology more like the ground true proteins. Visual inspection of

Table 3: k-nearest neighbors classification performance for generated protein sequences using embedded real protein neighbors, where k = 21. The set of real candidate neighbors will be proteins from the 30,000 sample subset of the test set that have one of the two attribute tags being contrasted.

| | Cellular Component | | Molecular Function | |
|---|---|---|---|---|
| **Model** | **Macro-F1** | **Macro-AP** | **Macro-F1** | **Macro-AP** |
| Ground truth | 0.916 | 0.974 | 0.962 | 0.995 |
| Rand 10% | 0.917 | 0.975 | 0.962 | 0.995 |
| Rand 25% | 0.860 | 0.947 | 0.937 | 0.982 |
| Rand 50% | 0.480 | 0.560 | 0.658 | 0.711 |
| ARLM uncond | 0.472 | 0.500 | 0.396 | 0.499 |
| ARLM | 0.509 | 0.557 | 0.646 | 0.752 |
| DDPM | 0.472 | 0.501 | 0.378 | 0.551 |

the protein sequences generated by the DDPM revealed that they were typically shorter and more repetitive than real sequences or those generated by the ARLM.

**Ablative Studies.** The goal of the ablative studies are to measure the impact of the conditioning tags. Both the IPR precision and recall and the k-nearest neighbors between mutually exclusive attributes metrics were used to compare the standard version of the model to one without conditioning ($C$ only includes the tag for the root taxa). For the IPR metrics, removing the conditioning from the ARLM reduced precision slightly and recall more significantly from 0.273 to 0.171 (Table 2). This seems to indicate that removing conditioning impacts models ability to generate realistic proteins less than it impacts it's ability to generate proteins that cover the full real distribution. This intuitively makes sense, as the conditioning should modify what is generated to cover specific parts of the true distribution.

For the k-nearest neighbors evaluation with the ARLM, conditioning improves performance across both metrics for cellular component and molecular function (Table 3). The improvement is larger for molecular function, where macro-AP rises from under 0.50 without conditioning to over 0.75.

## 5 Conclusion and Discussion

This project applied two related classes of generative models, autoregressive language models and discrete deep diffusion probabilistic models to conditional protein generation. Evaluation was done primarily using embeddings from a pretrained model, with the goal of moving evaluation into a space more representative of protein function than sequence similarity or auxiliary measures of realism like folding energy that are typically used.

My concern with our approach is that it may ultimately fall into some of the same pitfalls as MSAs and other sequence similarity approaches that leverage the large available datasets of proteins. These datasets are similar to the ones our embedding model was trained on in that they contain real world proteins. This means the included proteins are unlikely to contain small variations that "break" the intended function, as evolutional pressures would typically lead to these versions being less likely in nature. As the training set does not include examples of these functionally broken proteins, they are outside of the distribution of the model we are using to generate embeddings for evaluation. This out-of-distribution problem is the same reason why models like AlphaFold should not be used to evaluate the impact of point mutations on protein structure (6; 17). For multiple sequence alignments the issue is similar in that there are not many examples of broken proteins to be aligned against, so proteins will likely be aligned with functional proteins even if a significant point mutation changes the function. The hope of using embeddings from a model as we did here was that the model would have a better chance at evaluating the importance of differences in the sequence, but if this requires the model reasoning on things outside of its training distribution I'm more skeptical.

That said, the results from the ablation study comparing the ARLM with and without conditioning using the embedding based metrics seemed to behave as expected. For the IPR metrics, we would expect removing conditioning to impact the models ability to generate proteins that cover the whole true distribution (recall) more than it would impact the ability of the model to generate samples that seem to be covered by the true distribution (precision). This is because the conditioning should

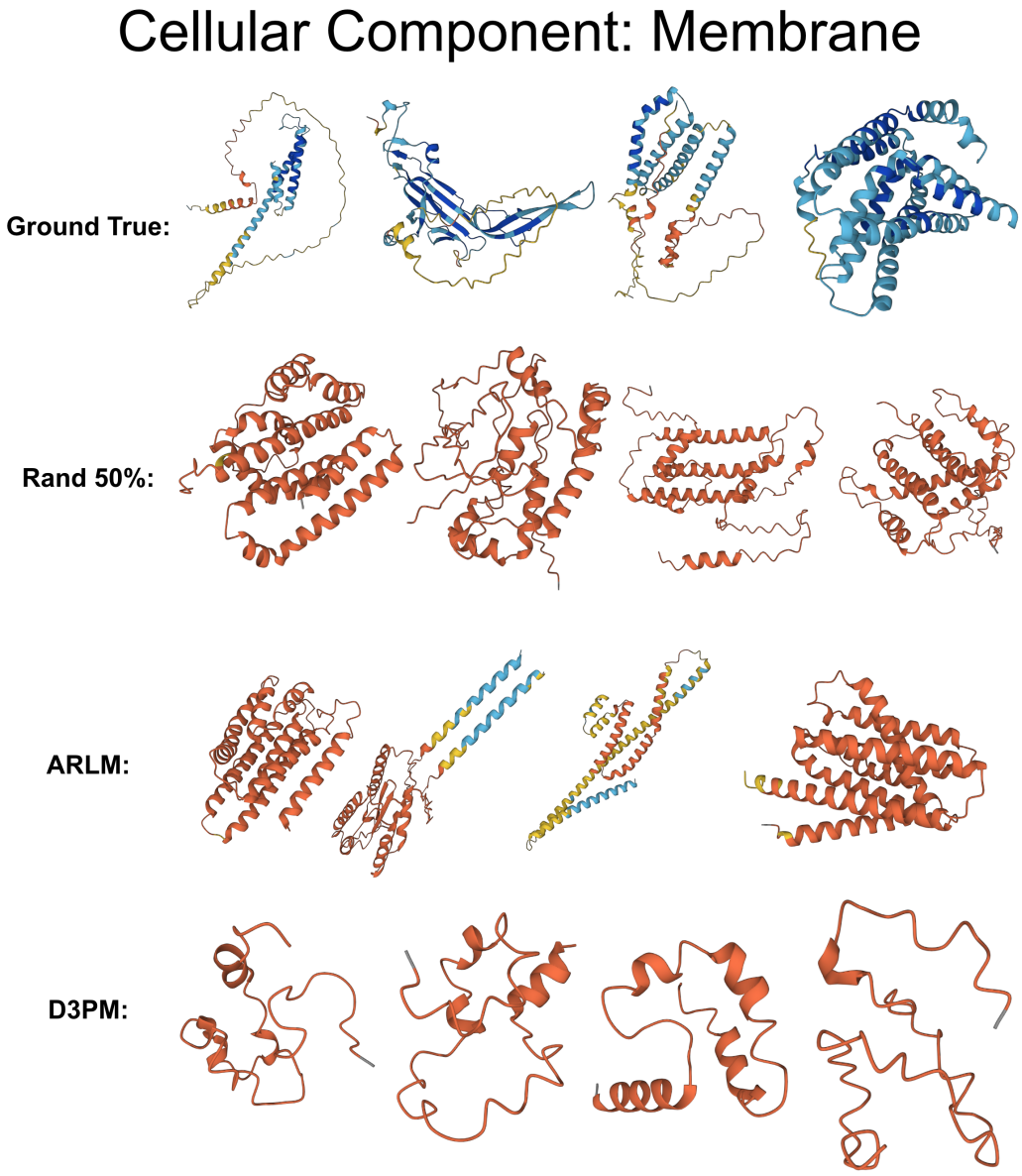


Figure 1: Predicted 3D structure of real, randomly substituted proteins, and ARLM generated proteins with Gene Ontology tag GO:0016021. Blue regions in the figures are where the ESMFold model is more locally confident in the predicted 3D structure, while more red areas are less confident.

identify different parts of the true distribution to cover, while even without specific conditioning the model should be generating proteins that seem like they belong to the true distribution. For the ARLM this was the case. Recall decreased 0.102 (37% change) when conditioning was removed, but precision decreased by just 0.004 (3% change).

The k-nearest neighbors evaluation directly evaluated the impact of the conditioning tags, so we expected the conditioning to improve performance. As expected we saw macro-F1 percent increases of 11.4% for the cellular component and 50.7% for the protein's molecular function with the ARLM. The out-of-distribution issue that makes AlphaFold bad at evaluating point mutations may also explain why proteins with 10% of their amino acids randomly reassigned performed as well as or better than the ground truth proteins for this evaluation method.

If given more time I would continue to work on the training of the DDPM. Currently, the model seems to stop improving around four to six hours into training. The ARLM with the same architecture

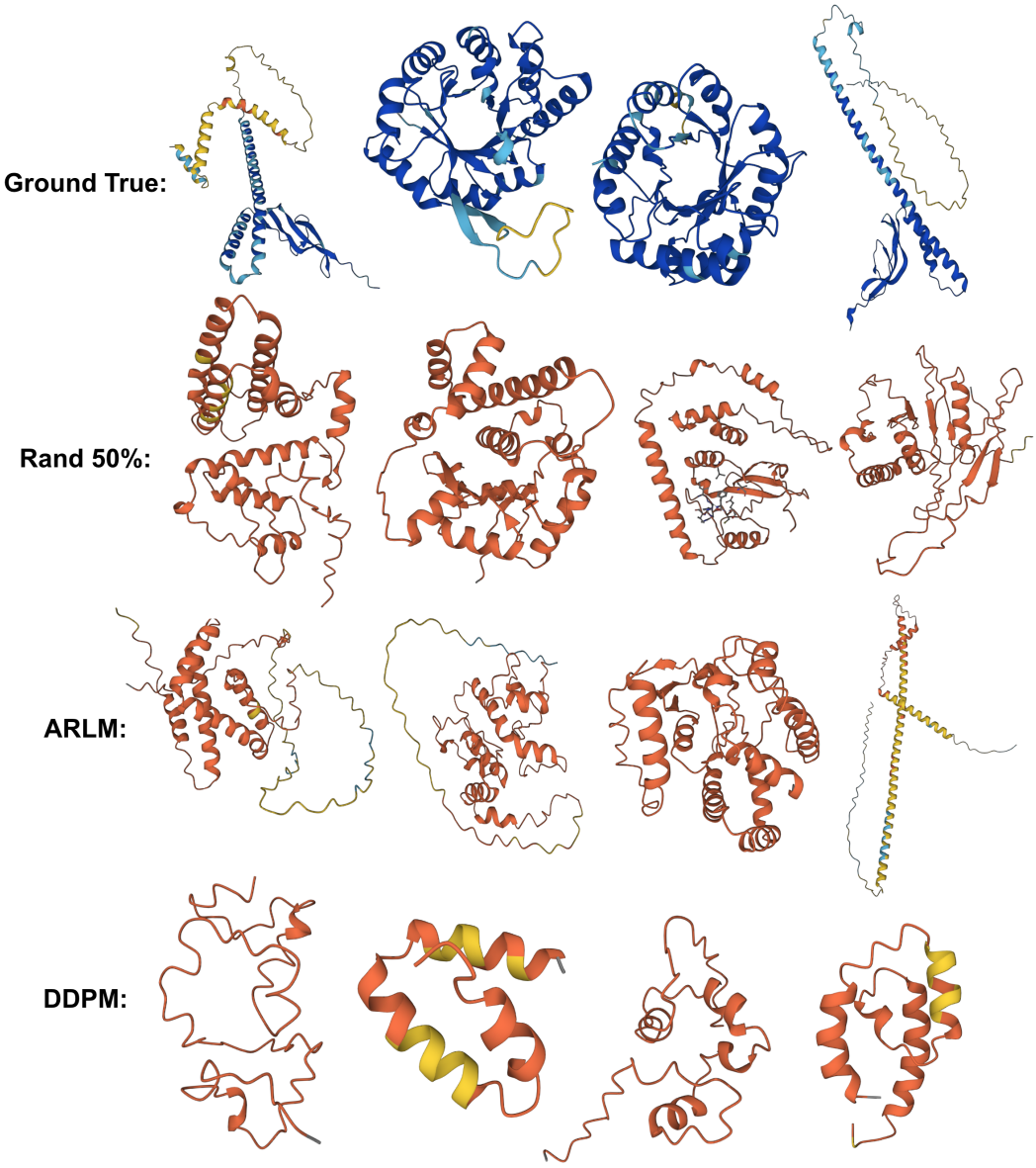


Figure 2: Predicted 3D structure of real, randomly substituted proteins, and ARLM generated proteins with Gene Ontology tag GO:0005737. Blue regions in the figures are where the ESMFold model is more locally confident in the predicted 3D structure, while more red areas are less confident.

took 48 hours to train. There are lots or training hyperparameters I can tune, but because evaluation requires generating proteins, generating embeddings, and then computing the metrics it has been hard to quickly explore all the options.

## References

[1] Rosetta-relaxbb. URL `https://www.rosettacommons.org/docs/latest/application_documentation/structure_prediction/relax`.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. ISSN 0022-2836. doi:

https://doi.org/10.1016/S0022-2836(05)80360-2. URL https://www.sciencedirect.com/science/article/pii/S0022283605803602.

[3] Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. 2022. doi: 10.48550/ARXIV.2205.15019. URL https://arxiv.org/abs/2205.15019.

[4] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, and et al. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000. doi: 10.1038/75556.

[5] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. 2021. doi: 10.48550/ARXIV.2107.03006. URL https://arxiv.org/abs/2107.03006.

[6] Gwen R. Buel and Kylie J. Walters. Can alphafold2 predict the impact of missense mutations on structure? *Nature Structural amp; Molecular Biology*, 29(1):1–2, 2022. doi: 10.1038/s41594-021-00714-2.

[7] Seth Carbon, Eric Douglass, Benjamin M Good, Deepak R Unni, Nomi L Harris, Christopher J Mungall, Siddartha Basu, Rex L Chisholm, Robert J Dodson, Eric Hartline, and et al. The gene ontology resource: Enriching a gold mine. *Nucleic Acids Research*, 49(D1), 2020. doi: 10.1093/nar/gkaa1113.

[8] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing, 2020. URL https://arxiv.org/abs/2007.06225.

[9] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1), 2022. doi: 10.1038/s41467-022-32007-7.

[10] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. 1992. doi: 10.1073/pnas.89.22.10915. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC50453/.

[11] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. 1999. doi: 10.1006/jmbi.1999.3091. URL https://pubmed.ncbi.nlm.nih.gov/10493868/.

[12] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models, 2019. URL https://arxiv.org/abs/1904.06991.

[13] Jin Sub Lee and Philip M. Kim. Proteinsgm: Score-based generative modeling for de novo protein design. *bioRxiv*, 2022. doi: 10.1101/2022.07.13.499967. URL https://www.biorxiv.org/content/early/2022/07/13/2022.07.13.499967.

[14] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[15] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *bioRxiv*, 2020. doi: 10.1101/2020.03.07.982272. URL https://www.biorxiv.org/content/early/2020/03/13/2020.03.07.982272.

[16] Erik Nijkamp, Jeffrey Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani. Progen2: Exploring the boundaries of protein language models, 2022. URL https://arxiv.org/abs/2206.13517.

[17] Marina A. Pak, Karina A. Markhieva, Mariia S. Novikova, Dmitry S. Petrov, Ilya S. Vorobyev, Ekaterina S. Maksimova, Fyodor A. Kondrashov, and Dmitry N. Ivankov. Using alphafold to predict the impact of single mutations on protein stability and function. *bioRxiv*, 2021. doi: 10.1101/2021.09.19.460937. URL https://www.biorxiv.org/content/early/2021/09/20/2021.09.19.460937.

[18] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall, 2018. URL https://arxiv.org/abs/1806.00035.

[19] Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H. Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, 03 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm098. URL https://doi.org/10.1093/bioinformatics/btm098.

[20] Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. 2022. doi: 10.48550/ARXIV.2206.04119. URL https://arxiv.org/abs/2206.04119.