ProtDiff: Function-Conditioned Masked Diffusion Models for Robust Directed Protein Generation

Vishrut Thoutam
Department of Software Engineering
High Technology High School

Yair Schiff
Kuleshov Lab
Cornell University

Sergey Ovchinnikov
SOLab
Massachusetts Institute of Technology

Pranam Chatterjee
Programmable Biology Lab
Duke University

Abstract

Developing function-specific protein sequences remains a critical challenge in drug discovery and protein design. Existing methods, such as structure-based models and protein-text models, face limitations due to inherent biases in structural predictions, the need for extensive fine-tuning, and difficulties in zero-shot generalization to novel functional tasks. To overcome these challenges, we propose ProtDiff, a novel protein sequence diffusion model conditioned on function tokens. While previous works rely on multimodal inputs or fine-tuning, ProtDiff employs a masked diffusion language model (MDLM) formulation with classifier-free guidance to generate protein sequences in a zero-shot manner using a predefined function token vocabulary. During the reverse diffusion process, ProtDiff uses a transformer backbone to iteratively reconstruct the sequences, enabling precise functional adaptations while maintaining stability. By training on diverse protein-function pairs from the InterPro database and incorporating classifier-free diffusion guidance, ProtDiff achieves state-of-the-art performance in both functional adaptation and de novo sequence generation tasks. Our evaluation demonstrates that ProtDiff generates novel, stable protein sequences that adhere to specific functional constraints, performing comparably to autoregressive models with higher parameter counts. These results suggest that ProtDiff not only advances the field of protein design but also opens new pathways for explainable and targeted protein generation, particularly in drug discovery applications.

1 Introduction

Protein design plays a crucial role in drug development, but manual approaches can take years. AI tools have emerged to accelerate this process by generating de novo proteins from existing datasets [24]. There are different protein models, including structure-based models that predict protein structures from sequences [3, 20, 37, 1], aiding in rational design through structural interactions [28]. However, these models have limitations due to structural prediction biases and lack of large-scale validation [6]. Protein language models have shown promise in design tasks, relying only on sequences rather than structures [7]. These models, built on transformer architectures [36], analyze proteins similarly to language in NLP models.

Despite their success, protein language models face challenges in specific tasks. They are often trained using a general task, such as masked language modeling [11]. Base protein language models require extensive fine-tuning for specific property/binding prediction, which necessitates large datasets. In addition, predictions are difficult to validate and optimize due to a lack of interpretability. Prior attempts to bridge text instructions and protein sequences for zero-shot conditional generation have struggled (see Appendix B). Instead, conditioning on function tokens from a predefined vocabulary offers a more promising approach by decreasing the complexity of the instruction space. Earlier methods used transformers with additive embeddings but relied on additional modalities like structure and pLDDT metrics [20], limiting their applicability [15].

Diffusion models offer an alternative for function-conditioned protein generation, simulating discrete mutations and achieving robust results with smaller models. Classifier-free guidance enables training based on function tokens without needing high-performance predictors. Recently, the masked diffusion language model formulation (MDLM) [32] has greatly improved sequence diffusion performance by imposing constraints on the reverse diffusion process. We introduce ProtDiff, a sequence diffusion model conditioned on function tokens. The model uses the MDLM method, progressively masking tokens during forward diffusion and procedurally reconstructing them in the reverse process. ProtDiff trains a transformer backbone model that accepts both function token and sequence embeddings on the MDLM task, using classifier-free guidance for training and sampling. Evaluations of the model show that ProtDiff performs better in functional adaptation and stable de novo sequence generation when compared to a conditioned transformer model and performs similarly to other function-conditioned zero-shot sequence generators with higher parameter counts.

2 Results

In this section, we provide a detailed overview of evaluations done on the proposed ProtDiff algorithm and other relevant baseline models (described in E.3). In addition, we show that ProtDiff generates sequences unique to the original function data while maintaining protein stability and synthesizability (evaluation formulations defined in Appendix E).

The ProtDiff model was trained using parameters and methods explained in Appendix D. A 1000-timestep diffusion process with a 6-layer backbone model and a model dimension of 128 was used for evaluations (see Table 2).

During the evaluation of the model, we create a generated set of proteins using ProtDiff to evaluate using metric learning benchmarks described in E.2. This generated set consists of 10,000 proteins. There are 2,000 proteins of 5 different categories. The first category involves conditioning on a single function token, the second category includes sequences conditioned on two tokens, and so on until 5-token-conditioned sequences are reached[1]. We use this generated dataset to evaluate function matching scores, evaluate de novo generation scores, and show embedding map comparisons across training and generated datasets.

We compare sequence understanding between training, validation, and generated sets. Sequence understanding is determined based on cross-entropy loss on the token level between the original

[1]5 tokens is the mean amount of function tokens within the training dataset, which is why we chose this value when creating our generated dataset

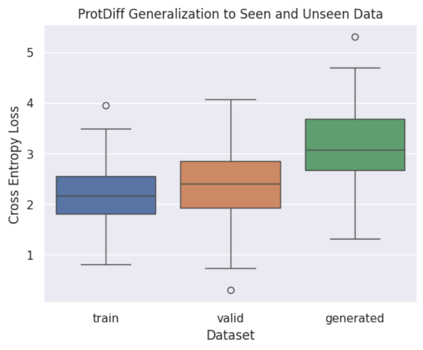

Figure 1: A comparison of cross-entropy losses between ground truth and reconstructed inputs in each of the train, validation an d generated sets show that ProtDiff can generalize to a wide set of proteins. This loss was calculated by taking a random sample of 100 sequences from each of the sets

and diffusion-reconstructed outputs. We find that ProtDiff performs best when reconstructing the training set. Validation metrics show similar performance to training reconstruction loss outputs, albeit slightly higher. Generated set reconstruction shows the highest loss value of all categories, but does not exceed the loss of the training and validation sets greatly. This implies that the generated set is within the bounds of the valid protein distribution, but extends beyond the training set distribution.

2.1 Function Matching

In this section, we compare function-matching scores (formulated in E.2.1) between ProtDiff, ESM3, and a vanilla autoregressive model[2]. We include two metrics: a protein language model-based contrastive evaluation (see Equation 11) and a conditional centroid-based contrastive evaluation (see Equation 12).

Evaluations show that ProtDiff's generations perform better than a vanilla autoregressive predictor in both formulations of the function adaptation task. In addition, ProtDiff shows the best performance at 2-token conditional tasks, decreasing as the number of conditional tokens increases. This occurs due to a natural increase in distribution complexity as more conditional variables are included.

ProtDiff shows worse performance than ESM3 in most tasks, with the exception of 2-token functional adaptation and single-token pLM-based function adaptation. This occurs due to ESM3's higher parameter count (1.4B parameters), which leads to correspondingly higher expressibility. In addition, ESM3 has been trained on more data samples, allowing it to have more accurate predictions. However, ProtDiff shows performance close to ESM3 despite the parameter count difference. This most likely occurs due to ESM3's reliance on more than just a sequence and function track (structure, pLDDT).

2.2 De Novo Generation

Finally, we evaluate ProtDiff on its ability to generate *de novo* proteins (formulated in E.2.2). In Table 1, we show that ProtDiff's generated dataset has high *de novo* scores. This shows that the model generates many sequences that are separate from the original training distribution. In addition, ProtDiff shows the highest performance in stable *de novo* sequence generation in 1 to 4-token settings. In contrast, ESM3 has higher raw *de novo* scores, but performs slightly worse in stable *de novo* scores.

[2]The autoregressive model is of the same architecture and tokenization method as the backbone model leveraged in ProtDiff. We include this to show the advantage of the diffusion process over raw autoregressive generation

Table 1: Table of function adaptation and *de novo* generation scores for ProtDiff, ESM3, and raw autoregressive model. PLM Scores denote the function annotation scores based on a protein language model backbone (ESM2-8M), while contrastive score shows the contrastive parameterization of the evaluation objective. Both raw *de novo* and stable *de novo* scores are included. This shows the estimated distance between training set embeddings and generated set values, while regularized scores dampen this with a stability score. Evaluations are shown for each number of tokens in the generated set (**Note:** Contrastive Score is converted to percentage using the following: $(1 - distance) * 100\%$ and distance is clamped to a maximum value of 1)

Model Name	Task Name	1 Token	2 Token	3 Token	4 Token	5 Token
ESM3	PLM Score	92.9%	91.8%	**90.2%**	*87.2%*	**84.4%**
	Contrastive Score	**93.3%**	92.1%	**91.1%**	**86.6%**	**85.2%**
	Raw *De Novo*	**1.484**	1.739	**1.923**	**2.326**	**2.722**
	Stable *De Novo*	0.903	0.812	0.725	0.433	**0.355**
Autoregressive	PLM Score	89.9%	90.2%	89.0%	85.9%	85.2%
	Contrastive Score	91.1%	90.4%	89.9%	84.7%	82.5%
	Raw *De Novo*	1.302	1.335	1.527	1.593	1.734
	Stable *De Novo*	0.705	0.522	0.459	0.198	0.131
ProtDiff	PLM Score	**93.1%**	**92.0%**	90.0%	*87.2%*	83.2%
	Contrastive Score	92.8%	**92.7%**	90.9%	85.2%	84.6%
	Raw *De Novo*	1.413	**1.778**	1.804	1.928	2.576
	Stable *De Novo*	**0.964**	**0.839**	**0.727**	**0.481**	0.343

This shows that ProtDiff is able to generate more stable sequences while generalizing to the *de novo* sequence space. ProtDiff also performs better than a vanilla autoregressive model on nearly all *de novo* generation benchmarks

3 Conclusions & Future Work

Function-conditional protein generation is important for explainable and stable protein generation. Previous methods, which tune on language-based inputs, do not perform as well in zero-shot function-based generation. As an alternative, we propose ProtDiff as a conditional diffusion model that can generalize to combinatorial sets of function tokens. ProtDiff's masked diffusion language modeling objective along with classifier-free guidance allows for the integration of the transformer architecture as a backbone model. ProtDiff is able to generate stable *de novo* in simple function spaces as well as proper functional adaptation of proteins.

In future works, we aim to increase ProtDiff's model depth to determine its range of protein representation properties. In addition, we plan to apply ProtDiff to directed evolution for protein optimization as well as function and target-conditioned binder design.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, May 2024.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023.

[3] Minkyung Baek, Ivan Anishchenko, Ian R. Humphreys, Qian Cong, David Baker, and Frank DiMaio. Efficient and accurate prediction of protein structure using rosettafold2. May 2023.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[5] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[6] Letícia M. F. Bertoline, Angélica N. Lima, Jose E. Krieger, and Samantha K. Teixeira. Before and after alphafold2: An overview of protein structure prediction. *Frontiers in Bioinformatics*, 3, February 2023.

[7] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, February 2022.

[8] Garyk Brixi, Tianzheng Ye, Lauren Hong, Tian Wang, Connor Monticello, Natalia Lopez-Barbosa, Sophia Vincoff, Vivian Yudistyra, Lin Zhao, Elena Haarer, Tianlai Chen, Sarah Pertsemlidis, Kalyan Palepu, Suhaas Bhat, Jayani Christopher, Xinning Li, Tong Liu, Sue Zhang, Lillian Petersen, Matthew P. DeLisa, and Pranam Chatterjee. Saltamp;peppr is an interface-predicting language model for designing peptide-guided protein degraders. *Communications Biology*, 6(1), October 2023.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[10] Tianlai Chen, Madeleine Dumas, Rio Watson, Sophia Vincoff, Christina Peng, Lin Zhao, Lauren Hong, Sarah Pertsemlidis, Mayumi Shaepers-Cheu, Tian Zi Wang, Divya Srijay, Connor Monticello, Pranay Vure, Rishab Pulugurta, Kseniia Kholina, Shrey Goel, Matthew P. DeLisa, Ray Truant, Hector C. Aguilar, and Pranam Chatterjee. Pepmlm: Target sequence-conditioned generation of therapeutic peptide binders via span masked language modeling, 2024.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and Aurelien Rodriguez. The llama 3 herd of models, 2024.

[13] 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, 2021.

[14] Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. Mol-instructions: A large-scale biomolecular instruction dataset for large language models, 2024.

[15] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. July 2024.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[19] Kanchan Jha, Sourav Karmakar, and Sriparna Saha. Graph-bert and language model-based framework for protein–protein interaction identification. *Scientific Reports*, 13(1), April 2023.

[20] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, July 2021.

[21] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[22] Maxat Kulmanov, Francisco J. Guzmán-Vega, Paula Duek Roggli, Lydie Lane, Stefan T. Arold, and Robert Hoehndorf. Protein function prediction as approximate semantic entailment. *Nature Machine Intelligence*, 6(2):220–228, February 2024.

[23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[24] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic level protein structure with a language model. July 2022.

[25] 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. A text-guided protein design framework, 2024.

[26] Zhiyuan Liu, An Zhang, Hao Fei, Enzhi Zhang, Xiang Wang, Kenji Kawaguchi, and Tat-Seng Chua. Prott3: Protein-to-text generation for text-based protein understanding, 2024.

[27] Claire D. McWhite, Isabel Armour-Garb, and Mona Singh. Leveraging protein language models for accurate multiple sequence alignments. *Genome Research*, July 2023.

[28] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, May 2022.

[29] Geraldene Munsamy, Ramiro Illanes-Vicioso, Silvia Funcillo, Ioanna T. Nakou, Sebastian Lindner, Gavin Ayres, Lesley S. Sheehan, Steven Moss, Ulrich Eckhard, Philipp Lorenz, and Noelia Ferruz. Conditional language models enable the efficient design of proficient enzymes. May 2024.

[30] Zhangzhi Peng, Benjamin Schussheim, and Pranam Chatterjee. Ptm-mamba: A ptm-aware protein language model with bidirectional gated mamba blocks. February 2024.

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[32] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024.

[33] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling, 2024.

[34] Jianlin Su. Roformer: Transformer with rotary position embeddings - zhuiyiai. Technical report, 2021.

[35] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. October 2023.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[37] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung

Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, July 2023.

[38] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.

[39] Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. Should you mask 15

[40] Wenkai Xiang, Zhaoping Xiong, Huan Chen, Jiacheng Xiong, Wei Zhang, Zunyun Fu, Mingyue Zheng, Bing Liu, and Qian Shi. Fapm: Functional annotation of proteins using multi-modal models beyond structural modeling. May 2024.

[41] Yuki Yasuda and Ryo Onishi. A theory of evidence lower bound and its application to super-resolution data assimilation (srda) using conditional variational autoencoders, 2023.

[42] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019.

A Background

In this section, we provide an overview of architectures [17] and modifications used in ProtDiff, including transformer-based protein language models and discrete masked diffusion models.

A.1 Discrete Diffusion

While initial diffusion models were formulated as continuous probability distribution learners through Gaussian diffusion [31], a general framework for diffusion models on discrete datasets, D3PM, was introduced [2]. Unlike the Gaussian forward diffusion used in continuous DDPMs, D3PM uses time-dependent transition matrices Q_t, where each entry in the matrix denotes a transition probability from one discrete state to another corresponding state. Forward transition state propagation is defined by the following rule:

$$q(x_t|x_{t-1}) = Cat(x_t; p = x_{t-1}Q_t) \tag{1}$$

This rule can be summarized into a single computation across a range of timesteps:

$$\overline{Q}_t = \prod_{i=1}^{t} Q_i \tag{2}$$

$$q(x_t|x_0) = Cat(x_t; p = x_0\overline{Q}_t) \tag{3}$$

Many parameterization methods for Q_t have been proposed. The simplest transition matrix formulation is a uniform matrix across the categorical distribution. Other methods leverage a discretized Gaussian distribution. In addition, absorbing state transition matrices are inspired by masked language modeling methods, gradually transitioning all values to an all-encompassing state, such as a `<MASK>` token.

The reverse diffusion process is parameterized to model the posterior of the forward diffusion process at the relevant timestep. This backbone model leverages an evidence-based lower bound loss formulation [41] to improve the reverse diffusion model over time. In contrast to continuous diffusion models, which estimate the noise added during the forward process, discrete diffusion models predict the reconstructed output and compare it to the original output accordingly.

A.2 Masked Diffusion Language Models

D3PMs proposed a general forward diffusion process that can be applied to many types of discrete data (e.g. discrete-color images, text). However, the general discrete diffusion formulation can be altered to increase inductive bias and simplify loss calculations for masked sequence data.

Masked Diffusion Language Models (MDLMs) [32] leverage the absorbing-state forward diffusion process along with specific reverse diffusion parameterization rules to simplify the computation of the loss function and increase model accuracy.

The absorbing state diffusion process is defined as a distribution parameterized by a time-conditioned noise schedule α_t that determines the probability of replacing a token with a mask token $\mathbf{m}$ at each timestep (similar to the beta parameter schedule in DDPMs).[3]

[3]Noise schedules are selected such that all tokens will be masked by the end of all timesteps of the forward diffusion process

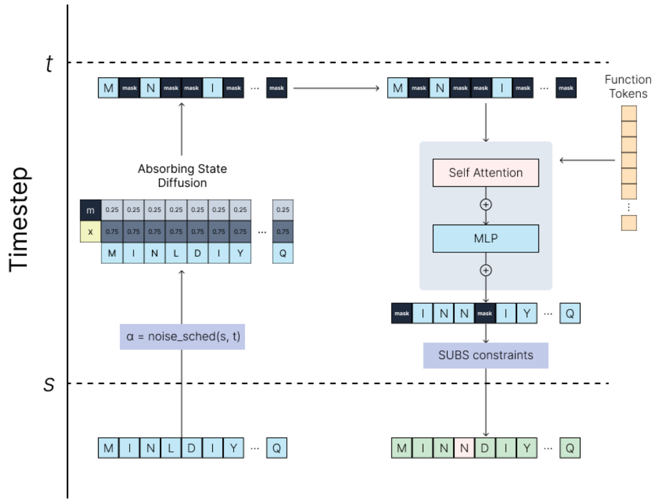

Figure 2: A full diagram of both the forward and reverse diffusion processes within the modified MDLM architecture used for protein-conditional training. This method starts with a full protein sequence and relevant function annotations. The sequence is corrupted through an absorbing state diffusion process with masking probability based on a noise schedule. The reverse process is parameterized by a conditional language model that integrate function tokens. SUBS parameterization constraints are leveraged to simplify loss calculations

$$q(z_t, x) = Cat(z_t; \alpha_t x + (1 - \alpha_t)\mathbf{m}) \tag{4}$$

This forward diffusion process ensures that tokens that are masked are not unmasked during the diffusion process.

The reverse diffusion process[4] $p(z_s|z_t)$ is parameterized by a categorical distribution that enforces specific constraints on the unmasking of tokens.

$$p_\theta(z_s|z_t, x) = \begin{cases} Cat(z_s; z_t) & z_t \neq m \\ Cat\left(z_s; \dfrac{(1-a_s)m + (a_s - a_t)x}{1 - a_t}\right) & z_t = m \end{cases} \tag{5}$$

Where x is defined by a time-conditioned backbone model that reconstructs the original input from the masked version:

$$x = x_\theta(z_t, t) \tag{6}$$

This parameterization (SUBS) adds restrictions on the original discrete diffusion formulation specific to absorbing state diffusion methods. The reverse diffusion process prevents tokens unchanged by the forward diffusion process from being altered[5] and prevents masked tokens from staying masked at the end of the reverse diffusion process[6].

The SUBS parameterization allows for the use of the modified NELBO loss function, a Rao-Blackwellized form of the original D3PM loss that eliminates the reconstruction loss term.

[4]matching the estimated forward diffusion posterior

[5]Logit probabilities are altered after backbone model execution to enforce this constraint

[6]This is done by setting the probability of selecting the `<MASK>` token to 0 in the logits' softmax formulation after the backbone model generates an output

$$\mathbb{E}_{q,t}\left[-\log p_\theta(x|z_{t(0)}) + T\left[\frac{a_t - a_s}{1 - a_t} log\langle x_\theta(z_t, t), x\rangle\right]\right] \tag{7}$$

Masked Diffusion Language Models' simplified parameterizations have shown high performance in language modeling tasks, but have not been applied to protein language modeling. Discrete diffusion models mimic the natural discrete mutations of proteins and can lead to more robust outputs with smaller backbone models compared to a raw autoregressive approach. Due to the increased inductive bias and affinity towards backbone models with lower parameter counts, we leverage a masked diffusion language model for function-conditional generation.

A.3 Protein Language Models

Diffusion models require a robust backbone algorithm to effectively model the reverse diffusion process across timesteps, capturing relevant features within data samples. Transformer models [36], known for their high performance in sequence modeling, have been particularly successful in protein sequence analysis [7]. Protein language models (pLMs), which are transformer models extensively trained on protein sequences, have been applied to tasks such as function prediction [22] and interaction prediction [19], [8], [10]. PLMs are trained extensively on protein sequences in a semi-supervised representation learning manner. The most common training method is masked language modeling [11], where a percentage[7] of tokens are replaced with either a mask token (80%), a random token (10%), or unaltered (10%). The model slowly learns protein representations by predicting these masked tokens.

While many studies augment these models with auxiliary information like structure integration [35], MSA data [27], and post-translational modifications (PTMs) [30], we opt for a sequence-only approach. This choice simplifies the conditional loss bound and avoids the complications of incorporating extra conditional information in addition to function tokens [18].

B Previous Work

In this section, we provide an overview of current methods for functional-conditional protein generation.

B.1 Protein Language Model Augmentation

Protein language model augmentation methods alter vanilla language model architectures to include multimodal adapters. These multimodal adapters integrate text instructions or function token information into a protein sequence's latent representation within the language model.

Methods such as FAPM [40] integrate functional annotations through a querying transformer architecture [23]. This architecture trains an auxiliary transformer [36] and a set of input queries, which interact with protein function data in text format to extract relevant data. The extracted query information is passed to a base protein language model architecture [13]. This architecture can be fine-tuned to detect specific molecular or biological processes [38].

Other methods attempt to directly bridge text and protein modalities. Protein-to-text architectures have been proposed to integrate high-level protein information into large language models. Methods

[7]While most methods choose 20% as a masking ratio, alternate methods have shown high performance at higher masking ratios, nearing 50% [39]

such as ProtT3 [26] propose to integrate protein information by passing hidden states from a protein language model into a cross-modal projection architecture. This information is then added to the language sequence to integrate protein information. Alternate methods leverage text-to-protein architectures [25], which involve projecting text embeddings into a shared embedding space [5] to decode into a protein. While these architectures are highly innovative, they still show low levels of generalization to varied function descriptions in a zero-shot manner.

B.2 Sequence Modification/Augmentation

Alternate methods of function conditional generation involve modifying the sequence input to a language model. Methods such as ZymCTRL [29] leverage control tags [21] to describe different functions. These control tags are defined such that similar functions are tokenized similarly. ZymCTRL has shown generalization to *de novo* protein sequences based on control tags.

Other methods integrate function embeddings on a token level. ESM3 [15] uses a tokenization method that adds token embeddings from multiple tracks, including sequence, structure, and function. Functional embeddings are created by extracting unigrams and bigrams from InterPro annotations. While ESM3 shows the most fine-grained function information integration, it does not perform well if all 3 main tracks are not present (sequence, structure, and function).

B.3 Protein Terminology Fine-Tuning

While most methods involve a protein language model as a main backbone, other methods leverage large natural language models [9]. These methods fine-tune language models on protein terminology and protein sequence datasets. Datasets such as Mol-Instructions [14] have been used to fine-tune medium-sized language models, such as `LLaMA-3-8B` [12]. These models have shown generalization to general protein terminology but not to protein sequence representations due to the lack of expressibility of fine-tuning methods and the lack of varied large-scale protein + text instruction data[8].

C Methods

In this section, we provide an overview of the ProtDiff architecture and alterations from previous models and components used.

C.1 Diffusion Process

The diffusion process used in this model is based on the masked diffusion language model (MDLM) formulation. The original masked diffusion process leveraged small language models[9] as backbone models. Subquadratic models were also evaluated[10]. Similarly, we train a small transformer model from scratch on protein sequences with the ability to accept conditional function information.[11]

[8]In addition, text and protein instruction data is very difficult to synthesize at scale due to the wide range of possible tasks

[9]The original language-based MDLM formulation uses `BERT-base`

[10]DiMamba [33] was used as a backbone during training

[11]In contrast to using a pretrained model, we train a model from scratch to closely fit the requirements of the MDLM process and create support for function tokens during conditional training

The reverse diffusion process is altered to account for function conditional information during training. We use the discrete-time diffusion configuration with a total of 1000 timesteps[12] and implement a log-linear noise schedule[13] to parameterize the forward process[14].

While the initial masked diffusion language model formulation includes time conditioning in backbone models, it has shown little model improvement. This is because time conditioning is implicitly integrated based on the proportion of masked tokens at each timestep. Therefore, we do not integrate time conditioning in our diffusion process and allow the model to implicitly learn timestep embeddings.

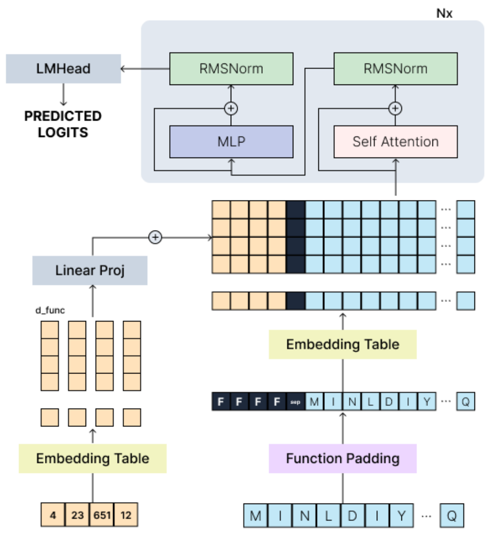

Figure 3: Data flow diagram through transformer pLM backbone model with function token integration. Function tokens are first added as placeholders in the original sequence with a `<SEP>` token between both modalities. The empty function tokens are then populated with embeddings from the function embedding table and projection. The aggregated embeddings are passed as input to the backbone model.

C.2 Diffusion Backbone Model

For the backbone model of the diffusion process, we leverage a transformer-based protein language model architecture. Previous works leverage pre-trained transformer models for masked diffusion language modeling [32]. However, we train a backbone model from scratch with the diffusion process to natively learn function token integration into the embeddings (see Figure 3).

Each transformer block consists of a multi-head self-attention block[15] [4] followed by a residual add [16] and RMSNorm [42] operation. Rotary positional embeddings [34] are used during the attention process to encode relative positional information. After this, an MLP layer with an expansion ratio of 4 and another residual + norm block is added.

[12]Smaller model variations leverage 250 and 500 total timesteps

[13]log-linear noise schedule is parameterized as the following: $a_t = \log(1 - \beta_t)$

[14]MDLM proposes a linear, log-linear, and polynomial-based noise schedule. Through short preliminary experiments, we determined that the log-linear noise formulation was most performant for the proposed task

[15]biases were not used on Q, K, V, and output projections in the attention process. model dimension d_{model} was also scaled down to d_{attn} to decrease computational requirements of the attention process

Algorithm 1 MDLM Training with Classifier-Free Guidance

```
Require: $x_\theta$: model, $p_{uncond}$: probability of unconditional training, $p_{data}$: data distribution, $\mathbf{T}$: maximum timesteps, $\mathcal{L}$: diffusion loss
repeat
    $(\mathbf{x}_0, c) = p_{data}(x|c)$
    $\mathbf{c} \leftarrow \varnothing$ with probability $p_{uncond}$
    $t = \mathbf{U}(1, T)$
    $a_t = \mathbf{noise}(t)$
    $\mathbf{z}_t = \mathbf{Cat}(\mathbf{z}_t; a_t\mathbf{x} + (1 - a_t)\mathbf{m})$
    $\mathbf{z}_0 = p(z_0|z_t) = \mathbf{SUBS}(x_\theta(\mathbf{z}_t, t, c))$
    $L = \mathcal{L}(z_0, x_0)$
    $x_\theta \leftarrow \mathbf{AdamW}(L, lr)$
until converged
```

C.3 Function Embeddings

The proposed diffusion method conditions on functional information to provide a form of control and explainability to generated proteins. Functional tokens are extracted from InterPro functional annotations, where each unique unigram and bigram (excluding stopwords) is added to a global vocabulary[16]. This method was chosen instead of using InterPro annotations as tokens directly to allow more fine-grained generations and higher capacity for *de novo* protein generation without being restricted by existing protein families/groups that the full InterPro annotations are based on.[17]

To integrate functional information conditional to the current sequence, we add a set of empty function tokens at the start of each sequence passed to the transformer. This creates two regions separated by a `<SEP>` token: function tokens and sequence tokens. The empty function tokens are then populated with embeddings from an embedding table and MLP projection specific to function tokens (see Figure 3). This allows for the generation of compatible coembeddings between sequence and function[18].

While many conditional diffusion models opt for cross-attention within backbone models, we use a token-level integration due to the simplicity of implementation while still being able to generalize to the de novo protein space [29].[19]

To encourage proper use of function token embeddings, we implement a classifier-free diffusion guidance method. During training, we sample a fixed probability of conditionality p_{uncond}. Based on this probability, we either provide relevant function tokens or leave empty function tokens in the sequence input (see Algorithm 1).

During inference, we use classifier-free guidance-based sampling along with the masked diffusion language model formulation. Classifier-free guidance sampling uses the following rule to modify the output such that it accounts for conditional and unconditional objectives during training:[20]

$$\tilde{x}_t = (1 + w)x_\theta(z_t, c) - wx_\theta(z_t, c) \tag{8}$$

[16] total vocab size: **58641**

[17] This method of extraction is analogous to the function tokenization method proposed in ESM3. This is used to allow direct comparisons to ESM3 as a baseline model during evaluations

[18] function embeddings are added to the empty function token embedding to allow the sequence embedding table to separate functional and sequential embedding spaces accurately

[19] function tokens are added only during the reverse diffusion process and not included in the forward diffusion methods or in loss function calculations.

[20] a value of 0.5 is used for w in all conditional sampling procedures

Table 2: Table of backbone model configurations that underwent training, fine-tuning, and evaluation processes with comparison to baseline models with similar parameters

d_{model}	n_{layers}	d_{attn}	p_{cond}	T
64	3	32	0.25	250
64	3	64	0.25	500
128	6	64	0.25	500
128	6	128	0.25	1000

This rule is modified from the original sampling rule, which uses a noise model ϵ_θ (based on continuous diffusion model settings). However, masked diffusion language models are instead parameterized to predict the reconstructed output x_θ specific to the timestep.

D Training Method

D.1 Training Dataset

Most protein language models train on open protein databases using the masked language modeling framework. However, ProtDiff requires functional annotations. We train ProtDiff on a subset of the InterPro database, which contains proteins along with relevant function annotations. We extract data from the InterPro database using its public API, and preprocess it using the following techniques.

1. We extract all InterPro annotations using a script that accesses each entry from the API
2. We remove stopwords from the annotations and extract relevant, repeated unigrams and bigrams as a vocabulary of function tokens[21]
3. We extract a set of proteins and their InterPro annotations from the InterPro API, and match the relevant annotations to their keywords
4. We pre-tokenize all sequences and function tokens based on index in a provided vocabulary file[22]

We extracted a dataset of 80,000 proteins[23] of variable length. During batched training, we pad sequences to a predefined maximum length[24] and pad function token sections to the length of the sequence with the most function tokens. In addition, we ensure a diversity of all InterPro annotations during extraction by looping through each InterPro annotation to extract from, extracting auxiliary annotations along with the specified token.

D.2 Model Dimensions

A total of 4 models were trained with various model dimensions and diffusion parameters. We train at model dimensions of 64 and 128 and attention dimensions of 32, 64, or 128 depending on the overall

[21]We use the same function tokens as ESM3 to allow for easy comparison to a provided pre-trained model during evaluation steps

[22]This is done to avoid tokenization re-computation delay across training runs

[23]See computational constraints

[24]A maximum length of 1024 is used for all models, with larger proteins being truncated to save computational resources

model size[25]. We also train with a different number of layers depending on the model dimension and leverage an increasing amount of timesteps based on the model dimension.

We train the specified models with the same data loader seed to prevent invalid comparisons between models. While we do train all models on the full dataset and the same number of gradient steps, evaluations provided are based on the backbone model with a dimension of 128 and 1000 timesteps, as it shows the best performance of all 4 backbone models in the diffusion formulation.

D.3 Loss, Optimizers, Schedulers

We leverage the NELBO loss proposed in [32], which equates to an average of masked language modeling losses per token.[26] During training, we use a gradient clip value of `1.0`.

We use the Adam optimizer without weight decay and beta values of `(0.9, 0.999)`.

The original MDLM implementation uses a warmup scheduler that starts from a learning rate of 0 and increases across 2500 gradient steps. ProtDiff leverages the same scheduler during its training loop.

D.4 Hyperparameter Selection

We swept through the following learning rates during our training process: `1e-3`, `8e-4`, `3e-4`, `1e-4`, `3e-5`,`1e-5`. A learning rate of `3e-4` performed best for all model sizes.

We train our model on a physical batch size of 8 along with gradient accumulation across every 4 iterations. A maximum sequence length cap of 1024 was used to prevent memory overflows on large sequences[27]. Both padding and truncation were used to achieve this overall sequence length, leading to a total of 32768 logical amino acid tokens being processed per gradient step (excluding function tokens).

D.4.1 Computational Constraints

ProtDiff was trained in a distributed data-parallel (DDP) fashion across two NVIDIA T4 GPUs. Each batch was split across versions of models on each GPU[28].

E Evaluation Methods

Function-conditional protein generative modeling in the sequence space does not have any prominent benchmarks with reference to a predefined set of non-sequential function tokens. In this section, we propose two evaluation methods for these models and denote a list of compatible baseline models that we compare to. In addition, we propose a function prediction model and a stability prediction model to be used during the benchmarking process as a control across all models.

[25]We choose attention dimensions at half the original model size and the full original model size for both d_{model} configurations

[26]While the loss function simplifies to a cross-entropy formulation, we leverage the original implementation, which was generalized to a D3PM loss formulation

[27]10.12% of sequences exceed this maximum length

[28]GPU compute was provided by Kaggle's notebook services

E.1 Evaluation Dataset

The evaluation dataset for ProtDiff and other baseline models is generated in a combinatorial fashion. Each data item consists of a set of specific function keyword tokens sampled randomly from the list of function tokens. The total size of the evaluation dataset is 10,000 items. This includes five categories of function token lists, each having respectively 1, 2, 3, 4, and 5 tokens per sample.

Multiple function token list sizes test the model's ability to generalize to specific conditional distributions within the full protein space. While it is easier for the model to generalize to 1-token conditional distributions, specific higher-token distributions require more constraints and test the model's ability to generate fine-grained proteins with many required features.

E.2 Benchmarks

In this section, we propose a set of benchmarks for evaluating the ProtDiff algorithm and other baseline models. This consists of a 2-part formulation to evaluate both function conformation and stable unique protein generation.

E.2.1 Function Matching Score

The function matching score measures the accuracy of conditional protein generation based on integrated function token information. Higher function-matching scores show higher performance in generating sequences within the specified function-conditional distribution based on information supplied to the backbone model.

Many generations from protein generative models do not have robust ground-truth function data to evaluate the function matching score. Instead, we develop a contrastive learning-based evaluation objective that integrates positive and negative conditional class information. Both positive and negative function token scores are leveraged in this manner to prevent protein functional side effects while prioritizing important functional attributes

The positive function matching score is computed using the following, given a scoring model $\mathbf{m}_\theta$ and a set of P positive function tokens from the function token set F such that $P \subset F$.

$$\mathcal{L}_{pos}(x, P|P \subset F) = \frac{1}{|P|} \sum_{p}^{P} \left[\mathbf{m}_\theta(x, p)\right]^2 \tag{9}$$

The negative function matching score is computed using the following equation given the above model and a set of N negative function tokens. This is modified from the positive function matching score to exclude the squaring term, such that functional side effects are not penalized as severely as positive function matching scores. This allows room for error in the case that related function tokens are not considered in the input list of tokens.

$$\mathcal{L}_{neg}(x, N|N = F \setminus P) = \frac{1}{|N|} \sum_{n}^{N} \mathbf{m}_\theta(x, n) \tag{10}$$

The function matching score is a sum of these individual positive and negative scores.

$$\mathcal{L}_{fm} = \mathcal{L}_{pos} - \mathcal{L}_{neg} \tag{11}$$

The scoring model m_θ takes a protein sequence and a function token label as input and outputs a similarity score, such that lower scores signal higher similarity to the relevant function token. This formulates the evaluation task as a metric distance computation in a shared log-embedding space.

We use two parameterizations of the m_θ evaluation model. One is a protein language model[29] fine-tuned on function token classification given a protein input. The fine-tuning process is done through Low-Rank Adaptation (LoRA) on the same dataset used for training ProtDiff.

In addition, we parameterize m_θ as the distance from a set of conditional clusters. The centroid c_f for each function token f is computed as the average embedding[30] of all sequences with the function token present in the training data.

$$m_\theta(x, t) = \|x - c_f\|^2 \tag{12}$$

We provide results using both parameterizations of the function adaptation model to ensure generation validity occurs due to the data itself without being influenced by errors/drawbacks within a fine-tuned protein language model.

E.2.2 Stable *De Novo* Generation Score

In addition to the function matching score, we compute a *de novo* generation score based on the deviation from previously seen protein inputs during training. This score determines a model's ability to generalize to the protein space external to the existing set of proteins[31]. This dataset is computed using embeddings from a protein language model[32].

Our proposed method of *de novo* generation evaluation is based on a set of k reference points from the training dataset. These reference points are computed as centroids of each cluster found in the embedding space of the training dataset. We compute k different clusters using a K-Means Clustering (KMC) algorithm and leverage these values as reference points.

The *de novo* generation score is computed as the average Euclidean distance between the generated protein and each reference point. Higher scores in this category determine higher deviation from the known protein embedding space, and ability to generate sequences in unexplored protein spaces.

Given a set of reference points R such that $|R| = k$, we compute the *de novo* generation score as the following:

$$\mathcal{L}_{dn}(x, R) = \frac{1}{|R|} \sum_{r}^{R} \|x - r\|^2 \tag{13}$$

The *de novo* generation score is clamped at a specified upper value l and normalized to a mean of 0 and standard deviation of 1. This method uses a standard scaler between 0 and the upper bound value across all raw *de novo* generation scores.

We balance this score with the protein stability score, which determines the feasibility and synthesizability of a protein. The protein stability score is determined by a fine-tuned protein language model

[29] We leverage ESM-2-8M as a classification model due to ease of fine-tuning and evaluation efficiency

[30] Using ESM2-8M with average pooling

[31] We formulate this as the proteins seen during training

[32] ESM-2-8M is used for the *de novo* generation scoring algorithm

s_θ that predicts the normalized[33] ΔTm value of a protein. The protein language model is fine-tuned on data from FireProtDB, given the sequence as input, as a regressor on output ΔTm values.

The protein stability score acts as a regulator of the *de novo* generation score, encouraging explorations separate from the existing protein space while still enforcing constraints on protein feasibility. These scores are combined using a simple addition to create the stable *de novo* generation score.

$$\mathcal{L}_{sdn}(x, R) = \min\left(\frac{1}{l|R|}\sum_{r}^{R}\|x - r\|^2, 1\right) - s_\theta(x) \tag{14}$$

Note: All datasets used for fine-tuning of evaluation models are public datasets and relevant licenses were taken into account. In addition, we make all evaluation datasets and models public for evaluations of future protein generative models.

E.3 Baseline Models

Many current function-conditional protein generators are not defined by the set of fine-grained function tokens used in ProtDiff. However, ESM3 is compatible with these function tokens. We leverage a pre-trained version of ESM3 as a baseline generative model to compare to[34].

In addition, we train a vanilla autoregressive model with function tokens as a baseline to compare to ProtDiff. The autoregressive model has the same architecture as ProtDiff's backbone model, allowing for a comparison between raw autoregressive and diffusion-based conditional generation.

[33]Score is normalized using Sigmoid to place between 0 and 1

[34]ESM3 (1.4B parameters) has a much higher parameter count than ProtDiff (9M parameters), which can lead to result imbalances. This occurred due to limitations in training infrastructure