Classifier Free Latent Diffusion for Antibody Design

Joshua Gilligan
Department of Mathematics
ETH Zürich

Supervised by
Dr. Valentina Boeva
Department of Informatics
ETH Zürich

Abstract

Latent diffusion has proven successful in conditional generative AI, especially in image generation, for producing varied yet controlled samples. Such a method could also prove useful for functional Computational Protein Design. While sequence based diffusion for protein design is computationally more efficient than structure based approaches, these methods have been focused on discrete diffusion. However, such methods are susceptible to rounding errors and limited in terms of conditional design. Latent diffusion promises the elimination of rounding errors and greater potential for design prompting. One area with significant potential is the design of new antibodies.

We show that a latent diffusion model over the ESMFold embedding space can be used to generate T-Cell Receptor Sequences. Our approach can successfully reconstruct fully noised sequences (96.84% Residue Accuracy and 3.67 Minimum Edit Distance). We extend this method towards conditional denoising with the goal of designing TCR sequences with high binding affinity towards a target antigen. Conditional generation of *de novo* antibodies demonstrate up to 42.08% match to known binding TCRs in the VDJdb.

1 Introduction

Diffusion Denoising networks have shown great success in image, music and video generation [19]. They represent a new paradigm for generative AI, one which brings varied yet controlled samples with greater fidelity than GANs or VAEs [6]. Recently, these models have been applied to the problem of Computational Protein Design. Two major classes of models have been introduced, structure based diffusion models and sequence based diffusion models [25], [20], [1].

Structure based diffusion applies the diffusion denoising framework to a 3D point cloud representing the C_α backbone of the polypeptide chain. At inference time, a 3D Gaussian sample or a sample from a prior distribution, is then denoised until the backbone structure of a protein is produced [25] . The sequence of the generated protein can be inferred from the structure through inverse folding techniques, such as ProteinMPNN [5]. One major problem for structure diffusion is the *Many-To-One* property of structure to sequence, where each sequence represents a distribution of possible folds [12]. Furthermore, protein sequences with intrinsically disordered regions (IDRs) or hyper-variable regions are not well reconstructed by structure only methods [1].

Sequence diffusion works directly on the amino acid sequence of the protein and therefore can maintain some of the unique properties not inferred by structure. Discrete diffusion models, such as EvoDiff, perform forward masked corruption based on a transition matrix and learn a backwards transition kernel [1] . While this method can model IDRs, discrete diffusion is limited in its ability

for conditional generation. At each stage, a *rounding error* is introduced - some elements of the sequence are still corrupted while others are fixed. Since the diffusion is not performed over a latent space, classifier based diffusion has to guide the loss. However, semi-corrupted sequences may make the classifier task difficult, as previously realised in text generation tasks [8].

A possible solution would be the application of Latent Diffusion. Introduced originally for image generation [19], the input sample is encoded and continuous diffusion is performed over this latent space representation. Then a decoder is trained to predict the original image from the denoised representation. One benefit of Latent Diffusion is that the embedding space can be a joint embedding space, such as CLIP [17]. Conditional generation can therefore be prompted during the denoising process using the same embedding space. In image generation, this allows for conditional generation based on a text prompt or based on images of a similar style. In a similar way as CLIP jointly embeds text and image, potentially a joint embedding space for structure and sequence bridges the gap between the two above approaches. Furthermore, this allows for greater fidelity in prompting while avoiding *rounding errors*. We could use this joint embedding space to conditionally generate for desired functional properties or infill with respect to other proteins in a complex.

Recently, Protein Language Models (PLMs) have seen increased development. Typically, such PLMs are trained using analogous training tasks as Large Language Models (LLMs), primarily Masked Language Modelling. At train time, residues are masked and the model predicts the missing residue [9]. Potentially, the masked language modelling task allows for the model to understand local structure elements of proteins. The ability for such models to recognise structural characteristics was confirmed in the development of ESMFold, where the attention weights of the model could predict protein contact maps [9] [23]. We believe that such a structural aware embedding space could be used for a latent diffusion framework for protein design.

2 Method

2.1 Latent Diffusion

We consider the general case for a diffusion process over some data space $x \in R^{h_d}$, the goal of which is to sample from the distribution $x \sim p_\theta(X)$. We follow the original formulation for the Diffusion Denoising Probabilistic Model Framework (DDPM) [10]. We assume this distribution decomposes as a Markov chain across latent variables $X_T, ...X_0$ such that $X_T \sim N(0, 1)$, a standard normal distribution. This decomposition gives us the following

$$q(x_{1:T}|x_0) = p(x_T)\Pi_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{1}$$

In the DDPM framework, we train a forward diffusion network p_θ to perform this step-wise denoising. At sample time, we can sample any Gaussian $Y_T \sim N(0, 1)$ and denoise to Y_0 such that Y_0 is now a representation of a possible new generation.

We consider an analogous task, where this generation step is aided by sample from some prior distribution $Y_T \sim \pi$. Generation from prior distributions help produce samples more characteristic of the data, although at the expense of diversity. They have shown success before in computational protein design tasks, [13], to avoid bad local minima in the beginning of the denoising process. We construct π by taking the mean representation of a sample of known sequences and fully noise over T steps. This ensures that the prior distribution still allows for general samples.

2.2 Architecture

The ESMDiff architecture has three main components: an encoder, the denoising network and the decoder.

Encoder For the latent space encoding, we use ESM2-T6-8M. As demonstrated in [23] and [9], structural characteristics can be inferred from the latent space. We hypothesise that using the pre-trained ESM protein language model allows for diffusion over a structure aware embedding space. ESM2-T6-8M produces a per-residue embedding with dimension 320, leading to an embedding size of $N_{tcr} \times 320$ per sequence, where N_{tcr} is the sequence length.

As noted in [4], diffusion over this latent space is improved with a reduced bottleneck dimension. We train a fully connected encoder to reduce this to a bottleneck dimension of 64, resulting in an embedding size of $N_{tcr} \times 64$.

Decoder The decoder is a feed forward network pre-trained on varied protein family sequences with weights frozen at inference time. The decoder takes the representation and predicts logits for each residue position over the possible residue labels. In returning a predicted sequence, we take the maximum logit for each label. However, beam search or other more sophisticated language model decoding strategies could be used. The encoder decoder structure is seen in Figure 1.

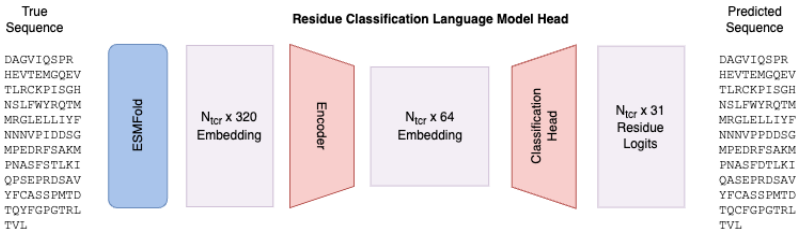

Figure 1: Encoder Decoder Architecture

Denoising Network We perform the latent diffusion using a stack of Multihead Attention Blocks followed by a fully connected layer with residual connections, a ResNet Block (see Figure 2). We consider the typical Attention Network design as demonstrated in [22]. Time and peptide embeddings are fed into separate MLPs. These predict scale and shift matrices, each $N_{tcr} \times 64$, which adjust the predicted noise depending on conditional generation prompting and time sampled. We perform diffusion over 500 time steps with noise schedule $\beta_t \in [0.0001, 0.015]$.

2.3 Training and Sampling

Train Time For each sequence, we sample a time step randomly, t_0, and noise according to the noise schedule. The noised representation can be decomposed as the original representation and noise matrix ϵ_{t_0}. The denoising network predicts this noise matrix from the noised representation as in Figure 2. In unconditional mode, the denoising network takes as input the noised representation and the time embedding, which is the standard sinusoidal time embedding [18]. In conditional mode, the peptide ESM embedding is passed through an MLP and then added to the time embedding. As per [2], we use conditional mode for 60% of train time.

Sample Time At inference, we perform the sampling algorithm as specified in [10]. First, we sample a $N_{tcr} \times 64$ joint Gaussian, or from a prior distribution. We assume that this is a true representation fully noised to time T. For each of the noising steps, we predict the ϵ_{t_0}, rescale by the β_t term and subtract the noise from the representation. Unlike the original sampling algorithm in [10], we don't re-introduce noise into the sampling step due to training stability. After T denoising steps, we decode the fully denoised representation using the decoder.

2.4 Data

PFam In training the encoder and decoder model, we use a large class of protein families represented in the PFam database [14]. As demonstrated in [3], training the decoder with a larger set of

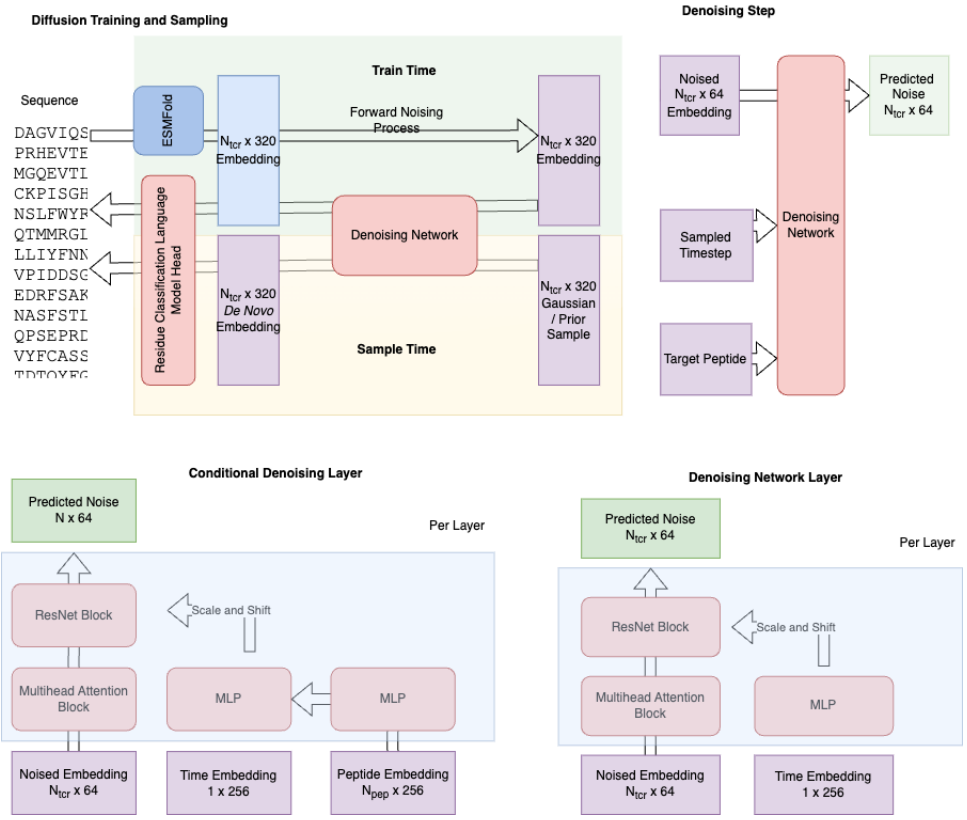

Figure 2: Diffusion Information Flow and Denoising Network

sequences ensures that the model accurately translates the ESM embedding back to the residue labels. Training the decoder solely on TCR sequences may limit the model's expressivity in generating novel sequences.

VDJdb For the diffusion network, we take 800 known TCR Antigen pairs with known binding specificities from VDJdb [16]. We use the full TCR sequence and each TCR-peptide pair has at least 15 associated TCR sequences. We first select only positive binded pairs and perform an 80:20 train to test split.

3 Evaluation Metrics

We consider two class of metrics:

- **Reconstruction Metrics**: Measure of how the network denoises known TCR sequences
- **Conditional Generation**: Measure of how the network conditionally generates TCRs with high binding affinity to a known peptide, completely *de novo*

3.1 Reconstruction Metrics

MSE of Predicted Noise The training metric is the L^2 norm of the predicted to true noise of the latent representation at a sampled time point.

$$MSE(\hat{\epsilon_t}) = ||\epsilon_t - \hat{\epsilon_t}||^2 \tag{2}$$

Residue Accuracy RA Similar to [4], we calculate the percentage of residues correctly reconstructed by the network. At each time step, we sample a number of sequences, noise them to the maximum noising step and iterate through each stage of the network (a full forward pass). We then predict the residue tokens using the token classifier and calculate the reconstruction accuracy. Since the network also predicts the end of sequence tokens and padding - this metric is a measure of both residue reconstruction and sequence length prediction.

$$AR(\hat{seq}) = \frac{\sum_{a_i \in seq} \mathbb{1}_{\hat{a_i}=a_i}}{|seq|} \tag{3}$$

Minimum Edit Distance MED We calculate a constant cost edit distance for each sequence based on the Levenshtein distance. The metric penalises each insertion, deletion or residue edit with a constant penalty. We consider this representative of insertions, deletions and mutations within the protein encoding gene. Furthermore, a reconstructed sequence out of alignment is penalised by a constant whereas recovery accuracy would return 0%.

3.2 Generation Metrics

Amino Acid Recovery Accuracy @ *k* samples We conditionally sample k TCR sequences prompted by the target peptide. We then report the highest RA to the known matched TCR sequence in the VDJdb.

Amino Acid Recovery Accuracy @ *k* samples Similar to the above, we calculate the MDE to the closest known binded TCR sequence in the VDJdb.

4 Results

ESMDiff Successfully Denoises known sequences We demonstrate that latent diffusion models are able to accurately reconstruct noised representations. We report high recovery accuracy and low minimum edit distance for fully denoised samples.

It is possible that the decoder is noise resistant - regardless of the denoising, the sequence would be recovered just from the decoder. This would demonstrate a collapse in the denoising network task and thus generation would no longer be accurate. We show that this is not the case, with a decoder only model reporting a significantly lower RA and a higher MED. We report the change in RA and and MED from including the denoising network in Table 1.

		RA	Δ RA	MED	Δ MED
Decoder Only	Train	$46.85 \pm 3.72\%$	-	64.40 ± 4.84	-
	Test	$49.92 \pm 4.39\%$	-	60.0 ± 5.71	-
Full Network	Train	$\mathbf{97.10 \pm 1.82\%}$	50.30%	$\mathbf{3.47 \pm 2.11}$	-60.93
	Test	$\mathbf{96.84 \pm 1.83\%}$	46.94%	$\mathbf{3.67 \pm 2.02}$	-56.4

Table 1: Reconstruction Accuracy and Edit Distance across Decoder Only and Full Network Models

ESMDiff Generates Conditional and Uncondition generations ESMDiff unconditionally generates sequences characteristic of TCRs

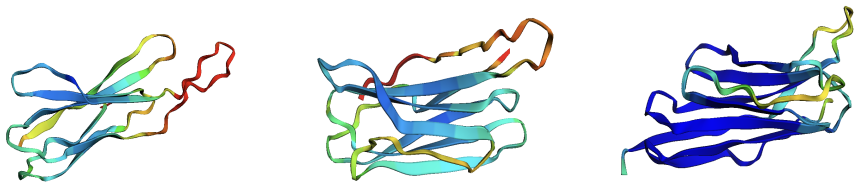

Figure 3: Unconditional *de novo* TCR Structures, Predicted AlphaFold Structures, Coloured by pLDDT

While unconditional generation is a heuristic validation that the model understands key structural components of TCRs (see Figure 3), the conditional generation task is more important from an immunotherapy perspective. We also demonstrate the ability for conditional generation. Conditionally generating from a prior distribution using the peptide embedding demonstrates large portions of the TCR sequence is reconstructed. Furthermore, we demonstrate the benefit of including the prior distribution and report the increase in RA and MED, see Table 2.

		RA K = 10	Δ RA K = 10	RA K = 50	Δ RA K = 50	RA K = 100	Δ RA K = 100
Prior Sampling	Train	36.93 ± 5.48%	30.75%	42.26 ± 3.72%	33.87%	**42.76 ± 2.59%**	33.4%
	Test	34.75 ± 6.26%	28.17%	40.27 ± 3.89%	32.37%	**42.08 ± 4.25%**	33.46%
No Prior Sampling	Train	6.18 ± 1.53%		8.39 ± 1.74%		9.36 ± 1.48%	
	Test	6.58 ± 1.71%		7.90 ± 1.65%		8.62 ± 1.51%	

Table 2: Conditional Generation Accuracy using Prior and Non-Prior Sampling

The use of the prior distribution in sampling does drastically increase the RA and lower the MED for the conditional generation task. Since this prior was constructed only from the train data, there is a possibility sequences would be representative of the train data. However, we see in Table 2, that the model doesn't show signs of over-fitting. It should be noted, that potentially the use of the prior eliminates some possible sequences otherwise generated through complete Gaussian sampling as specified in [10].

Figure 4: Conditional Generation for a target peptide

5 Conclusion

ESMDiff explores a potentially new paradigm for functional, computational protein design. After surveying the limitations of structural and sequence diffusion models, our approach remedies such limitations and allows for complex conditional generation.

We demonstrate that ESMDiff has the ability to successfully reconstruct noised protein sequences with high residue recovery. ESMDiff can unconditionally generate sequences characteristic of TCRs and can recover large portions of known binding TCRs prompted only by the peptide embedding.

As PLMs are trained on a greater range of property and structure prediction tasks, latent diffusion represents an extensible framework for computational protein design. Potentially PLM embeddings allow for reliable property predictions such as solubility, theromostabilitiy and binding affinity. This would allow for greater breadth of conditions for generating samples.

Future work could include specific architectural changes for the antibody design task. We recognise that certain regions (such as the CDR3 loop) have a greater importance for binding affinity and are hyper variable [21]. Potentially a 2 stage architecture approach could be used. Other work could include a more sophisticated decoding strategy - potentially implementing from the language modelling framework. Finally, further scaling could improve results, i.e. remove the need for the prior, allow for the original formulation of the sampling algorithm and in turn generate a greater range of novel samples.

References

[1] S. Alamdari, N. Thakkar, R. v. d. Berg, A. X. Lu, N. Fusi, A. P. Amini, and K. K. Yang. Protein generation with evolutionary diffusion: sequence is all you need. Pages: 2023.09.11.556673 Section: New Results.

[2] T. Capelle. Training classifier free diffusion models.

[3] T. Chen, P. Vure, R. Pulugurta, and P. Chatterjee. AMP-diffusion: Integrating latent diffusion with protein language models for antimicrobial peptide generation.

[4] T. Cohen and D. Schneidman-Duhovny. Epitope-specific antibody design using diffusion models on the latent space of ESM embeddings.

[5] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using ProteinMPNN. 378(6615):49–56. Publisher: American Association for the Advancement of Science.

[6] P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis.

[7] J. Dunbar, K. Krawczyk, J. Leem, T. Baker, A. Fuchs, G. Georges, J. Shi, and C. M. Deane. SAbDab: the structural antibody database. 42:D1140–D1146.

[8] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong. DiffuSeq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9868–9875. Association for Computational Linguistics.

[9] B. Hie, S. Candido, Z. Lin, O. Kabeli, R. Rao, N. Smetanin, T. Sercu, and A. Rives. A high-level programming language for generative protein design. Pages: 2022.12.21.521526 Section: New Results.

[10] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models.

[11] D. N. Kim, A. D. McNaughton, and N. Kumar. Leveraging artificial intelligence to expedite antibody design and enhance antibody–antigen interactions. 11(2):185. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[12] T. J. Lane. Protein structure prediction has reached the single-structure frontier. 20(2):170–173. Publisher: Nature Publishing Group.

[13] K. Martinkus, J. Ludwiczak, K. Cho, W.-C. Liang, J. Lafrance-Vanasse, I. Hotzel, A. Rajpal, Y. Wu, R. Bonneau, V. Gligorijevic, and A. Loukas. Abdiffuser: Full-atom generation of in vitro functioning antibodies, 2024.

[14] J. Mistry, S. Chuguransky, L. Williams, M. Qureshi, G. Salazar, E. L. L. Sonnhammer, S. C. E. Tosatto, L. Paladin, S. Raj, L. J. Richardson, R. D. Finn, and A. Bateman. Pfam: The protein families database in 2021. 49:D412–D419.

[15] A. Neto. What is latent diffusion in AI?

[16] A. Omer, O. Shemesh, A. Peres, P. Polak, A. J. Shepherd, C. Watson, S. D. Boyd, A. M. Collins, W. Lees, and G. Yaari. VDJbase: an adaptive immune receptor genotype and haplotype database. 48:D1051–D1056.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision.

[18] N. Rogge and K. Rasul. Annotated diffusion model.

[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models.

[20] A. Shanehsazzadeh, M. McPartlon, G. Kasun, A. K. Steiger, J. M. Sutton, E. Yassine, C. McCloskey, R. Haile, R. Shuai, J. Alverio, G. Rakocevic, S. Levine, J. Cejovic, J. M. Gutierrez, A. Morehead, O. Dubrovskyi, C. Chung, B. K. Luton, N. Diaz, C. Kohnert, R. Consbruck, H. Carter, C. LaCombe, I. Bist, P. Vilaychack, Z. Anderson, L. Xiu, P. Bringas, K. Alarcon, B. Knight, M. Radach, K. Bateman, G. Kopec-Belliveau, D. Chapman, J. Bennett, A. B. Ventura, G. M. Canales, M. Gowda, K. A. Jackson, R. Caguiat, A. Brown, D. Ganini Da Silva, Z. Guo, S. Abdulhaqq, L. R. Klug, M. Gander, E. Yapici, J. Meier, and S. Bachas. Unlocking *de novo* antibody design with generative artificial intelligence.

[21] I. Springer, N. Tickotsky, and Y. Louzoun. Contribution of t cell receptor alpha and beta CDR3, MHC typing, v and j genes to peptide binding prediction. 12:664514.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need.

[23] R. Verkuil, O. Kabeli, Y. Du, B. I. M. Wicky, L. F. Milles, J. Dauparas, D. Baker, S. Ovchinnikov, T. Sercu, and A. Rives. Language models generalize beyond natural proteins. Pages: 2022.12.21.521521 Section: New Results.

[24] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. D. Bortoli, E. Mathieu, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek, and D. Baker. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. Pages: 2022.12.09.519842 Section: New Results.

[25] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. De Bortoli, E. Mathieu, S. Ovchinnikov, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek, and D. Baker. De novo design of protein structure and function with RFdiffusion. 620(7976):1089–1100. Publisher: Nature Publishing Group.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

- [x] I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

- [ ] I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

- [ ] I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Classifier Free Latent Diffusion for Antibody Design

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Gilligan	Joshua

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zürich, 1 April 2024	

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard