
DiMA-Based Protein Design: Generating Antimicrobial Peptides Using Diffusion Models

John Maris Elias Georgoulis

University of Crete & FORTH
Graduate Programme: Data Analysis & Machine-Statistical Learning

Deep Generative Models
June, 2024

Contents

Introduction	2
0.1 Motivation	2
0.2 Related work	3
0.3 Baseline Models	3
0.3.1 EvoDiff	3
0.3.2 SeqDesign	3
0.3.3 ProteinGAN	3
0.3.4 nanoGPT	4
1 Problem Definition	4
1.1 Quality-Diversity Tradeoff	5
1.1.1 Evaluation Metrics	5
2 Proposed Methodology	6
2.1 Diffusion on LLM Representations of Protein Sequences	6
2.1.1 Training Phase	6
2.1.2 Noise Schedule & Self-Conditioning	7
2.1.3 Decoder	8
2.1.4 Diffusion Architecture	8
3 Results	10
3.1 AFDB & SwissProt Dataset	10
3.2 Antimicrobial Peptides (AMPs) Dataset	12
3.2.1 AlphaFold-2 Visualization	12
3.2.2 UMAP Projection	13
3.2.3 Classification Experiments	14
3.2.4 pLDDT & pTM Performance	14
4 Manual	15
5 Conclusions	15
References	16



Abstract

The generation of protein sequences with specific properties is a pivotal task in computational biology and bioengineering, with significant implications for medicine, biotechnology, and materials science. Traditional protein design methods are often slow and costly, relying on experimental procedures and heuristics. The advent of deep , particularly the use of language models for biological sequences, has revolutionized this field. The primary paper we are referencing [7] introduces DiMA, a novel method that leverages protein language models (pLMs) and diffusion models to generate high-quality, diverse protein sequences. DiMA employs the ESM-2 language model to embed protein sequences into a continuous space and applies diffusion processes to improve the quality, diversity, and functional relevance of the generated sequences.

Keywords: Protein Sequence Generation, Diffusion Models, Protein Language Models, Antimicrobial Peptides, ESM-2, AlphaFold-2, Bioengineering, Synthetic Biology, Sequence Embeddings.

Introduction

Proteins, the fundamental building blocks of life, are traditionally represented by their linear amino acid sequences and their three-dimensional (3D) structures. The sequence of amino acids dictates the protein's 3D configuration, which in turn defines its function. Despite the sophistication of current models that generate 3D protein structures, they often rely on limited experimental data [3], which can introduce biases and restrict coverage to specific protein families.

In this project, we explore the capabilities of the DiMA [7] (Diffusion on Language Model Embeddings) model for generating antimicrobial peptides (AMPs), which are vital in combating a range of pathogens including bacteria, viruses, fungi, and parasites. These peptides play a crucial role across various organisms' innate immune systems and are especially valuable in the fight against antibiotic-resistant infections.

DiMA leverages the ESM-2 [4] protein language model to embed protein sequences into a continuous space refined through a diffusion process. This approach enhances the generation of protein sequences by effectively capturing complex sequence relationships and balancing quality with diversity, which are challenging for traditional methods.

Training DiMA on AMP data enables us to generate peptides with potential antimicrobial properties. This study demonstrates the practical application of the DiMA model in protein design and synthetic biology, showcasing the utility of deep generative modeling in addressing critical biomedical challenges.

0.1 Motivation

The motivation for developing DiMA stems from the pressing need to enhance the generation of protein sequences with specific properties, such as antimicrobial peptides. Traditional experimental approaches for protein design are not only labor-intensive and costly but also limited in their capacity to explore vast sequence spaces. Computational methods offer a promising alternative, yet they often grapple with achieving a balance between sequence quality and diversity. Additionally, many existing computational models fall short in capturing the intricate dependencies and patterns inherent in protein sequences, which are crucial for functional and structural fidelity.

Advances in deep generative AI, particularly the use of protein language models like ESM-2, have shown significant promise in understanding and predicting protein structures and functions. These models can capture complex relationships within sequences, providing a rich representation that can be leveraged for sequence generation. However, integrating these embeddings into a



generative framework that can produce high-quality, diverse, and biologically relevant sequences remains a challenge.

The integration of diffusion models with PLM embeddings represents a novel approach to overcome these limitations. Diffusion models, which have been successful in other domains like image and text generation, offer a robust mechanism for generating high-quality data by iteratively refining noisy inputs. By applying these principles to the embeddings produced by pLMs, DiMA aims to generate novel antimicrobial peptides that are not only diverse but also functional, addressing the critical need for new antimicrobial agents in the fight against antibiotic-resistant infections.

0.2 Related work

Some related work in the field of protein sequence generation involves the integration of advanced machine learning models, such as diffusion models and language models, to enhance the quality and diversity of generated sequences. Meier et al. (2022) [6] introduce a framework that employs continuous diffusion for protein design, showcasing how diffusion models can generate high-quality protein sequences by iteratively refining noisy representations. Another relevant study is by Lin et al. (2024) [15], which presents TaxDiff, a taxonomic-guided diffusion model for protein sequence generation. This approach integrates taxonomic information to guide the diffusion process, enhancing the biological relevance and diversity of the generated sequences. The use of taxonomic data as an additional layer of information helps in producing sequences that are more aligned with natural evolutionary patterns. Wang et al. (2024) [12] explore the use of transformers in diffusion models for protein generation. Their study demonstrates how transformer architectures, known for their capability to capture long-range dependencies in data, can be effectively combined with diffusion processes to improve the quality and functionality of the generated protein sequences. This integration leverages the strengths of both transformers and diffusion models, providing a robust framework for protein design.

0.3 Baseline Models

The DiMA model is evaluated against four baseline models: the AR transformer (NanoGPT), AR CNN (SeqDesign), GAN (ProteinGAN), and the categorical diffusion model (EvoDiff), all trained from scratch on the same datasets as DiMA.

0.3.1 EvoDiff

EvoDiff [1] is a generative diffusion model trained on large-scale evolutionary protein data. It includes two forward processes: EvoDiff-OADM, an order-agnostic autoregressive diffusion, and EvoDiff-D3PM, a discrete denoising diffusion probabilistic model. The model innovatively corrupts sequences to simulate mutations, which are then denoised by a neural network, enabling the generation of high-quality, diverse protein sequences.

0.3.2 SeqDesign


SeqDesign [11] uses an autoregressive model with a residual causal dilated convolutional neural network architecture. This includes six blocks of nine dilated convolutional layers, enhanced with weight normalization and layer normalization, making it capable of managing variable sequence lengths and complex mutation effects.

0.3.3 ProteinGAN

ProteinGAN [10] features a generator and a discriminator, both integrating ResNet blocks and three convolution layers. The network architecture also includes batch normalization, leaky ReLU activations, and a self-attention layer. Spectral normalization is used across all layers to ensure stable training. The model has been trained on a dataset of bacterial malate dehydrogenase sequences, demonstrating that a significant fraction of generated sequences exhibit solubility and wild-type catalytic activity.



0.3.4 nanoGPT

NanoGPT  employs a streamlined architecture for efficient training of medium-sized GPTs. It is optimized for rapid development cycles and supports functionalities like loading pretrained GPT-2 weights.

1. Problem Definition

The primary problem addressed in the referenced paper is the generation of protein sequences with specific properties, focusing on enhancing the quality, diversity, and biological relevance of these sequences. Traditional methods for protein design are labor-intensive and costly, limiting the exploration of extensive sequence spaces. Moreover, existing computational models often fail to capture the intricate dependencies and patterns within protein sequences, leading to suboptimal designs.

DiMA aims to solve this by utilizing the ESM-2 protein language model to embed sequences into a continuous space, followed by refining these embeddings through a diffusion process. This approach effectively balances quality and diversity, improving the overall relevance of the generated sequences.

In this project, we build upon the foundation laid by DiMA to generate antimicrobial peptides. Specifically, we focus on the AMP data to address the urgent need for new antimicrobial agents capable of combating antibiotic-resistant infections. Our training and validation datasets are sourced from `LAMP_pretrain_train_positive.fasta` and `LAMP_pretrain_val_positive.fasta`, respectively¹.

By applying the DiMA model to these datasets, we aim to generate novel peptides with potential antimicrobial properties. To mathematically define the transformer model used in DiMA, consider the following components:

1. **Embedding Layer:** Each protein sequence $X = (x_1, x_2, \dots, x_n)$ is converted into a continuous representation using the ESM-2 language model:

$$E(X) = (e_1, e_2, \dots, e_n)$$

where $e_i \in \mathbb{R}^d$ is the embedding of the amino acid x_i .

2. **Transformer Layers:** The transformer processes these embeddings through multiple layers, each consisting of self-attention and feed-forward networks:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where $Q = K = V = E(X)$.

The output of the attention mechanism is then passed through a feed-forward network:

$$\text{FFN}(H) = \max(0, HW_1 + b_1)W_2 + b_2$$

where H is the output from the attention layer, and W_1, W_2, b_1, b_2 are learned parameters.

3. **Diffusion Process:** The embeddings are refined using a diffusion process defined by:

$$z_t = \sqrt{\alpha_t}z_0 + \sqrt{1 - \alpha_t}\epsilon_t$$

where z_0 is the initial embedding, α_t is a noise schedule, and ϵ_t is Gaussian noise.

4. **Denosing Model:** A neural network parameterized by θ predicts the original embedding from the noisy version:

$$\hat{z}_0 = f_\theta(z_t, t)$$

The model is trained to minimize the mean squared error between \hat{z}_0 and z_0 .

¹Available in this [repository](#).

1.1 Quality-Diversity Tradeoff

The Quality-Diversity Tradeoff in protein generation concerns balancing the accuracy (quality) and variability (diversity) of generated protein sequences. For quality, metrics like pLDDT (predicting the accuracy of structural predictions) and TM-score (measuring structural similarity between generated and target proteins) are crucial. These evaluate how well a model predicts protein structures that align with known, functional conformations. For diversity, metrics such as Sequence Diversity (SeqD) and Maximum Mean Discrepancy (MMD) are used to assess how varied the generated sequences are, ensuring the model can generate a wide range of functional proteins without overfitting to a narrow set of templates.

1.1.1 Evaluation Metrics

The performance of the generated antimicrobial peptides is evaluated using several key metrics:

- **Perplexity (PPL):**

$$\text{PPL}(X) = \exp \left(-\frac{1}{n} \sum_{i=1}^n \log P(x_i | x_{<i}) \right)$$

- **pLDDT (predicted Local Distance Difference Test):**

$$\text{pLDDT} = \frac{1}{N} \sum_{i=1}^N \text{accuracy}(i)$$

- **TM-score (Template Modeling score):**

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

where L is the length of the peptide, d_i is the distance between residues i in the predicted and reference structures, and d_0 is a distance scaling factor.

- **BLAST Identity:**

$$\text{BLAST Identity} = \frac{\text{Number of matching amino acids}}{\text{Sequence length}} \times 100$$

- **Fréchet ProfT5 Distance (FPD):**

$$d(X_1, X_2)^2 = \|\mu_1 - \mu_2\|^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2})$$

where $X_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $X_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$.

- **Maximum Mean Discrepancy (MMD):**

$$\text{MMD}_k^2(X, Y) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n [k(x_i, x_j) + k(y_i, y_j) - 2k(x_i, y_j)]$$

where k is the kernel function, typically a radial basis function (RBF).

- **1-Wasserstein Optimal Transport (OT):**

$$\text{OT}(\text{Dataset}, \text{Query}) = \text{avg}(\text{pairwise Levenshtein distances between optimal pairs})$$

- **Sequence Diversity (SeqD):**

$$\text{SeqD}_{\text{absolute}} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{\text{levenshtein}(s_i, s_j)}{\max(\text{len}(s_i), \text{len}(s_j))}}{|G|(|G| - 1)} \times 100$$

where G represents the generated sequences and $\text{levenshtein}(\cdot, \cdot)$ is the Levenshtein distance between sequences.



2. Proposed Methodology

The following figure illustrates the complete process of training and sequence generation:

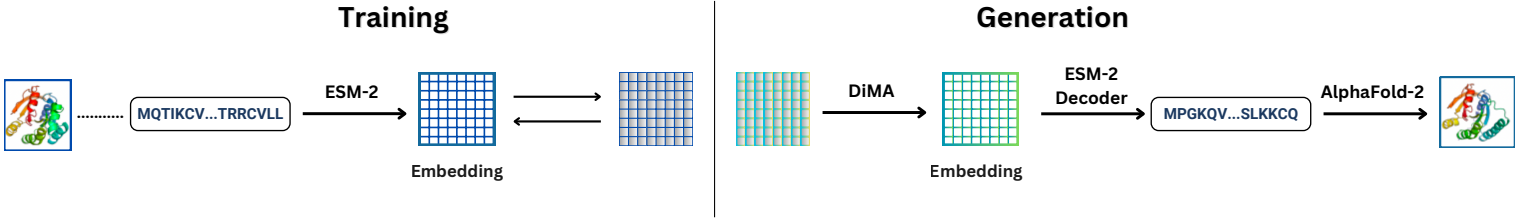


Figure 1: Overview of the proposed diffusion model on pLM embeddings for protein sequence generation: During training, ESM-2 encodes amino acid sequences into continuous representations. The diffusion model is trained to reconstruct corrupted embeddings. In inference, starting from random Gaussian embeddings, iterative refinement generates a protein embedding, which the decoder then maps to an amino acid sequence.

2.1 Diffusion on LLM Representations of Protein Sequences

2.1.1 Training Phase

In the training phase a pre-trained transformer-based pLM is utilized, specifically **ESM-2**, as an encoder. This version of ESM-2 (esm2_t6_8M_UR50D²) is a lightweight model with around 8 million parameters and an embedding size of 320. It was pre-trained on a masked language modeling task using 65 million amino acid sequences, achieving high performance in various structure prediction tasks.

The encoder maps the amino acid sequences $y = [y_1, \dots, y_s]$ of length s to latent vectors $x = [x_1, \dots, x_s]$ in $\mathbb{R}^{s \times d}$, where $x = \mathcal{E}(y)$ and $d = 320$. Then dimension normalization is applied to ensure that each component of a vector in x has zero mean and unit variance, denoted as $z_0 = \text{Normalize}(x)$. This transformation adapts the discrete protein input to a Gaussian diffusion framework.

A continuous denoising network is trained, $\hat{z}_\theta(\cdot)$, to recover z_0 from z_t defined as:

$$z_t = \sqrt{\alpha_t} z_0 + \sqrt{1 - \alpha_t} \epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$ and $t \sim U[0, 1]$.

The objective function for training the denoising network is:

$$\mathcal{L}(\theta) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I), t \sim U[0, 1]} \|z_0 - \hat{z}_\theta(z_t, t)\|^2$$



During both the training and sampling phases a **Variance Preserving Stochastic Differential Equation** (VP-SDE) is solved. This type of SDE is used in the context of continuous diffusion models for generative purposes. Mathematically, it can be defined as follows:

1. Forward SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)} dw_t$$

where $\beta(t)$ is a time-dependent noise schedule, x_t is the state at time t , and w_t is a standard Wiener process.

- **Training Phase:** During training, the model learns to predict the noise added to data at various timesteps. This involves solving the forward SDE to add noise and training the neural network to denoise the data by approximating the score function $\nabla_{x_t} \log p_t(x_t)$.

² '6' stands for the number of the transformer layers, '8M' represents the model's parameters & 'UR50D' is the corresponding training dataset. the github code for the ESM  can be found [here](#) 

2. Reverse SDE:

$$dx_t = \left(-\frac{1}{2}\beta(t)x_t - \beta(t)\nabla_{x_t} \log p_t(x_t) \right) dt + \sqrt{\beta(t)} dw_t$$

where $\nabla_{x_t} \log p_t(x_t)$ represents the score function, which is approximated by a neural network trained to denoise the perturbed data³.

- **Sampling Phase:** During sampling (also known as inference or decoding), the reverse-time SDE is solved iteratively. Starting from random noise, the model iteratively refines the noisy data through multiple timesteps until it generates a clean sample.

The ‘reverse’ method constructs the reverse-time SDE used during the generative process, often implemented with Euler-Maruyama or other numerical solvers.

2.1.2 Noise Schedule & Self-Conditioning

During the training process, it is identified by the authors that conventional noise schedules, such as the linear scheduler utilized by Song et al. (2020) and Ho et al. (2020), as well as the cosine scheduler proposed by Nichol Dhariwal (2021), are not optimal for the protein domain despite their effectiveness in the image domain. These traditional schedules often result in diffusion models exhibiting reconstruction loss curves with extensive regions of low magnitude, indicating suboptimal training efficiency.

This inefficiency is primarily due to the high discreteness of the protein latent space, which makes the reconstruction of z_0 from z_t straightforward for a considerable duration, leading to inefficient learning. To address this issue, the noise schedule proposed by Hooeboom et al. (2023), referred to as Simple Diffusion (SD) is employed.

The SD noise schedule is defined as:

$$\alpha_t = \frac{1}{1 + d^2 \tan^2(\pi t/2)}$$

where d is a hyperparameter controlling the rate of the schedule. A larger value of d results in a higher data corruption rate.

Recent advancements in sequence generation are leveraged by the implementation of the **self-conditioning** technique proposed by Chen et al. (2022). Typically, a denoising network predicts \hat{z}_θ using the latent variable z_t and the timestep t as inputs. Self-conditioning enhances this process by utilizing the predicted \hat{z}_0 from a previous timestep s for the estimation of \hat{z}_0 at the current timestep t , where $t < s$. During iterative sampling at inference, the prediction \hat{z}_0 is already computed from the previous timestep, eliminating the need for additional model launches at inference. However, this requires modifying the training process to ensure the diffusion model can effectively exploit the additional input \hat{z}_0 .

For instance, in generating a sequence of numbers like [2, 4, 6, 8, 10], the model uses each generated number as input for the next step. Starting with 2, it then uses 2 to help generate 4, and subsequently uses both 2 and 4 to generate 6, ensuring consistency and accuracy throughout the sequence.

During a standard training iteration, the timestep t is sampled from a uniform distribution $t \sim U[0, 1]$. Initially, in half of the cases, no additional input is provided to the model, setting $\hat{z}_0 = \emptyset$. In the remaining cases, \hat{z}_0 is estimated as $\hat{z}_\theta(z_t, t, \emptyset)$, where \emptyset is a zero vector. Subsequently, \hat{z}_0 is computed using the previously predicted estimation $\hat{z}_\theta(z_t, t, \text{SG}[\hat{z}_0])$, where $\text{SG}[\cdot]$ denotes the stop-gradient operation. The final estimation is used to compute the loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I), t \sim U[0, 1]} \{ \|z_0 - \hat{z}_\theta(z_t, t, \text{SG}[\hat{z}_0])\|^2 \}$$

This training procedure enables sampling with zero self-conditioning at the first iteration of generation. Additional conditioning is implemented on the previous estimation to optimize the diffusion model for enhanced sequence generation.

³The code can be found on the [diffusion_dynamic_sde.py](#) file.

2.1.3 Decoder

The proposed architecture incorporates the decoder from the ESM-2 model, which was pre-trained alongside the encoder using masked language modeling objectives. However, it was observed that further training of the decoder enhances the accuracy of reconstructing amino acid sequences from the latent representations x during inference. The decoder architecture consists of two linear layers and an activation function, which collectively refine the sequence reconstruction process.

2.1.4 Diffusion Architecture

Our model utilizes a 12-layer Transformer with 16 attention heads and a hidden size of 320 as the backbone for the diffusion model. To ensure effective denoising diffusion in the context of protein-related data, specific modifications were made. One significant change is the incorporation of time embedding into each transformer block, achieved by applying a linear projection before summation in each block. This approach has proven effective for time conditioning. Additionally, long skip connections are integrated, inspired by the U-Vit architecture, further enhancing the model's performance.

Here's a visualization of the PDM architecture:

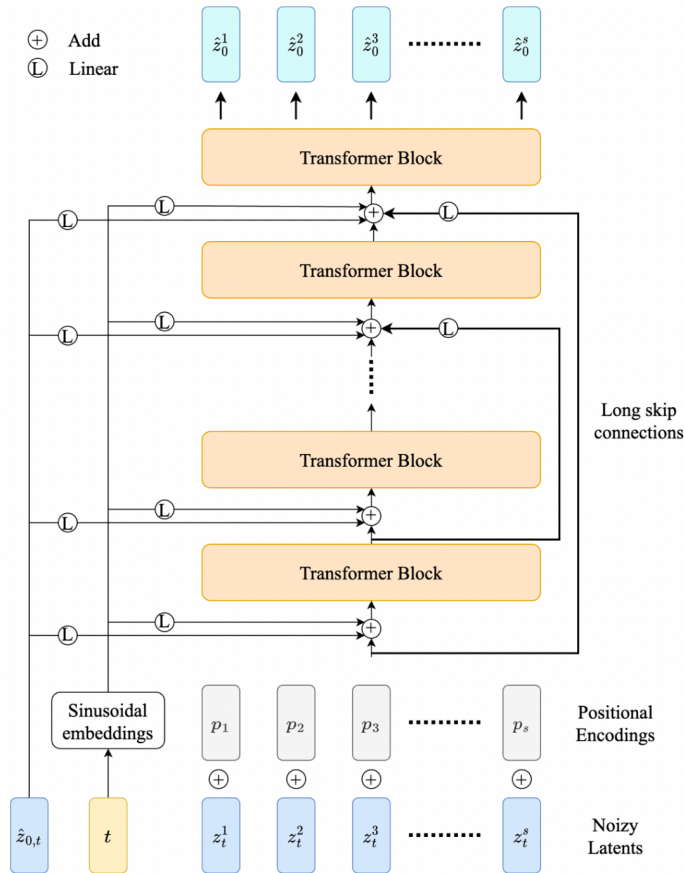


Figure 2: Architecture of the diffusion model [7].

Furthermore, the subsequent Figure [3] provides a comprehensive diagram; illustrating the entire procedure within the context of our specifically adjusted architecture.

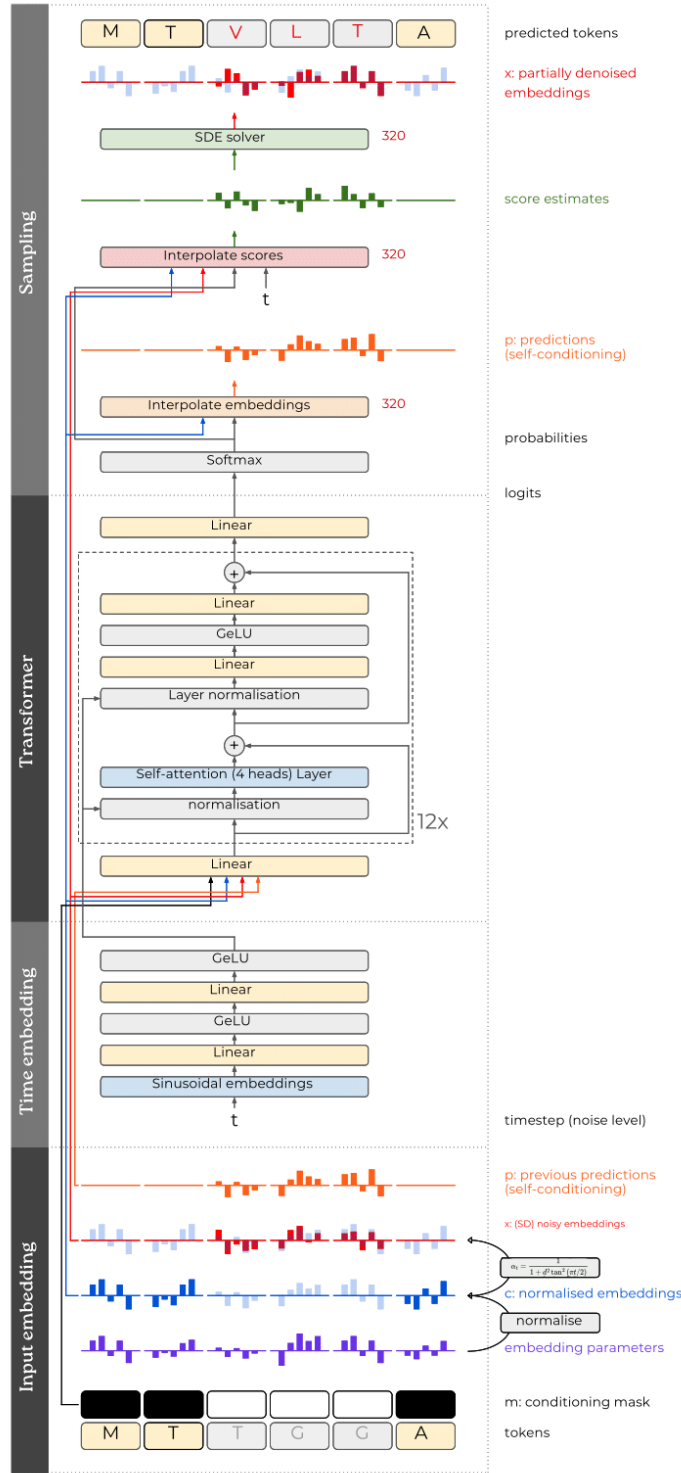


Figure 3: Transformer using the continuous diffusion framework adapted for protein sequences. The conditioning mask indicates which sequence positions are given ('clean') and which are to be generated ('noisy'). During training, SD noise is added to the embeddings for noisy positions. The Transformer input consists of a timestep (t) embedding, along with the concatenation of the mask, token embeddings c for clean positions, and noisy embeddings x and previous predictions p for noisy positions. The output logits are used to interpolate the vocabulary embeddings (to produce new predictions) and the corresponding score functions (to produce score estimates). The score estimates are used to partially denoise the noisy embeddings x .

3. Results

We will discuss the results from the mentioned paper [7], alongside our findings on the AMPs data, followed by a comparative evaluation.

3.1 AFDB & SwissProt Dataset

Model	pLDDT (\uparrow)	ESM-2 ppl (\downarrow)	scPerplexity (\downarrow)	TM-score (\uparrow)	BLAST (\uparrow)	FPD (\downarrow)	MMD (\downarrow)	OT (\downarrow)
SwissProt (dataset)								
Dataset	80.7	5.35	1.88	0.80	100	0.13	0.00	1.08
Random sequences	25.0	21.54	2.77	0.33	0	3.97	0.20	3.88
nanoGPT	61.0	8.18	2.04	0.63	43	1.24	0.03	2.53
EvoDiff-OADM	37.1	15.77	2.44	0.42	12	1.49	0.11	2.63
SeqDesign	43.1	11.89	2.35	0.41	17	3.53	0.19	5.12
proteinGAN	30.4	16.48	2.57	0.33	0	2.94	0.17	3.98
DiMA	80.8	5.20	1.80	0.85	68	0.41	0.01	1.41
w/o skip connections	77.3	5.84	1.87	0.82	61	0.48	0.02	1.51
w/o time layers	79.4	5.49	1.83	0.85	66	0.44	0.02	1.44
w/o ESM encoder	62.7	9.22	2.09	0.71	48	1.05	0.04	2.14
w/o self-conditioning	68.2	9.18	2.08	0.74	46	0.54	0.04	1.61
w linear schedule	77.0	6.29	1.89	0.82	58	0.50	0.02	1.51
w cosine schedule	54.1	10.86	2.16	0.60	34	0.97	0.06	2.02
AFDB (dataset)								
Dataset	83.9	5.79	1.75	0.92	100	0.18	0.00	1.57
Random sequences	26.2	21.67	2.75	0.35	0	3.02	0.18	4.15
nanoGPT	68.5	8.21	1.94	0.77	40	0.62	0.02	1.99
DiMA	73.9	8.50	1.90	0.85	48	0.69	0.03	1.86
w/o self-conditioning	56.3	12.08	2.18	0.69	31	0.96	0.05	2.29

Table 1: Performance comparison between DiMA and baseline architectures of the same parameter count trained on SwissProt and AFDBv4-90 datasets [7].

In the evaluation of sequence generation on the SwissProt dataset (Table [1]), DiMA outperforms all baselines, achieving metric values closely aligned with the dataset. NanoGPT, while strong, falls short of matching the dataset’s metrics. Other baselines, such as SeqDesign and ProteinGAN, perform poorly due to their focus on narrower protein classes. EvoDiff, although better than SeqDesign and ProteinGAN, still shows metrics closer to a random sample, consistent with its original findings.

On the AFDBv4-90 dataset, the performance gap between DiMA and NanoGPT narrows. DiMA achieves higher values in pLDDT, TM-score, and BLAST metrics, while NanoGPT excels in ESM-2 pseudoperplexity. Both models are similar in scPerplexity scores. Despite these achievements, neither model fully matches the dataset’s metrics.

In terms of distributional similarity, DiMA surpasses all baselines on the SwissProt dataset. On AFDBv4-90, both DiMA and NanoGPT show strong results, with NanoGPT slightly ahead in FPD and MMD. However, DiMA demonstrates better clustering performance, with a higher percentage of generated sequences having at least 30% similarity to dataset sequences. This indicates DiMA’s ability to produce sequences closely resembling the training data.



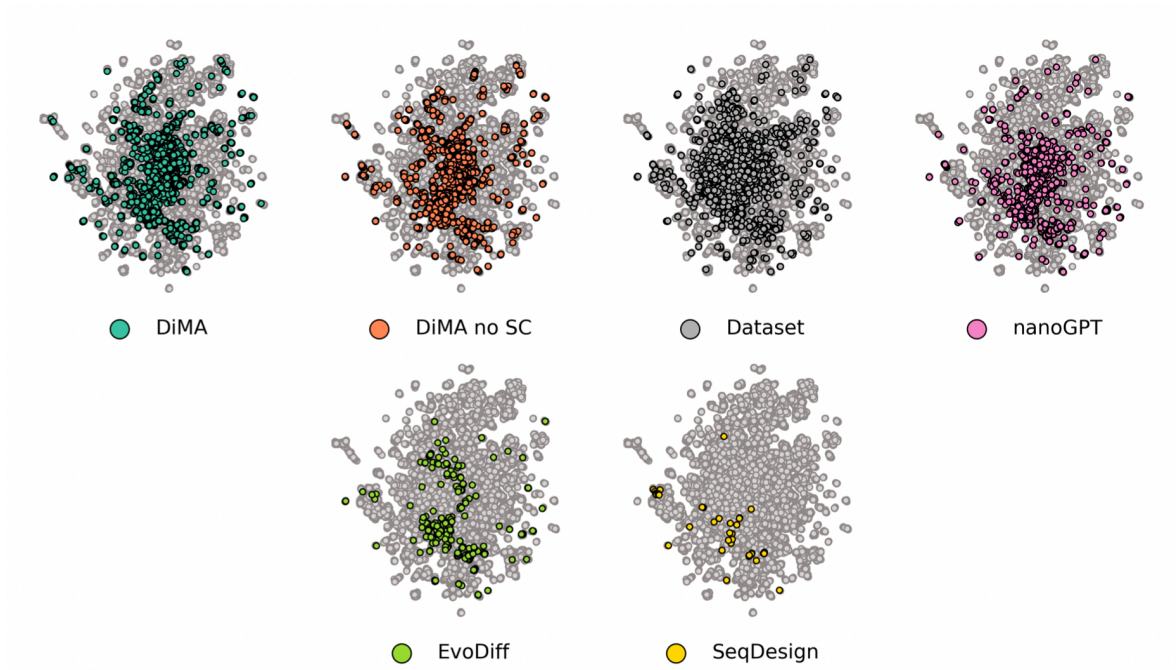


Figure 4: UMAP projection of sequences from PC. Training dataset - SwissProt. Grey background points - dataset sequences from PC [7].

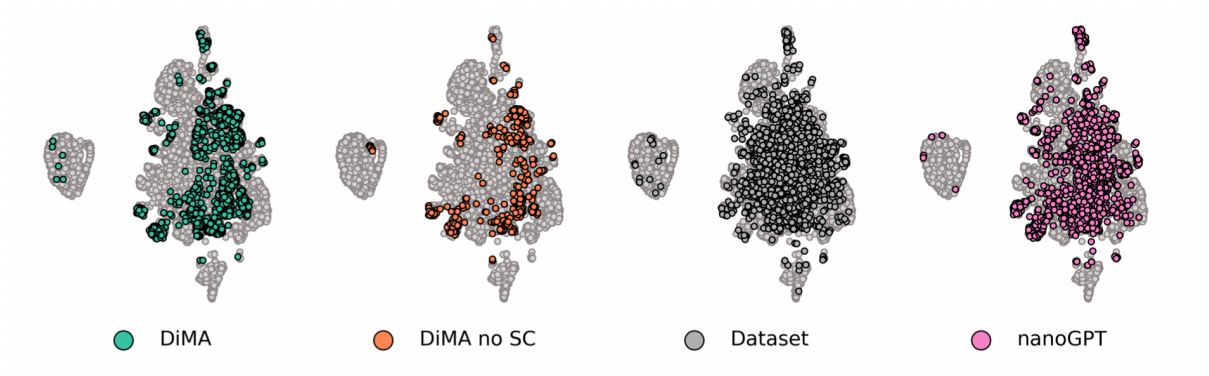


Figure 5: UMAP projection of sequences from PC. Training dataset - AFDBv4-90. Grey background points - dataset sequences from PC [7].

To visually represent the distribution of generated sequences, UMAP is utilized, trained on all sequences from the PC for all models, with parameters set to 25 neighbors and a minimum distance of 0.5. The UMAP plots (Figures [4] and [5]) indicate that, although DiMA without self-conditioning has a higher SeqD metric, DiMA with self-conditioning achieves similar coverage on SwissProt and even greater coverage on AFDBv4-90. This, along with the closer alignment of DiMA to the dataset in distribution learning metrics, suggests that DiMA without self-conditioning achieves better diversity by generating sequences that significantly differ from the dataset.

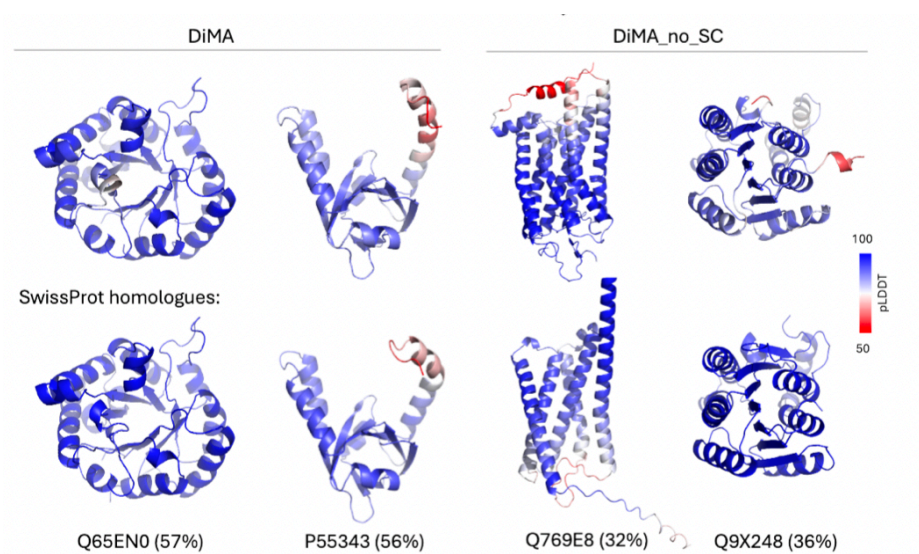


Figure 6: *ESMFold* predicted representative examples of proteins generated by DiMA (top) and the closest hit SwissProt (bottom) with UniProt IDs and the homology % [7].

3.2 Antimicrobial Peptides (AMPs) Dataset

3.2.1 AlphaFold-2 Visualization

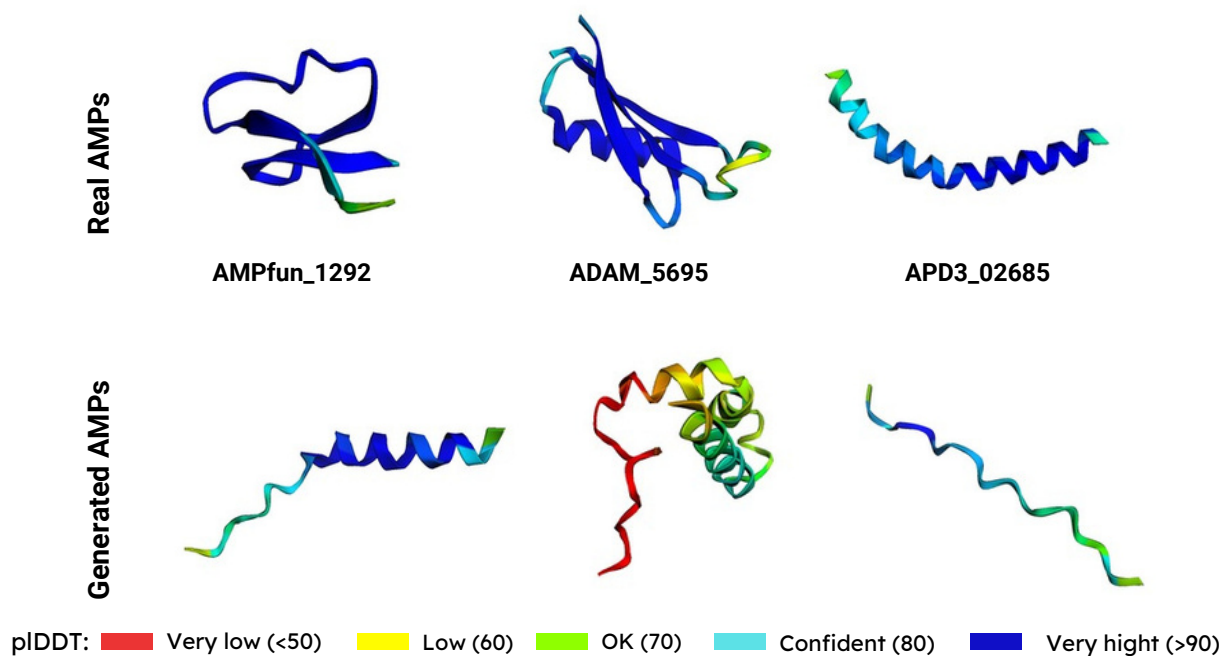


Figure 7: 3D structures generated by AlphaFold-2 for real AMPs (top) compared to those generated by DiMA (bottom).



3.2.2 UMAP Projection

In the following UMAP figures, we denote different runs⁴ as generated i , for $i = 1, 2, 3$.

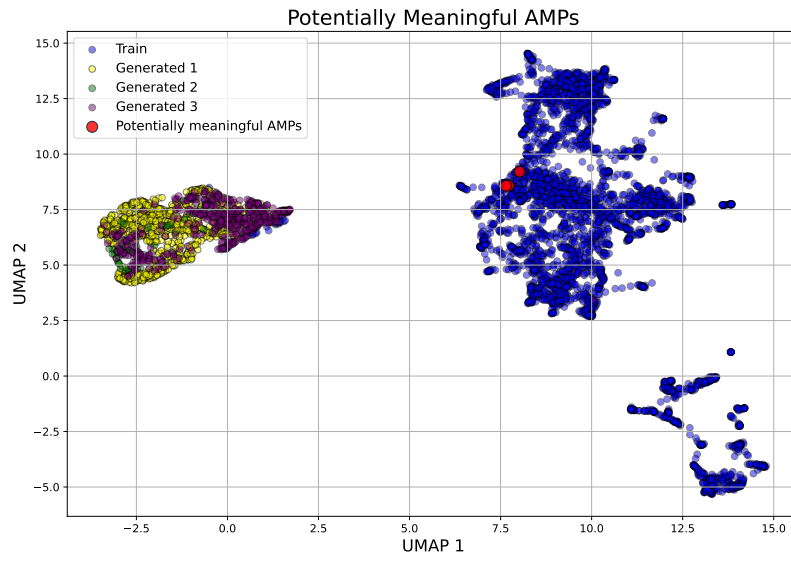


Figure 8: 2D UMAP projection of AMPs.

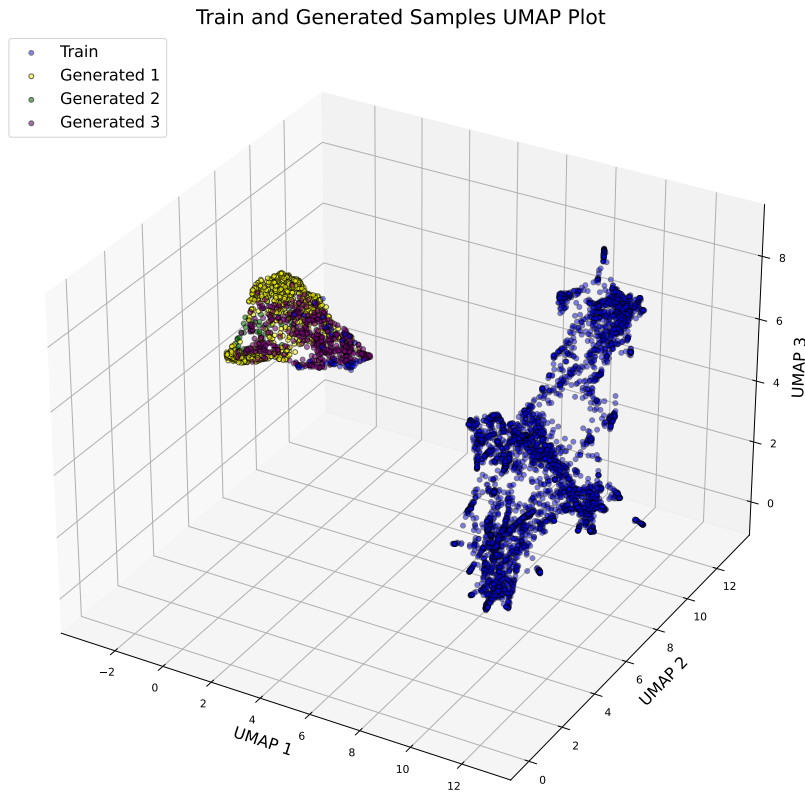


Figure 9: 3D UMAP projection of AMPs.

⁴Referring to different model configurations.

From the UMAP plots above, we can observe that the majority of the generated AMPs form a separate cluster from the training data. Additionally, for each run, the generated sequences appear to form a common cluster.

These results indicate that the distribution of the training data differs from that of the generated data.

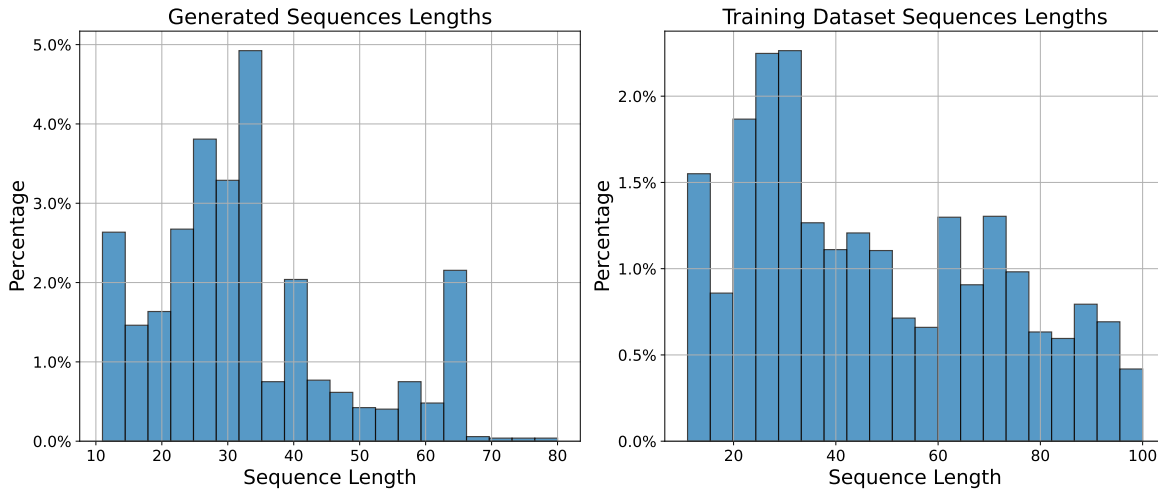


Figure 10: The distribution of lengths in the training (right) and generated (left) sets.

From the above histograms, we can observe that the length distribution of the generated sequences closely matches that of the training sequences, which is an expected outcome. We observe that for sequences with a length of 70 and above, only a small number of samples are generated.

3.2.3 Classification Experiments

We conducted two classification experiments to validate our results, utilizing a Logistic Regression classifier, which has proven reliable for similar tasks.

In the first experiment, we combined the generated embeddings from various runs into a single dataset and assigned them a positive label. A logistic regression classifier was then trained using a dataset containing both positive and negative samples. The model was subsequently used to predict the labels of the generated samples, successfully classifying all of them as AMPs.

In the second experiment, our objective was to distinguish between real and generated samples. We merged the generated embeddings with real samples from both positive and negative classes into a single dataset, assigning labels of 1 for generated and 0 for real. A logistic regression model was trained on this combined dataset, and it accurately classified all generated sequences as artificial.

The results of these experiments demonstrate that while the generated sequences exhibit AMP characteristics, they can be readily distinguished from real AMPs, likely indicating significant differences between the generated sequences and actual AMPs.

3.2.4 pLDDT & pTM Performance


For each protein, five AlphaFold-2 models were fitted, each using three different seeds. For each model-seed combination, pLDDT and pTM values were calculated. The script computes the average pLDDT and pTM for each model-seed combination and the overall averages for each protein. Additionally the total average pLDDT and pTM across all generated samples was calculated. These metrics are essential for evaluating the quality of protein models: pLDDT (predicted Local Distance Difference Test) measures the confidence in the local structure prediction, while pTM (predicted Template Modeling score) assesses the overall similarity of the predicted structure to the target, in-




dicating the accuracy of the folding prediction. The global average **pLDDT** was found to be **60.28**, and the average **pTM** was **0.891**. These scores are on the higher end compared to other methods presented in Table 1. However, these metrics alone are not sufficient to fully assess the quality of the generated samples. Comprehensive evaluation should also include additional structural and functional analyses.

4. Manual

For those interested in the code procedure, you can follow the steps outlined below to generate your own sequences using DiMA:

1. Download code repository from [here](#) .
2. Move to DiMA-2FBF folder.
3. Run `"pip install -r requirements.txt"` in order to install the required libraries.
4. Create a folder named "data" in the working directory and inside a folder with your datasets name.
5. Edit "config.py" file. Define project name, train and test dataset locations.
6. Run `"python -m utils.get_statistic"` to calculate the component average and variance of the embeddings of the text which are necessary to train the diffusion model.
7. Create **wandb** account and login in order to observe training.
8. Then run `"python train_decoder.py"` in order to fine-tune ESM2 decoder for the task as suggested by the authors.
9. Then run `"--nproc_per_node=1 --master_port=31345 train_diffusion.py"` to train the model.
10. Access results in the `generated_seqs` folder.

An overall analytical notebook can be found [here](#) . It contains the overall code used to obtain results for the AMPs data using the **DiMA** model with adjusted architectures to fit the smaller sample size we had to deal with. Furthermore, it includes a variety of functions used for the AlphaFold-2 analysis and overall visualizations. Ultimately, in the last part of the code, one can find the discriminator that classifies whether an AMPs sample is real or generated.

5. Conclusions

The authors demonstrate that DiMA consistently outperforms the baseline models used for comparison. The study highlights that integrating self-conditioning with continuous diffusion and embeddings generated by a protein language model like ESM-2 is a promising approach for generating protein sequences with specific functionalities.

Our application of the aforementioned method on antimicrobial peptides (AMPs) yielded poor results, likely due to insufficient computational resources to handle an adequate amount of training data and model complexity. However, some of the generated sequences appeared meaningful, as mentioned earlier, though further investigation is necessary.



References

- [1] S Alamdari, N Thakkar, R van den Berg, AX Lu, N Fusi, AP Amini, and KK Yang. Protein generation with evolutionary diffusion: Sequence is all you need. *bioRxiv* 2023. *Google Scholar*.
- [2] Emily Engelhart, Ryan Emerson, Leslie Shing, Chelsea Lennartz, Daniel Guion, Mary Kelley, Charles Lin, Randolph Lopez, David Younger, and Matthew Walsh. A dataset comprised of binding interactions for 104,972 antibodies against a sars-cov-2 peptide. *Scientific Data*, 9, 10 2022.
- [3] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin ídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy 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*, 596:583 – 589, 2021.
- [4] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.
- [5] Liuzhenghao Lv, Zongying Lin, Hao Li, Yuyang Liu, Jiaxi Cui, Calvin Yu-Chian Chen, Li Yuan, and Yonghong Tian. Prollama: A protein large language model for multi-task protein language processing. *ArXiv*, abs/2402.16445, 2024.
- [6] Ali Madani, Bryan McCann, Nitish Keskar, Namrata Anand, Raphael Eguchi, Possu Huang, and Richard Socher. Progen: Language modeling for protein generation, 03 2020.
- [7] Viacheslav Meshchaninov, Pavel Strashnov, Andrey Shevtsov, Fedor Nikolaev, Nikita Ivanisenko, Olga Kardymon, and Dmitry Vetrov. Diffusion on language model embeddings for protein sequence generation. *arXiv preprint arXiv:2403.03726*, 2024.
- [8] Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.
- [9] Sergey Ovchinnikov and Po-Ssu Huang. Structure-based protein design with deep learning. *Current Opinion in Chemical Biology*, 65:136–144, 2021. Mechanistic Biology * Machine Learning in Chemical Biology.
- [10] Donatas Repecka, Vyintas Jauniskis, Laurynas Karpus, Elzbieta Rembeza, Irmantas Rokaitis, Jan Zrimec, Simona Poviloniene, Audrius Laurynenas, Sandra Viknander, Wissam Abuajwa, et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3(4):324–333, 2021.
- [11] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):2403, 2021.
- [12] Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024.
- [13] Zachary Wu, Kadina E. Johnston, Frances H. Arnold, and Kevin K. Yang. Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27, 2021. Mechanistic Biology * Machine Learning in Chemical Biology.
- [14] Ke Yan, Hongwu Lv, Yichen Guo, Wei Peng, and Bin Liu. sampred-gat: Prediction of antimicrobial peptide by graph attention network and predicted peptide structure. *Bioinformatics (Oxford, England)*, 39, 11 2022.
- [15] Lin Zongying, Li Hao, Lv Liuzhenghao, Lin Bin, Zhang Junwu, Chen Calvin Yu-Chian, Yuan Li, and Tian Yonghong. Taxdiff: Taxonomic-guided diffusion model for protein sequence generation. *arXiv preprint arXiv:2402.17156*, 2024.

