Latent Spaces for Antimicrobial Peptide Design

Samuel Renaud

A Thesis
in
The Department
of
Physics

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Science (Physics) at
Concordia University
Montréal, Québec, Canada

March 2023

Abstract

Latent Spaces for Antimicrobial Peptide Design

Samuel Renaud

Current antibacterial treatments cannot overcome the growing resistance of bacteria to antibiotic drugs, and novel treatment methods are required. One option is the development of new antimicrobial peptides (AMPs), to which bacterial resistance build-up is comparatively slow. Deep generative models have emerged as a powerful method for generating novel therapeutic candidates from existing datasets; however, there has been less research focused on evaluating the search spaces associated with these generators. In this research I employ five deep learning model architectures for de novo generation of antimicrobial peptide sequences and assess the properties of their associated latent spaces. I train a RNN, RNN with attention, WAE, AAE and Transformer model and compare their abilities to construct desirable latent spaces in 32, 64, and 128 dimensions. I assess reconstruction accuracy, generative capability, and model interpretability and demonstrate that while most models are able to create a partitioning in their latent spaces into regions of low and high AMP sampling probability, they do so in different manners and by appealing to different underlying physicochemical properties. In this way I demonstrate several benchmarks that must be considered for such models and suggest that for optimization of search space properties, an ensemble methodology is most appropriate for design of new AMPs. I design an AMP discovery pipeline and present candidate sequences and properties from three models that achieved high benchmark scores. Overall, by tuning models and their accompanying latent spaces, targeted sampling of anti-microbial peptides with ideal characteristics is achievable.

Acknowledgments

The research in this thesis was performed under the guidance of Prof. Re Mansbach and the past three years have been an extraordinary journey, fraught with challenges that would make an adventure novel proud. Since its inception in 2019, the Mansbach Research Lab has achieved numerous first-time accomplishments, culminating in the publication of our co-authored paper, 'Latent spaces for antimicrobial peptide design', in February 2023. This success would not have been possible without the vigilant guidance of Re, whose leadership was instrumental in our growth. As the lab expanded, I found great strength and gained valuable insights from my new labmates.

I am deeply grateful to my parents for their unwavering support and encouragement throughout my years of study and research. Their constant motivation has empowered me to seek new challenges and achieve my goals.

The thesis features a modified version of the "Latent spaces for antimicrobial peptide design", originally published in the Digital Discovery journal on February 16 2023. https://pubs.rsc.org/en/content/articlelanding/2023/dd/d2dd00091a

This research was supported in part by Discovery Grant #RGPIN-2021-03470 from the National Sciences and Engineering Research Council of Canada. This research was enabled in part by support provided by Calcul Quebec (www.calculquebec.ca) and Compute Canada (www.computecanada.ca). This research was undertaken, in part, thanks to funding from the Canada Research Chairs Program. I would like to personally thank Dr. Yiming Xiao, Dr. Brandon Helfield, Mansbach research lab members for useful discussions.

Contribution of Authors

The following lists the the contributions of the authors for the *Latent Spaces for Antimicrobial Peptide Design* chapter of this thesis. Rachael A. Mansbach: conceptualization, methodogy, resources, writing - review and editing, supervision, project administration, funding acquisition. Samuel Renaud: methodology, software, validation, formal analysis, investigation, data-curation, writing - original draft, editing and visualisation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rise in prominence of deep learning in recent years there is a growing interest utilizing it in the field of biophysics. A biomolecular researcher who might have required years to develop a new candidate drug, iteratively selecting candidate compounds, can now leverage computational resources to greatly accelerate the filtering process and find new candidates for drug development. The astounding pace of research in the field of deep learning has left in its wake many unanswered problems, one of which is the interpretability crisis. While deep learning models have been setting records across disciplines, there is yet to be a concise theoretical development on exactly what is being "learned" by these models. To elucidate the inner workings of these models for antibiotic design we build, train and compare different generative deep learning architectures, and their respective search spaces.

Specifically in this thesis I present a thorough investigation of 5 generative deep learning models tasked with de novo antimicrobial peptide (AMP) generation. One objective of this research is to elaborate on how different generative models construct representational spaces when learning about antimicrobial peptides. It is of interest to map the complex high dimensional representations learned by deep learning models down to lower dimensions that are more readily interpretable and comparable. Once understood a deep learning models strengths and weaknesses can be properly assessed and downstream AMP candidate generation can be performed with greater precision and quantified uncertainty.

In Chapter 2, I briefly introduce the crucial deep learning models and overarching architectures featured throughout this research. In Chapter 3, I will provide context for this research through an overview of previous research on AMPs and de novo AMP design using deep learning. In Chapter 4, I will present the work I have done on leveraging deep generative models for informed de novo AMP design. In this research I comparatively investigate the differences between the constructed latent spaces, their respective interpretability and their ability to be efficiently sampled for de novo design of AMP candidates. Finally I will finish with closing thoughts and remarks for future AMP researchers using deep learning models.

Chapter 2

Deep Learning and Generative Models Overview

This chapter introduces the background on various deep learning models I investigated during my thesis work. The main goal of this chapter is not to elaborate on all the details and inner workings of these models but to present the frameworks on which they operate and the main differences between each model. The chapter first introduces machine learning and deep learning broadly, and then introduces the concept of a neural network with a brief description of their objectives and training methodology. Using the basic neural network as a starting point, I introduce important neural networks such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Finally I introduce the concept of a generative deep learning model and outline popular architectures such as the Variational Autoencoder (VAE), Adversarial Networks and the Transformer. The distinction between the neural networks and the architectures is that architectures are composed of neural networks and feature a variety of the aforementioned fully connected, CNN and RNN networks as blocks within their designs.

2.1 Deep Learning

Machine learning is an optimization procedure that makes use of large quantities of data to minimize objective functions and achieve a goal [1]. A machine learning model is typically used when a researcher wants to find patterns in large quantities of data. To implement a machine learning model it is necessary to establish what the over-arching goal, model-type, input data, objective function and optimizer are. The goal serves to direct the researcher towards a particular framework and objective function and joined with the input data they will be the basis for the model-type. Examples of input data are: text, images, audio and sensor data.

I begin by introducing the notion of a linear regression model from machine learning and show that a fully connected neural network, a simple deep learning model, is a twist on linear regression. Linear regression models have as objective function the minimization between a learned linear curve and a dataset. The relation captured by a linear regression model is given by equation 1,

$$y = \mathbf{W}^T\mathbf{X} + b \tag{1}$$

Where $\mathbf{W}$ is the learned parameter matrix of the model, b is a learned bias constant and $\mathbf{X}$ is an

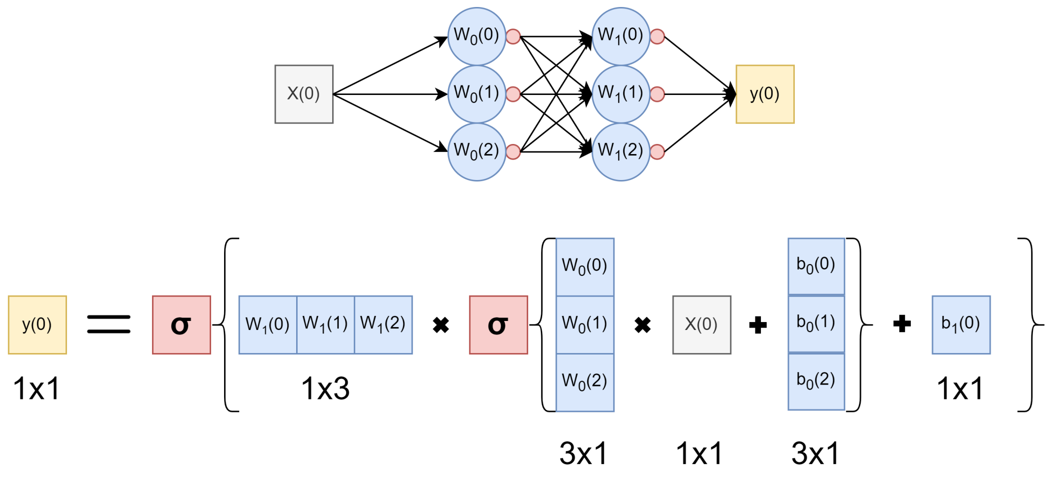

Figure 2.1: A visualisation of a neural network as a directed graph (top), along with its mathematical matrix equivalent (bottom). The bias, b_i, is omitted from the graph representation but is implied through the passing of the W_i layer.

input data matrix. We can discern from this function that a linear regression model can adapt to fit linear trends in data.

While traditional machine learning models are capable of tasks beyond linear fits, they have struggled to solve complex high dimensional problems such as voice or image recognition [4]. Deep learning models have been introduced to address the shortcomings of traditional machine learning and have been very successful. Neural networks are deep learning models that approximate a function $y = f(X; W)$, where X is the input data and W are the learned weights. A fully connected neural network is a network that can be represented as a directed graph with each node from the previous layer being connected to all nodes in the following layer (Fig. 2.1). Each node in the graph represents a multiplication operation between input data and learned parameters. The matrix view of Fig. 2.1 clearly depicts the similarities between the linear regression model and the fully connected model where an input X is multiplied by a weight matrix W and a bias is added at the end. The key distinction between fully connected networks and linear models is the stacking of linear layers, two in the model shown in Fig. 2.1, and the non-linear activation function applied after the multiplication of the inputs X by the parameters, shown as the red dot at the end of the weight circle in Fig. 2.1.

Non-linearities or activation functions are non-linear functions that serve to augment the complexity of the models. Examples of commonly used activation functions are the Rectified Linear Unit (ReLU), $ReLU(x) = Max(0, x)$, the Sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, and the hyperbolic tangent function, $tanh(x)$. The use of a non-linear activation function transforms the linear network into a non-linear model that is now capable of mapping the input X to a non-linear output y. The sacrifice made by switching to a non-linear activation function is that the optimization problem is no longer convex and can no longer be solved in closed form [1]. The advantage of learning non-linear mappings usually outweighs the cost of having to solve the optimization problem iteratively.

The iterative solution to the optimization problem is typically a form of Stochastic Gradient

Descent (SGD). Intuitively SGD is like rolling a ball down a mountainous region with the height of the ball representing the minimization of the objective function. SGD is composed of three main steps,

(1) The *forward pass*, performing the multiplications on the input x by the randomly initialized parameters θ and applying the activation functions.

(2) The *loss function calculation*, calculates the difference between the model output $\hat{y}$ and the true data label y.

(3) The *backpropagation*, calculates the gradient of the loss function with respect to the model parameters and iteratively update all the parameters θ, of the model, by taking a fixed-distance step in the direction opposing the gradient.

The backpropagation step is intuitively understood with the mountainous region example since it is desirable to find the direction of steepest descent at any given step. In the following sections I will present neural network variations that were used in the research for this thesis. The variations typically place an emphasis on learning useful representations of different input-data modalities by modifying the underlying connections in the network graphs.

2.1.1 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) were inspired by the animal vision system [5]. When compared to fully connected neural networks convolutional neural networks have sparse local connections between the inputs and model parameters (Fig. 2.2). CNN's start with randomly initialized weight matrices called kernels. A kernel is a sliding window that passes over the input data and multiplies the data values within the window frame by the fixed kernel values. CNN's optimise the kernel weights to enforce learning useful representations of the data. In Fig. 2.2 data is passed from the input layer at the bottom to the following layer above. The grey coloring shows both the kernel, of size $k = 3$, applied to nodes, x_2, x_3, x_4, and how the following layer node, s_3, only receives information from these three preceding nodes. This is in contrast to the fully connected network underneath which receives input from all preceding nodes. In this manner CNN's extract local features from the data while fully connected networks extract global features.

While CNN's have been shown to perform well on images, they can also be applied to 1-dimensional data such as a sequence of text or characters. It is in this context we apply CNN's in this research, to learn patterns and features from peptide sequences, see the methods section of chapter 4.

2.1.2 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNN) are specialised in sequential data processing [6]. Sequential data has the shape x^t, where x is a vector and t denotes the index in the sequence. An example of sequential data is a sentence, where each word in the sentence can be represented by a unique vector and ordered with an integer, t, determined by its respective position in the sentence. An RNN takes as input a vector h^{t-1} as a hidden state and a sequence vector x^t and outputs an updated

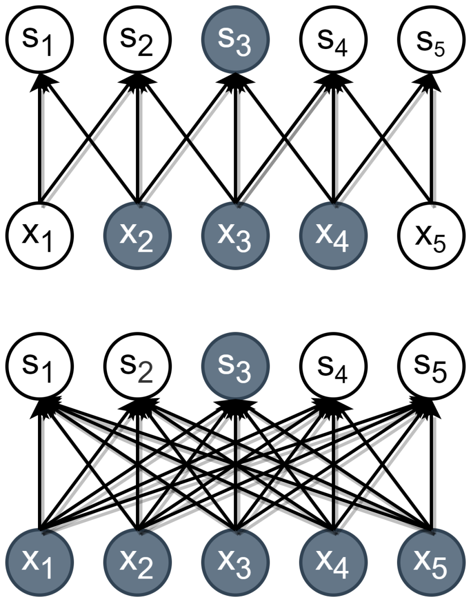

Figure 2.2: A graphical comparison of a CNN (top) with a kernel of size 3, and a fully connected neural network (bottom). The grey coloring indicates the information that is passed from the bottom layer to the center node, s_3 on the top layer, The figure was edited from [1].

hidden state h^t. It is typical to set the initial hidden state, h^0 to be a zero-vector.

$$h^t = f(h^{t-1}, x^t) \tag{2}$$

The recurrent aspect is visualized by unrolling equation 2 over multiple iterations, revealing each update of the network from an initial hidden node, h^0, to the final node, h^t, where t is an integer number of iterations. In contrast to CNN's the parameters of the RNN are shared between each hidden state update (Fig. 2.3). This also means that the backpropagation step will repeatedly update the same weight matrix at each hidden node which leads to exploding and vanishing gradients, a known issue in RNN's where the numerical values of the gradients will become either very large (explode) or very small (vanish). The shared nature of the weights of RNN's lead them to be referred to as the network's memory. Sharing weights over all hidden states results in a lossy memory, stemming from the inability for models to output an identical copy of the input text due to the limited capacity of the model to simultaneously store all the feature information of the previous input vectors. Of the five models built for the AMP research project 4 of them feature a kind of RNN called a Gated-Recurrent-Unit (GRU), introduced by Cho et al. [7] as a successor to RNN's, addressing prevailing RNN issues such as memory loss and vanishing or exploding gradients.

2.2 Generative Deep Learning

Generative deep learning enables the synthesis of new data from a known dataset. Generative models work by using deep neural networks to learn representations of an otherwise intractable probability distribution of the original dataset [1]. Formally, given a dataset $\mathcal{X}^d = \{x_1^d, x_2^d, ..., x_n^d\}$,

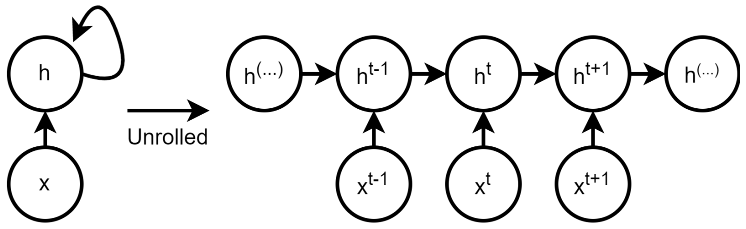

Figure 2.3: A graphical representation of a rolled RNN (left) and its unrolled form (right), The figure was edited from [1].

where d is the dimensions of the samples, assumed to be independently and identically distributed, the generative model must learn about the probability distribution of the original $\mathcal{X}^d$ space and construct an analogous space $\mathcal{Z}^h$, where h denotes the dimensions of the samples in the new representation. Once trained, sampling from a generative model should give new data-points as though they had been sampled from the original $\mathcal{X}^d$ dataset. The analogous space, $\mathcal{Z}^h$, is commonly referred to as the "latent space". It should be noted that the dimensions of the input data d do not need to be equal to the latent representation shape h. It is often desirable for $h < d$ such that both the computational efficiency and interpretability in the $\mathcal{Z}^h$ space are improved. The model that learns the mapping from $\mathcal{X}^d \rightarrow \mathcal{Z}^h$, is commonly referred to as the encoder, and can be used to encode inputs into their latent representations. The model that learns the mapping from $\mathcal{Z}^h \rightarrow \mathcal{X}^d$, is commonly referred to as the decoder and once trained can be used to effectively generate samples from the complex $\mathcal{X}^d$ distribution. The following section introduces widely used deep learning architectures such as VAE's, GAN's and Transformers.

2.2.1 VAE

The variational autoencoder (VAE) is a latent variable model first introduced in Kingma and Welling [8] that combines an "encoder" and a "decoder" to form a generative model. The VAE has two objective functions that must crucially be balanced for a model to operate as intended. The first objective function measures the accuracy of the reconstructed outputs, with respect to the original inputs. This measures the VAE's ability to effectively compress data with the encoder and decompress it with the decoder. The second objective function measures the distribution of the encoded data in the latent space and compares it to a chosen prior distribution. The inclusion of a chosen prior stems from the mathematical derivation of VAE's and is explained in detail below. The second objective function can be thought of as a latent space regularization function that brings the latent probability distribution closer to the desired true data distribution. The VAE model is based on a bayesian statistics derivation of the sought after true data distribution $p(x)$, x $\in \mathcal{X}$. The problem is introduced as follows, we can approximate the true data distribution $p(x)$ by a joint distribution over the data, x, and the latent variables, z.

$$p_\theta(x) = \int p_\theta(x, z)\, dz = \int p_\theta(x|z)p_\theta(z)\, dz \tag{3}$$

Where $p_\theta(x)$ is a model approximation of the true data distribution with parameters θ and $p_\theta(x, z)$ is the joint distribution over both the data x and z the latent variables. While the joint distribution,

$p_\theta(x, z)$, is computable, The integral on the right hand side of equation 3 is intractable.

Instead of dealing with this integral we can approach the problem from a bayesian inference perspective, given a neural network that learns an approximation $q_\phi(z|x)$ of the true posterior $p_\theta(z|x)$, where ϕ are the parameters of the neural network, such that $q_\phi(z|x) \approx p_\theta(z|x)$. The error in the approximation can be evaluated using the Kullback-Leibler divergence, $KL(q||p) = \int q(x) log \frac{q(x)}{p(x)}\, dx$. We will momentarily turn our attention to this error term and find our way back to the true data distribution, $p_\theta(x)$, through it.

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \int q_\phi(z|x)\, log(\frac{q_\phi(z|x)}{p_\theta(z|x)})\, dz \tag{4}$$

making use of the following probability product rule,

$$p_\theta(z|x) = \frac{p_\theta(x, z)}{p_\theta(x)} \tag{5}$$

The KL error term can be re-written as,

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \int q_\phi(z|x)\, log(\frac{q_\phi(z|x)p_\theta(x)}{p_\theta(z, x)})\, dz \tag{6}$$

Using log properties we can split the product in the integral,

$$KL(q_\phi(z|x)||p_\theta(z|x)) = \int q_\phi(z|x)\, log(\frac{q_\phi(z|x)}{p_\theta(z, x)})\, dz + \int q_\phi(z|x)\, log(p_\theta(x))\, dz \tag{7}$$

The first term, on the right of the equal sign, is called the Evidence Lowerbound, (ELBO) and while the equation has gotten more complicated, we have managed to isolate the true data distribution, $p_\theta(x)$ on the right side. If we now solve for the rightmost term we get,

$$\int q_\phi(z|x)\, log(p_\theta(x))\, dz = -\int q_\phi(z|x)\, log(\frac{q_\phi(z|x)}{p_\theta(z, x)})\, dz + KL(q_\phi(z|x)||p_\theta(z|x)) \tag{8}$$

Since $p_\theta(x)$ is independent of $q_\phi(z|x)$ we can remove it from the integral and we can get rid of the negative in front of the ELBO using log properties giving,

$$log(p_\theta(x)) = \int q_\phi(z|x)\, log(\frac{p_\theta(z, x)}{q_\phi(z|x)})\, dz + KL(q_\phi(z|x)||p_\theta(z|x)) \tag{9}$$

Thus using the KL-error term we have managed to obtain an equation for the true data distribution in terms of the latent variable z and the original variables x.

$$log(p_\theta(x)) = ELBO_{\theta,\phi} + KL(q_\phi(z|x)||p_\theta(z|x)) \tag{10}$$

Moving the KL to the left side of the equation it is observed that the ELBO is actually a lower bound on the desired data distribution $p_\theta(x)$ and as KL goes to 0 the ELBO becomes equal to

$log(p_\theta(x))$.

$$ELBO_{\theta,\phi} = log(p_\theta(x)) - KL(q_\phi(z|x)||p_\theta(z|x)) \tag{11}$$

Having identified the ELBO as a tractable optimization objective we now need to elaborate on the optimization details. In order to optimize over the network parameters θ and ϕ found in the ELBO we need to compute the gradients with respect to said parameters $\nabla_\theta, \nabla_\phi$. The gradient with respect to the θ parameters presents no difficulties, by definition we can expand the KL term as,

$$KL(q_\phi(z|x)||p_\theta(z|x)) = q_\phi(z|x)\, log(\frac{p_\theta(z,x)}{q_\phi(z|x)}) \tag{12}$$

$$\nabla_\theta ELBO_{\theta,\phi} = \nabla_\theta \int q_\phi(z|x)\, log(\frac{p_\theta(z,x)}{q_\phi(z|x)})\, dz \tag{13}$$

$$\nabla_\theta ELBO_{\theta,\phi} = \int q_\phi(z|x)\, (\nabla_\theta log(p_\theta(z,x)) - \nabla_\theta log(q_\phi(z|x)))\, dz \tag{14}$$

$$\nabla_\theta ELBO_{\theta,\phi} = \nabla_\theta log(p_\theta(z,x)) \tag{15}$$

However the gradient with respect to the ϕ parameters is more challenging due to the $q_\phi(z|x)$ in the integral. The solution to this problem was coined the reparameterization trick and expresses the expectation found in the ELBO, $\mathbb{E}_{q_\phi(z|x)}[f(q_\phi(z|x))] = \int_z q_\phi(z|x) f(q_\phi(z|x))$ as found in the integral in equation 14, as a transformation of a new differentiable and invertable function of the latent parameter z,

$$z = g(\epsilon, x, \phi)\ ; \quad \epsilon \sim p(\epsilon) \tag{16}$$

where $\epsilon \sim p(\epsilon)$ is a noise sample from a selected random distribution.

$$\mathbb{E}_{q_\phi(z|x)} = \mathbb{E}_{p(\epsilon)} \tag{17}$$

Now the gradient can now be brought into the expectation $\mathbb{E}_{p(\epsilon)}$,

$$\nabla_\phi ELBO_{\theta,\phi} = \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(z)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(z)] \sim \nabla_\phi f(z) \tag{18}$$

Where $z = g(\epsilon, x, \phi)$ is the new function with random noise sampled from $p(\epsilon)$.

Making use of the reparameterization trick and canceling the ELBO with the $p_\theta(x)$ term, as we would like for them to be equal once the model is optimized, the optimization objective is re-written as,

$$ELBO_{\theta,\phi} - log(p_\theta(x)) = \mathcal{L}_{min} = \mathbb{E}_{q_\phi(z|x)}[log(\frac{p_\theta(z,x)}{q_\phi(z|x)})] \tag{19}$$

$$\mathcal{L}_{min} = \mathbb{E}_{q_\phi(z|x)}[log(p_\theta(z,x)) - log(q_\phi(z|x))] \tag{20}$$

$$\mathcal{L}_{min} = \mathbb{E}_{p_{(\epsilon)}}[\log(p_\theta(x,z)) - \log(q_\phi(z|x))] \tag{21}$$

Where z is now a differentiable and invertible transformation of another random variable ϵ,

$$z = g(\epsilon, \phi, x) \tag{22}$$

This new estimate of $\mathcal{L}_{min}$ provides an equation whose partial derivatives with respect to both θ and ϕ can be readily computed because the p_θ and q_ϕ terms are now separate inside the expectation.

It is common to discretize the problem and form a Monte-Carlo estimate of $\mathcal{L}_{min}$ by sampling $\epsilon^{(l)}$, where $l = 1, 2, 3, ..., L$, and the expression for $\mathcal{L}_{min}$ now takes the form,

$$\mathcal{L}_{min} = \frac{1}{L} \sum_{l=1}^{L} [\log p_\theta(x, z) - \log q_\phi(z|x)] \tag{23}$$

While VAE's are well grounded with a foundation in bayesian statistics the introduction of the reparameterized latent space, with a commonly applied gaussian prior, resulted in blurry images on image generation tasks. The blurry image problem has since been addressed with various new and more complex architectures but is still an active area of research [9], [10]. In the context of VAE's for antimicrobial peptide design the gaussian prior enforced on the latent space results in a single peptide being encoded as a distribution over a subset of similar peptides, and has a noticeable effect on reconstruction accuracy.

2.2.2 Adversarial-Models

Adversarial deep learning models such as Generative Adversarial Networks (GAN) [11], and Adversarial Autoencoders (AAE) [12], are deep learning models that, in the simplest case, feature two competing neural networks (Fig. 2.4). Both networks have cost functions that when minimized will improve overall performance while increasing the difficulty for the opposing network. GAN architectures feature so-called "generator" and "discriminator" models. The generator model will receive noise as input and attempt to construct a realistic sample similar to those featured in the training dataset. The discriminator model will receive "real" and "generated" data and attempt to discern which is which. Corresponding loss functions will evaluate the generator's performance and the discriminator's performance individually, and training both models concurrently will result in a competition to produce realistic samples similar to the training data. GAN's can be difficult to train because the discriminator and generator must always be kept performing at approximately the same level. If the discriminator significantly outperforms the generator a "collapse" effect will be seen causing the discriminator to overfit and the generator to essentially give-up. The reverse is also true, if the generator outperforms the discriminator the discriminator will also stop improving. Another limitation of GAN's is that they do not directly encode input data to a latent representation, making the default sampling a randomized process using a random noise seed. This will be remedied with AAE models introduced next.

The AAE in contrast is built on top of the VAE framework, where an encoder model embeds input data to a latent space and a decoder model transforms latent data back into data samples. The AAE adds an additional model named the discriminator (Fig. 2.5). The AAE discriminator functions in a similar manner to the GAN discriminator but instead of receiving two samples in the form of the training data, it receives latent vectors, from encoder-embeded input data, and artificial latent vectors created by sampling numbers from a gaussian distribution. The discriminator learns to discern between the true gaussian-sampled vector and the encoder generated latent vectors. This forces the AAE encoder to generate latent vectors that match the desired prior distribution, which in the trivial case, is the gaussian distribution.

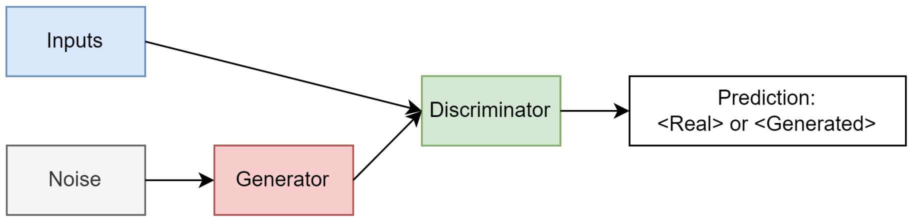

Figure 2.4: The GAN architecture featuring the generator and discriminator neural network blocks. The generator receives a noise sample from which it generates an imposter sample as close to data drawn from the true dataset. The discriminator will attempt to correctly determine which is the imposter data and will receive feedback on this choice for training.

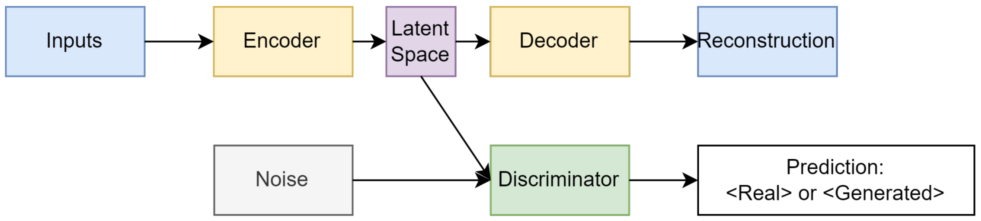

Figure 2.5: The AAE architecture featuring the encoder and decoder neural network blocks as well as the adversarial discriminator network. The encoder receives a data sample as input and passes it through the encoder that transforms the data to its the latent representation. As in the traditional autoencoder a decoder layer will transform the latent vector back into the original data. A separate latent vector copy will be passed to the discriminator model along with a noise sample intended to enforce the prior distribution on the latent representation as discussed in the VAE model. The discriminator must now predict whether the latent vector or the random noise vector is the real noise sample, thus regularizing the learned encoder representations.

2.2.3 Transformers

The Transformer model originally introduced by Vaswani et al. [2] established a paradigm shift in the field of deep learning. Specifically the Transformer model addressed challenges in the domain of natural language processing where, at the time, recurrent neural networks dominated. The typical framework for a language model was to leverage RNN encoder and decoder blocks to process sequences of text. The Transformer model sets aside RNN's entirely and solely leverages the *attention mechanism* for both the encoder and the decoder parts of the model. The model architecture presented in [2] is shown in Fig. 2.6. Since the Transformer was introduced there have been countless variations on the architecture, most of which have also opted to use the label "Transformer", even when the models only feature either the encoder or decoder portions of the original transformer model, and in some cases it is sufficient that the idea be based on the paper by [2] to label the model a variant of the Transformer. This has been the source of confusion for many in the field, trying to make sense of the significance and order of the many architectures.

The attention mechanism emerges from the world of seq-2-seq, or sequence-to-sequence, models where a deep neural network receives a text prompt and must return an appropriate sentence

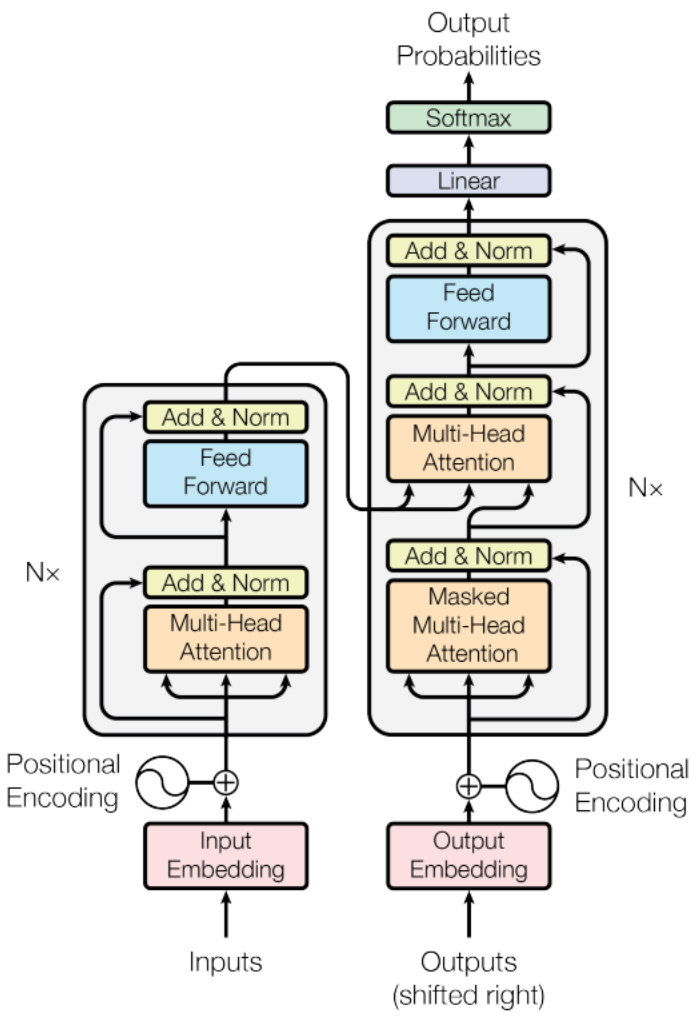

Figure 2.6: The transformer model as presented in [2]. The left side unit is the encoder and the right side unit is the decoder.

as output [13]. The attention mechanism attributes weights between words in a sentence creating weighted connections between pairs of words (Fig. 2.7). Formally the attention mechanism operates through a set of matrix dot products between embeddings of words in a sentence. The embedded words are linked to query-key-value vectors, see equation 24.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{24}$$

Where $Softmax()$ is the softmax activation function, Q is the query vector, K is the key vector and V is the value vector. The factor $\sqrt{d_k}$ in the denominator normalizes the results to prevent the dot products from becoming too large given it is an unbounded operation. The initial dot product between the query, a word embedded as a vector, and a key, another word embedded as a vector, can be conceptualized by imagining vectors, with a magnitude and direction, who's dot

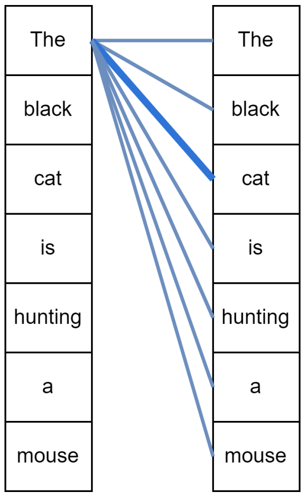

Figure 2.7: A visualisation of the attention mechanism. Here the attention mechanism identifies the important relation between the words ”The” and ”cat” and attributes a greater weight to this word-pair. The attention connections for subsequent words in the sentence are hidden for ease of view.

product will reveal the distance between these word embedding vectors. The $Softmax$ function sets strong query-key pairs to 1 and weak pairs to 0. The value vector serves as a learned weight scaling feature, hence the use of the word “value”, it establishes whether or not the model should care about a given pair connection.

When referring to the Transformer model it is useful to break it into its constituent encoder and decoder parts, and it should be noted that often only one of the two units are necessary for a particular task. For brevity I will omit mention of the fully connected layers featured after each attention layer, these layers serve to aggregate the outputs from the multiple attention heads after all attention blocks.

The encoder model, shown on the left side of Fig. 2.6, takes in embedded sequences of text and attends to all the words in the input sentence. Once the output is passed to a fully connected layer we have completed the encoding phase and now have text embeddings as featured in the BERT encoder model [14] [15].

The decoder model, shown on the right side of Fig. 2.6, is more complex and can be broken down into 3 separate pieces. The self-attention over the output (the bottom right masked attention layer in Fig. 2.6), The attention over the embedded inputs and self-attention-outputs (the top right attention layer in Fig. 2.6), and finally the output predictions (the softmax block at the top right of Fig. 2.6).

The decoder self-attention, (bottom right), receives an embedding of the output sequence. If this is the first pass through the decoder the first token will be the sequence start token and subsequent sequence positions will be padded empty, up to the max sequence length. This first part of the decoder also usually features a mask over the padded future-words to be predicted to prohibit the model from attending to words beyond the word immediately following the previous token. The embedding is passed through the masked self-attention layer and the output is sent to the following decoder part. This first block can be thought of as a staircase, where at the base of the stairs you pick up a <*start*> token, but cannot see what lies on the upcoming steps. With each step you pick up a new token, (the one the model has just predicted), and, while looking at all the pieces you have picked up on the lower steps, use attention to find relationships between these words. With these initial relations established you are now ready to move to the next attention block, where you will receive the unmasked input-sentence embeddings and try to make sense of the all information gathered so far.

The middle attention block of the decoder receives the decoder self-attention as keys and the embedded encoder self-attention, as queries and values, (see Section 2.1.3 for the attention equation). The key, query and value distinction here emphasizes what is being attended to in this block, namely the connection between the input sequence and the currently generated output sequence. It is interesting to note that while the embedded outputs were masked the embedded inputs that are attended over in this part are not. This allows the decoder to attend over the entire input sentence to generate the most appropriate following output token.

Following the input-output attention layer the outputs are sent to a $Softmax$ function that will generate probabilities for words that will fill the next token position.

Examples of well-known decoder only models are GPT-1 and GPT-2 models from OpenAI which formed the foundation for GPT-3 [16] and the Chat-GPT chat-bot model that has seen widespread popularity as of early 2023.

Chapter 3

Review of Generative Deep Learning for AMP Design

Growing antibiotic resistance is a major threat to global health, turning infections that were once easy to treat into life-threatening illnesses [17][18]. Such rapidly increasing resistance to traditional antibiotic regimens has led to the necessity of finding new and unique treatment methods [19]. One possible recourse is the use of antimicrobial peptides (AMPs) [20][21], which have been identified within the innate immune systems of a multitude of species, including humans, and act to modulate the proliferation of infectious diseases [22]. AMPs comprise a candidate class of short proteins, generally no longer than 100 amino acids and often much shorter [23]. Many of them preferentially partition onto prokaryotic (bacterial) over eukaryotic (mammalian) cell membranes and cause bacterial cell death through destabilization of the membrane and expulsion of the intracellular contents[24][25]. Because of the indiscriminate manner with which they attack an integral part of the cell, build-up of resistance to AMPs is relatively slow. Databases of around 10,000 discovered AMP sequences [26][27][28][29], have been made publicly available, but there there remains significant need for further cataloging and disparate analyses.

One tool of growing importance in the arsenal of the biomolecular designer is generative deep learning, a subset of deep learning [30]. Deep learning, in which a computer optimizes the parameterization of a neural network model in a data-rich regime, has become a powerful player in recent years by leveraging the increasing amounts of scientific data and the phenomenal growth in computational power over the last decades. Neural networks are formed by connecting simple functional units known as "perceptrons" through their inputs and outputs to form complex function approximators. Non-linear activation functions between the linear operations increase model representational capacity at the cost of having to solve a non-convex optimisation task [1]. Generative deep learning in specific refers to the use of a model trained from a large body of data to sample new datapoints from an underlying probability distribution intended to mimic as closely as possible that of the training data. One particular variety of generative deep learning model demonstrating competitive performance in the sphere of biomolecular design is the variational auto-encoder (VAE) and its variants [8, 31], in which data points are explicitly embedded in a smooth, continuous "latent space" that can function as a search space for optimization [32].

In the past five years, there has been an explosion of research in the field of deep generative models for small molecule drug design [32][33][34][35][36]. Most work in this area has, until now, has been focused on the design of small molecules, with only a few studies on the design of short peptides [37][38][39], but in the last year or two research in this area has begun to grow [37]. For example, Das et al. [37] constructed a Wasserstein Autoencoder (WAE) model that formed a complete pipeline for de-novo protein design and identified two sequences with potent antimicrobial activity against E.coli. In another paper by the same team[38], they created a tool, (IVELS), for analysing VAE models, throughout the training process, with which they could select models that perform best. Two other relevant recent studies have reported on the use of an Long Short-Term Memory (LSTM) model to design peptides with experimentally-verified activity against multi-drug resistant pathogens, and the use of generative Recurrent Neural Networks (RNNs) to design non-hemolytic AMPs [40][41].

One primary point of interest that is beginning to attract attention is the quality of the latent space itself. Because in data-rich regimes, VAEs and other generative models can be extremely powerful, the focus in past work has been primarily on improving performance [33][42][43] [44][45][46]. However, construction of an interpretable and well-organized latent space not only tends to make targeted feature sample generation easier, it also significantly improves the ability of the user to apply domain expertise to the problem at hand. Therefore, although we do not ignore model performance, a major focus of this article is on interpretability, trustworthiness, and organization of the latent spaces themselves.

In work by Gómez-Bombarelli et al. [32][47][45], the authors demonstrated that training a property predictor on the latent space and forcing the model to consider the quality of its predictions as part of its overall goal leads to an ordering of the latent space, which is desirable for generating candidates with particular features. Because latent spaces tend towards dimensionalities on the order of 30 or above, this orderedness has been visually presented through projection onto a two-dimensional space, often by using principal components analysis (PCA) to identify the most relevant variables. Since PCA is a linear projection, it remains unclear how well this visual orderedness is preserved in the high-dimensional space, and therefore we make a point to address the trustworthiness of this simple, rapid, and useful analytic technique.

Chapter 4

Latent Spaces for Antimicrobial Peptide Design

4.1 Introduction

While previous methods have focused on the longer proteins that represent a larger fraction of the known proteome this research focuses on small antimicrobial peptides (AMPs). Much of the previous work discovering protein embeddings with deep neural networks has used large latent space representations [48][49][50] to maximize data throughput or graph-based representations which require the use of graph neural networks to process the protein graphs. In this work we emphasize small latent representations and model interpretability in order to construct interpretable search spaces for AMP design.

More generally, in this chapter, we focus on the question of latent space quality and interpretability for the relevant test case of generation of novel antimicrobial peptide sequences. We train five different deep generative models with VAE-like latent spaces and assess and compare their different behaviors in terms of reconstruction accuracy, generative capacity, and interpretability. We demonstrate that, as expected, a property predictor leads to an ordering in the latent space. We quantify this ordering and assess how well a PCA projection captures the properties of the original latent space. We argue that the better the PCA projection, the more interpretable the latent space, because we can apply a ranking to linear combinations of the latent space dimensions, allowing us to more easily identify bridge variables that tell us what the model considers important. We also show that the models are capable of generating unique and diverse sequences that grow more AMP-like in specific regions of the latent space.

4.2 Methods

We trained five different models of three different latent-space sizes on a dataset of approximately 300,000 short peptides. In the following sections, we describe and justify the dataset and its properties, the model architectures and the training procedure with relevant hyperparameters.

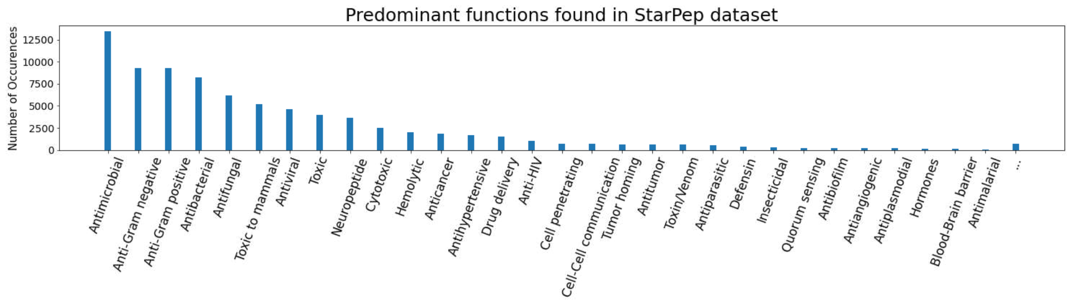

Figure 4.1: Histogram of the StarPep database peptides according to their bio-active functions. Peptides featuring multiple functions appear multiple times. The right most bar combines peptides featuring function labels not shown in the histogram.

4.2.1 AMP Dataset Construction

The datasets used in this study are a subset of peptides from the Uniprot database [51], composed of 268,195 short peptide sequences with a maximum sequence length of 50 residues, combined with a set of 35,806 sequences from the StarPep database, which features short bio-active peptides [52]. We restrict our training set to sequences of lengths 2-50 amino acids because a significant majority of AMPs are short peptides with length < 50, and 85% of the peptides featured in StarPep are between 2 and 50 amino acids long [53]. Preliminary testing demonstrated that retaining peptides up to 100 amino acids in length led to worse performance, presumably due to the heightened sparsification of the dataset in that regime.

Out of the total 45,120 peptides in the StarPep database, 13,437 of them are labelled as having antimicrobial properties (Fig. 4.1). StarPep aggregates bioactive peptides from over 40 existing datasets featuring peptides with sequence lengths between 2 and 100 amino acids. After removing from the dataset the 15% of the peptides featured in StarPep with sequence lengths greater than 50 amino acids and peptides with non-standard amino acids, we retained 35,806 of the original 45,120 sequences, of which 10,841 are labelled as having antimicrobial properties. The remaining 24,965 peptides from Starpep did not have the label Antimicrobial.

Although certain deep learning algorithms can perform well when trained on “smaller” datasets of tens of thousands of datapoints [40][41], these are more typically classifiers rather than generators. When we trained our models on the 35,806 datapoints from the StarPep database alone, we found that the models demonstrated poor reconstruction accuracy, low robustness, and high error. One method for solving this is to pretrain on a large corpus and fine-tune on a smaller one; however, we chose instead to train all at once on a larger corpus, as has previously been done for natural language models and chemical language models [32][33][34][35][36].

Since preliminary analysis on the StarPep database sequences indicated that 35,806 was not a large enough dataset to properly train our generative VAE-based models, we expanded the dataset by adding negative examples from Uniprot. After executing the query [(length:[2 TO 50]) AND (keyword:(KW-0929))], we found 714 of unreviewed peptide sequences with the label “antimicrobial” and, placing “NOT” before the keyword, found 3,713,736 of unreviewed peptide sequences without the label “antimicrobial.” We also found 1,108 of reviewed peptide sequences with the label “antimicrobial” and 11,941 reviewed peptides without the label “antimicrobial”. Since we would not have been able to significantly expand our dataset by using solely reviewed sequences,

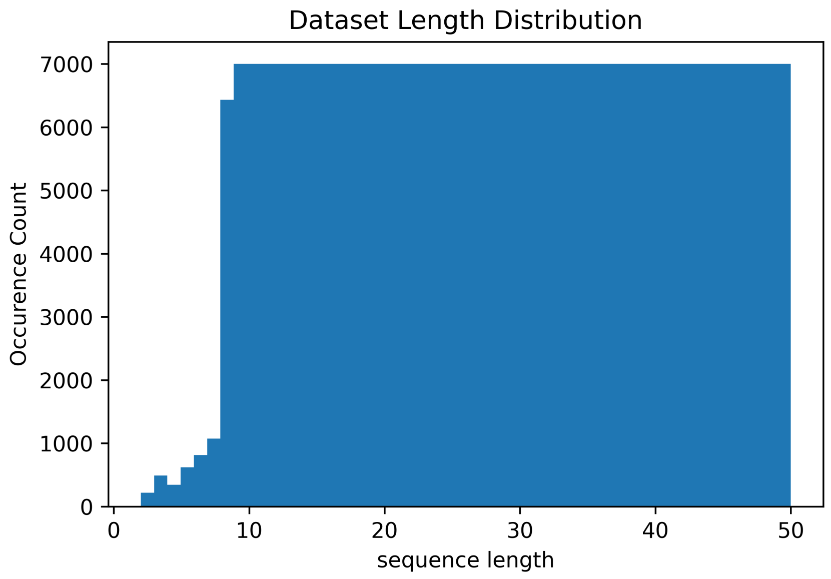

Figure 4.2: Length distribution of the entire peptide dataset used in this study.

we choose to include some non-reviewed sequences. We recognize that in the worse case it is possible that this may include a potentially substantial number of AMPs that have not been identified as such. We considered the possibility of utilizing an existing sequence predictor for AMPs to identify these possibly unlabeled AMPs; however, we decided against it to avoid introducing unknown assumptions into the data at this stage. Instead we consider a different goal for ordering the space. We employ a property predictor that classifies as hits peptides with experimentally-verified antimicrobial properties (which we label with integer 1) and as misses peptides without experimentally-verified antimicrobial properties (which we label with integer 0) and demonstrate that training such a predictor enforces an ordering of the space that allows generation of AMP-like sequences.

To avoid having a prohibitively small percentage of experimentally-verified AMPs in the dataset and to match the sizes of datasets used in previous experiments with similar architectures, we subsampled 10% of the 3 million unreviewed datapoints of Uniprot and selected 268,195 to form an additional set of datapoints. Subsampling was done at random, subject to the constraint of retaining a roughly equal number of peptide sequences of each length to ameliorate the extent to which the models focused on this aspect, though there are fewer very short sequences due combinatoric constraints (Fig. 4.2).

Together the 268,195 peptides from Uniprot and 35,806 from StarPep form our full dataset of 304,001 peptides, of which 10,841 were labeled ”verified AMPs” (all from StarPep) and 293,160 were labeled “non-verified-AMPs” (a mixture of Uniprot and Starpep sequences). We note here again that we only labeled those sequences as AMPs that are experimentally verified as such. Although this could introduce a bias into the property predictor from the unreviewed sequences, we believe that this choice is justified because our goal is not to classify sequences but to order the space in a sensible manner. After ensuring there was no sequence redundancy between the merged datasets, we inspected various physicochemical properties of the “verified AMP” and "non-verified-AMP" labeled data sets to ensure they represented a broad distribution over peptide sequence space and had largely similar distributions of various physico-chemical properties (Fig. 4.3A-J).

4.2.2 Deep Learning Models

Deep generative models are powerful research tools for de novo drug design. Throughout the training process these models can construct smooth latent space embeddings. Once fully trained the learned latent spaces can be explored, and information-rich clusters can be identified. Such continuous latent spaces allow users to sample and decode novel molecules with desirable features.

We investigated five different deep learning architectures applied to the task of generating new peptide sequences from a learned distribution, all of which incorporate a variational autoencoder (VAE) element, along with a VAE-like latent space.

The variational autoencoder is a model introduced in Kingma and Welling [8] that combines an "encoder" and a "decoder" to form a variational inference model. In specific, given a dataset with a marginal likelihood $p_\theta(x)$, the objective of the VAE is to learn the parameter set $\{\theta\}$ that most closely reproduces the data's distribution $p(x)$ [31][54][55]. VAEs operate under an evidence lower bound (ELBO) maximization objective that leads to a joint maximisation of the marginal likelihood $p_\theta(x)$ and a minimization of the Kullback-Leibler (KL) divergence of the decoder-approximated posterior $q_\phi(z|x)$ from the true posterior $p_\theta(z|x)$, where z represents the embedding variables of the latent space and $p_\theta(z)$ is the latent space prior, which for the standard VAE is a Gaussian distribution.

There are multiple types of generative models; the two most well-known and often-used are VAEs and Generative Adversarial Networks (GANs). We choose to employ VAE-based models because VAEs boast a number of properties of particular relevance for our specific case of developing smooth, interpretable latent spaces with potential use as search spaces for computer-aided biomolecular design. In comparison to GANs, VAEs have a more natural formulation of an associated latent space, with a rigorous mathematical derivation from Bayesian probability theory [31]. The assumption of a Gaussian prior enforces a certain level of smoothness and continuity that is desirable in a design space, and allows for the sampling of Gaussian distributions "near" any defined point in the space. Finally, on a practical note, they have demonstrated utility in the field of *de novo* small molecule design [32], and we wanted to determine their utility for sequence-based AMP design as well. VAE's also feature an encoder which can directly map new samples of interest to their respective latent vectors, a feature not present, out of the box, in GAN architectures.

The architectures in this research all feature a latent space comparable to that of a VAE, which in theory should constitute the minimal explanation of the data. It is also desirable for the latent space to be ordered in such a way that it is understandable or interpretable for a subject-matter expert. This latent space is used to interpret the model's latent mapping of inputs from the prior distribution. The latent space also allows for visualisation of embeddings and feature clustering of proteins.

In what follows, we introduce the VAE models and their unique differences. Briefly the RNN, RNN-Attention and Trans-VAE models all employ the typical VAE bottleneck with mean (μ) and log of the variance (σ) network layers while the AAE and WAE employ unique loss functions which regularize the latent space to match a set distribution.

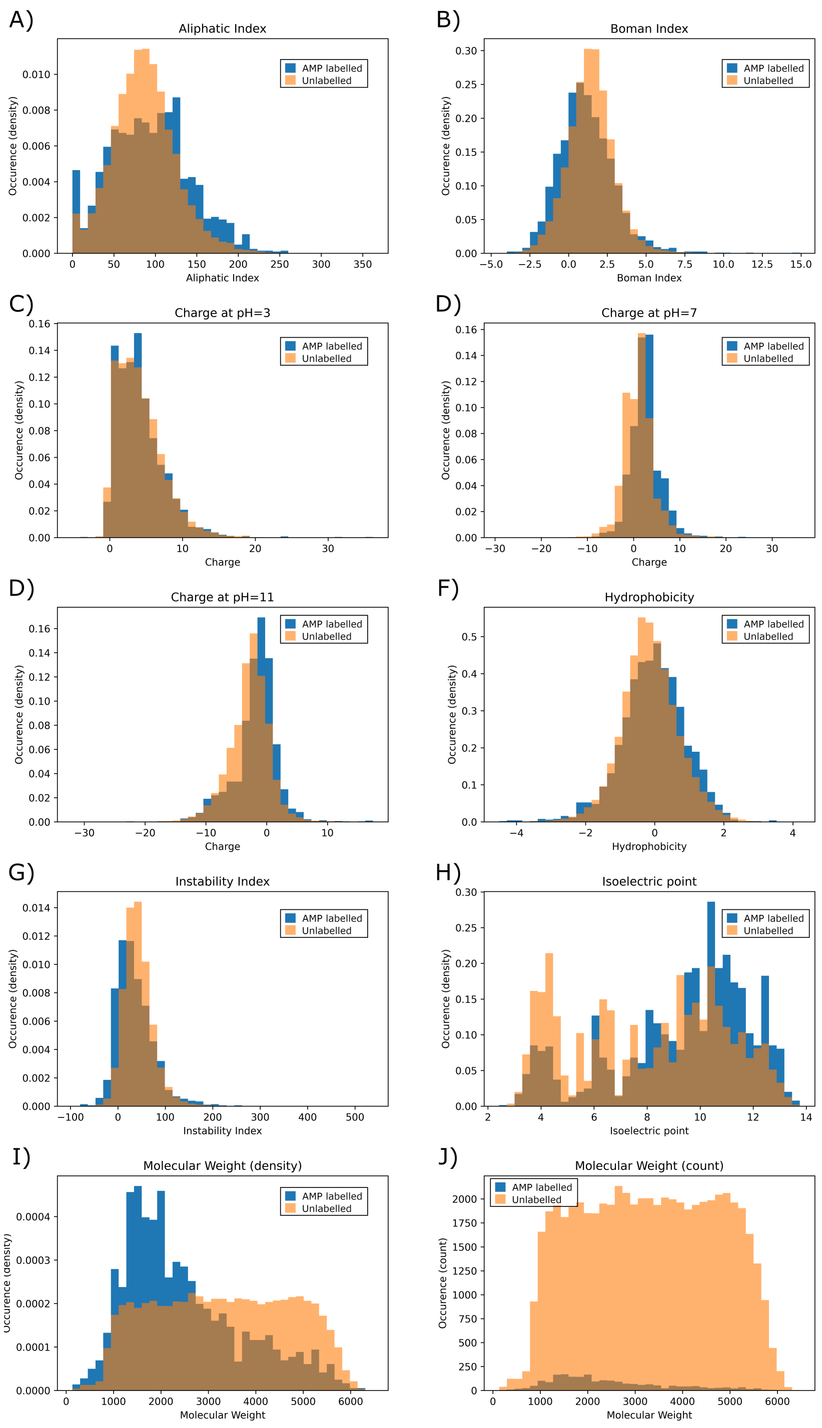

Figure 4.3: Density based distributions for the AMP labelled peptides (blue) and the unlabelled peptides (orange), for all investigated physicochemical properties, namely the aliphatic index (A), Boman index (B), charge at pH=3 (C), charge at pH=7 (D), charge at pH=11 (E), hydrophobicity (F), instability index (G), isoelectric point (H), molecular weight density plot (I) and molecular weight by count (J).

We started with three publicly available models modified from work by Dollar et al. [42], in which the authors compared a recurrent neural network (RNN), a RNN with an attention layer and a Transformer to assess their respective capabilities for generating SMILES strings describing novel molecular compounds for drug design. The models all made use of a VAE architecture that generated smooth latent spaces. The results demonstrated the benefits of including self-attention to deep generative models, and the authors concluded there is a trade-off between variational models in terms of their ability to properly reconstruct input data and their ability to embed complex information about the inputs into a continuous latent space. We modified these three models to take as input and output sequences of up to fifty amino acids rather than SMILES strings. In addition, we derived two more models, an adversarial autoencoder (AAE) [12], and a Wasserstein autoencoder (WAE) [56], by modifying the latent spaces and architectures of the original three.

The RNN model (Fig. 4.4A) is a traditional VAE featuring a Gated Recurrent Unit (GRU)-based Recurrent Neural Network (RNN) as the encoder and as the decoder [7]. The RNN model first embeds the input sequences into fixed length vectors of length 128. The embedded vectors are then passed to the GRU, with N=3 layers, that generates a hidden layer vector of length 128. The hidden layer is layer-normalized and sent to two separate linear layers, the mean layer (μ) and the log-variance layer(σ). The linear layers act as the “bottleneck” and changes–where necessary–the GRU output shape to the appropriate latent space vector shape–either 32, 64 or 128 dimensions, depending on which architecture is being trained. The mean and logvar output are then combined according to the reparametrization trick and Gaussian noise is added [8]. A ReLU unit followed by a layernorm is then applied to the noisy memories. The resulting vector is sent to the decoder GRU. The decoder outputs a 128 dimensional hidden vector that passes through a layernorm. Finally, this hidden vector is sent through a linear layer that generates token predictions. All models except the Transformer implement a GRU with teacher forcing during training. Teacher forcing passes the true sequence to the decoder along with the latent vector and improves training stability by removing the requirement of learning entire sequences and instead prohibits the accumulation of errors on individual tokens.

The Adversarial Autoencoder (AAE) model (Fig. 4.4B) uses the same GRU model as the RNN but exchanges the re-parameterized latent space proposed in Kingma and Welling [8] for an adversarial counterpart Makhzani et al. [12]. As proposed in [12] this model regularizes the latent space with an attached neural network, named the discriminator, that matches the aggregated posterior $q(z)$ to the desired prior $p(z)$, which in this model is a Gaussian prior. The discriminator is a fully connected network with 256 hidden layers, each with a fixed hidden vector of size 640. The encoder network acts as the embedder for the decoder and the “generator” for the discriminator. The decoder learns to reconstruct original data, as in the RNN model case, and the discriminator discriminates between an ideal prior, modeled by a vector filled with normally distributed noise as a proxy for a Gaussian prior, and the current latent vector. This model is hypothesised to perform well on reconstruction tasks given the relative success of adversarial models on the task of image generation [57] [58].

The WAE model proposed in Tolstikhin et al. [56] introduced a new family of regularized autoencoders under the name Wasserstein Autoencoders that minimize the optimal transport $W_c(p(x), p'(x))$, where $p(x)$ is the true data distribution and $p'(x)$ is the generative model data distribution, for a cost function c. The WAE's regularizer, computed as part of the overall loss function, forces the continuous mixture $q_\phi(z) = \int q_\phi(z|x)dp(x)$ to match the prior p(z) instead of forcing each input

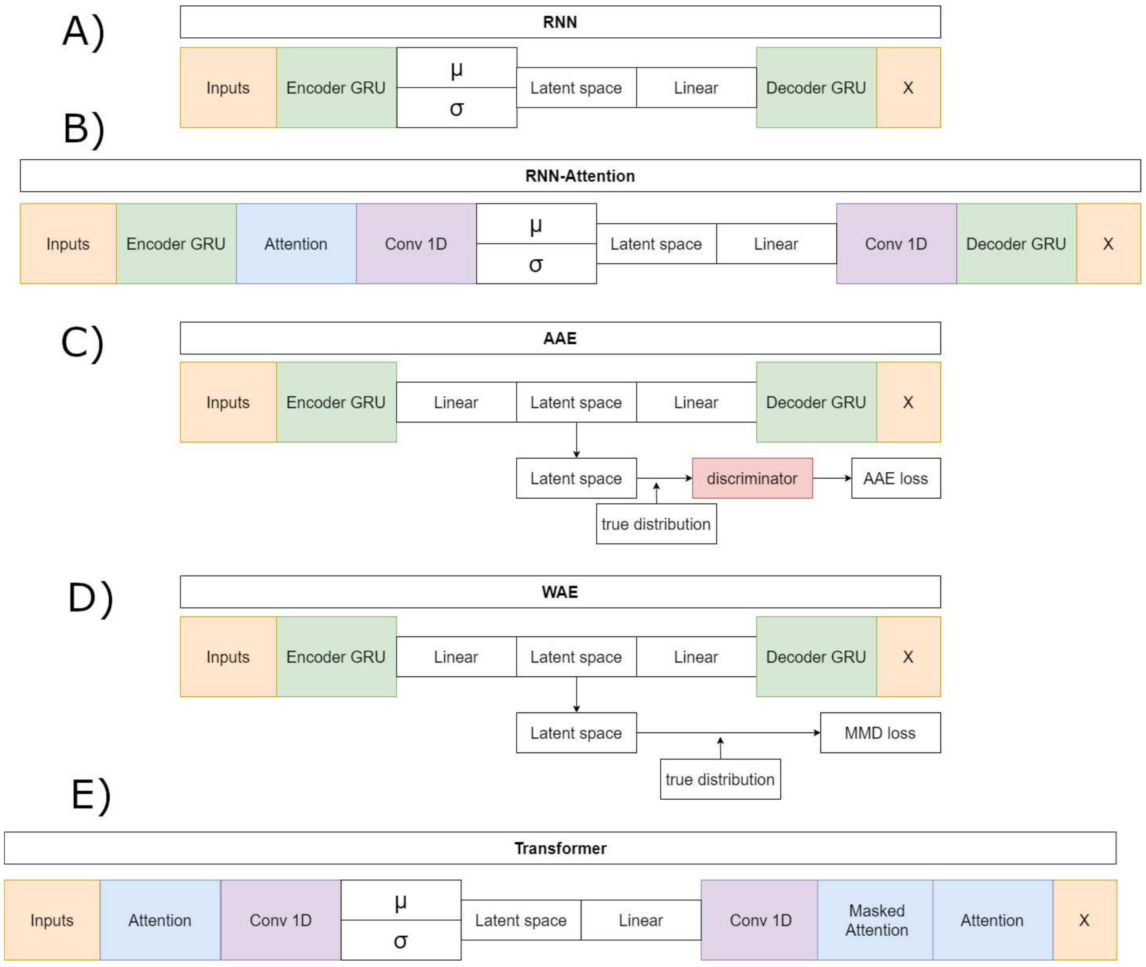

Figure 4.4: Schematic of the five deep learning model architectures, highlighting their similarities and differences. The RNN features a GRU encoder, a variational latent space and a GRU decoder. The RNN-Attention inserts an attention layer after the encoder GRU and uses convolutional layers as input and output from the variational latent space. The AAE features a non-variational latent space but inserts a discriminator network at the latent space. The WAE also features a non-variational latent space but calculates the maximum mean discrepancy between gaussian noise and the encoder generated latent space. The Transformer model implements the generic architecture with a first self-attention layer then a convolutional layer into the latent space followed by deconvolution into the masked attention and the final self-attention layer.

to match the prior as done in the VAE, where $q_\phi(z|x = x_i)$ for all input i in x. The WAE constructed in this work, (Fig. 4.4C), uses the MMD penalty for prior regularization [59]. The WAE implemented uses the same model as the RNN but exchanges the split latent space with a single linear layer. The regularization is performed in the loss function, where the inverse multi-quadratic kernel basis function is used and the MMD is computed between this latent vector and a vector with values sampled from a normal distribution.

The RNN-Attention (RNN-Attention) model, (Fig. 4.4D), attaches a single self-attention head after the GRU of the RNN model to leverage the attention mechanism [13]. The attention mechanism allows a model to store sequence wide connection weights between tokens in a sequence. Each token in the sequence is made to "attend" and give a weight to all other tokens in the sequence thus creating an attention map over all the tokens in the sequence. The single attention layer has been shown to increase interpretability and improve reconstructions over longer sequences [42].

The VAE-Transformer (Transformer) model, (Fig. 4.4E), integrates a VAE reparameterization into a full Transformer model as proposed in Vaswani et al. [2] and implemented in Dollar et al. [42]. Transformers have taken the world of sequence2sequence learning by storm and have demonstrated their abilities to create extended relationships between sequence tokens that can be readily interpreted through attention head visualisations [60][16][61]. The Transformer no longer makes use of the GRU and instead first sends the inputs to a self attention head layer that outputs attention weights that are matrix multiplied onto the output vector. The output is then sent to a convolutional bottleneck that filters over the joined attention weights and embedded sequence before sending the output to the mean and log-variance linear bottleneck layers which reduce the dimensionality of the latent vector to one of 32, 64 or 128 dimensions. Once the VAE section is passed and noise has been added to the latent vector the result is sent back to a convolutional layer followed by the masked attention and source attention layers as originally outlined in Vaswani et al. [2].

All models feature an associated property predictor in the form of a binary classifier that predicts whether the latent representation of a given peptide is an verified-AMP or not. This classifier is a two layer fully connected neural network trained with a binary cross-entropy loss that is joined to the KL-divergence loss and the peptide reconstruction accuracy loss of each model.

4.2.3 Model Training Procedure and Hyperparameter Configuration

The different subsets were collated into a total dataset with 304,001 peptides with lengths between 2 and 50 amino acids. This total dataset was split into a 80-20, train-test split, resulting in a training set of 243,201 training sequences and a test set of 60,800 sequences. The model hyperparameters were set to be identical for all 15 models and their three respective latent dimensions. We fixed the parameters to maintain consistency between the different models and allow authentic comparison of the fully trained networks. The RNN, RNN-Attn and Transformer feature a Kullback-Leibler divergence annealing term β (see Table. 4.1). β starts small and increases in weight as training progresses to avoid posterior collapse. Batch Sizes of 200 were found to be ideal in that they fit on the NVIDIA Tesla P100 graphics cards used for training and results in good performance. 300 epochs was found to be an appropriate number by *a posteriori* inspection of the loss curves.

Model training hyperparameters are outlined in Table. 4.1.

Table 4.1: Hyperparameter table with batch size, initial and final annealing values, learning rate and number of training epochs.

Hyperparameter	Value
Batch Size	200
β final	5e-2
β initial	1e-8
Adam learning rate	3e-4
Epochs	300

4.3 Results

We assess and compare the models and their associated latent spaces in a number of ways. We begin by performing a traditional assessment through model reconstruction accuracies on the training and testing datasets. To assess the generalizability of the models, we benchmark their generative capacities in terms of the distributions they are capable of sampling, a process which is becoming more and more imperative as generative models become more commonplace [33][42]. To assess the meaningfulness of associated distance metrics, we perform a quantitative and qualitative assessment of the models' ability to cluster the data in a low-dimensional space produced by principal components analysis (PCA) and a detailed analysis of the top five principal components. To assess model interpretability and address a longstanding problem in the field, we employ metrics from manifold theory to assess the extent to which PCA distorts the latent space embedding. Leveraging PCA's enhanced interpretability we connect latent representations to peptide properties through bridge variables. Finally, we demonstrate the use of our models in a pipeline. Unless otherwise noted, all analyses are performed on the test set data.

4.3.1 Verification of Model Reconstruction Accuracy

We verify the basic performance of the five trained models through two metrics that demonstrate the ability of the models to recapitulate the input data as they were trained to do. We consider the position-based reconstruction accuracy and the overall sequence reconstruction accuracy.

In Fig. 4.5, we plot the average position-based reconstruction accuracy $\langle R_i^{\text{acc}} \rangle$ versus amino acid index for the five models and for three different latent space sizes, and, in Fig. 4.6, we plot the entire-sequence accuracy. The position-based reconstruction accuracy measures the model's ability to predict the correct amino acid at a given index in the sequence and is defined as,

$$\langle R_i^{\text{acc}} \rangle = \frac{\sum_{n=0}^{N-1} \delta(S_{n_i} = k)}{N} \tag{25}$$

where the sum evaluates whether the correct amino acid, k, was predicted for the i^{th} index of the n^{th} sequence, S_{n_i}, and $N = 50$ is the maximum length of the sequence string in the model.

The mean overall sequence accuracy $\langle R^{\text{acc}} \rangle$ is a measure of how many full sequences the model was able to correctly reconstruct in their entirety,

$$\langle R^{\text{acc}} \rangle = \frac{\sum_{n=1}^{N} (S_n = S_{true})}{N}, \tag{26}$$

where S_n is the n^{th} reconstructed sequence and S_{true} is the original sequence.

Models of 64 dimensions or greater generally display moderate to high performance on the reconstruction task. They exhibit an accuracy of at least 60% for tokens up to position twenty (Fig. 4.5), of interest because most AMPs have sequence lengths of under twenty amino acids. The entire-sequence reconstruction accuracy (Fig. 4.6), is between 50% and 70% for the 128-dimensional AAE, RNN, RNN-Attention, and WAE, and about 97% for the Transformer, although reconstruction accuracy diminishes with decreasing latent space dimensionality.

It is of interest to note that in all models except the 128-dimensional Transformer, we observe an almost monotonic decrease in reconstruction accuracy for tokens later in the sequence. All 32 dimensional models, except the Transformer approach the accuracy of a random guess (5% for any one of twenty possible amino acids) near the 50th position. This clear length-dependent effect on accuracy is expected behavior arising from the increasing difficulty of the predictive task for each successive token position: the prediction depends primarily on the previous tokens in the sequence, leading to a compounding error effect. The difficulty of this task is also exacerbated by model training with teacher forcing, which corrects mistakes earlier in the sequence during training but not during testing. The 128-dimensional Transformer clearly has the capacity to represent and retain entire sequences in a holistic manner, as it was intended to do.

Indeed, the Transformer model achieves by far the highest accuracy (97%, 128 dimensions) but is also most affected by a diminishing latent dimensionality: the mean accuracy drops precipitously from 97% to 26% when going from 128 dimensions to 64 dimensions, whereas the AAE and the RNN both achieve nearly 50% reconstruction accuracy with 64 dimensions. All models display increasing accuracy with increasing dimensionality. All attention based models display a greater improvement when increasing from a dimensionality of 64 to a dimensionality of 128 than when increasing from a dimensionality of 32 to a dimensionality of 64.

4.3.2 Analysis of Top Principal Components Quantifies Differences in the Latent Space

It has been shown that the simultaneous training of a property predictor on a latent space leads to a desirable "ordering" of that space visually identified in the top principal components. Although there are many such dimensionality reduction techniques available, we employ PCA here because it provides an interpretable and invertible mapping to a visualization-friendly space and has been previously used in the field of *de novo* molecular generation [32]. In this section, we identify ordering in the top principal components (PCs) in our models and go one step further by quantifying the extent of the ordering and the PCs in which it appears.

We employ a simple binary classifier trained to predict whether or not a given sequence falls within the experimentally validated AMPs category, or the unvalidated category. Although such a predictor can be of use in terms of assessing the specific properties of some given sequence of interest, in this case we use it rather as a tool to order the space such that we may expect interpolation between points in the latent space to generate sequences with desirable properties. In other words, for our case the property predictor primarily functions to ensure that the neighboring points of an experimentally-verified AMP in the latent space will display similar properties. We can consider this to be a way of enforcing a meaningful distance metric and quantifying how well we have done so. In Sec. 4.3.5, we will also address the question of how meaningful it is for us to

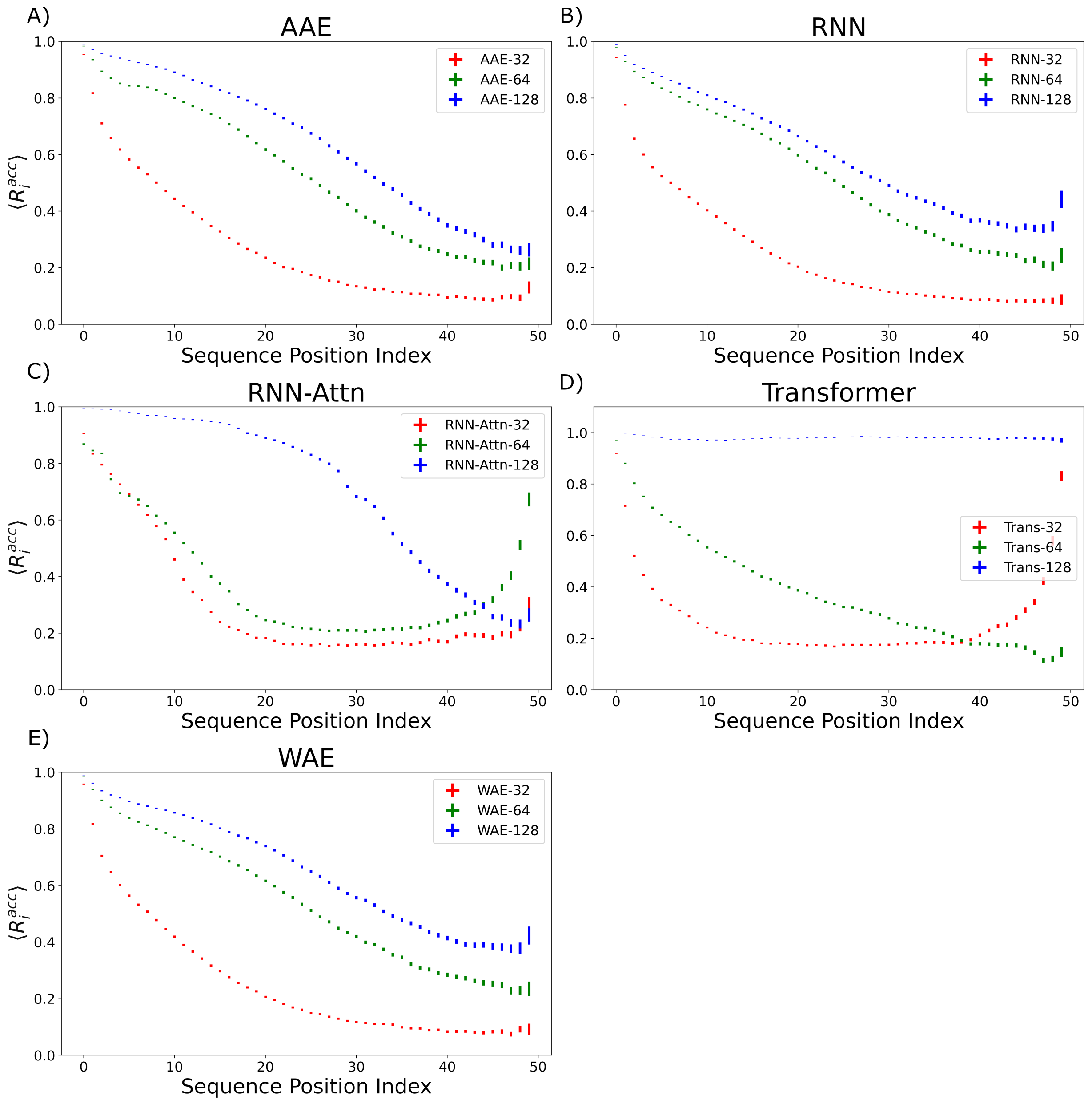

Figure 4.5: The position-based average reconstruction accuracies, R_i^{acc} with corresponding confidence intervals for the five models. The x-axis refers to the token position along the sequence, running from the first generated token to the last generated token. Each model features three variants with latent space sizes of 32, 64 or 128 dimensions, respectively. Error bars are 95% confidence intervals on the respective index-wise means computed with the assumption of Bernoulli sampling. As expected, model performance decreases with distance along the sequence in most cases.

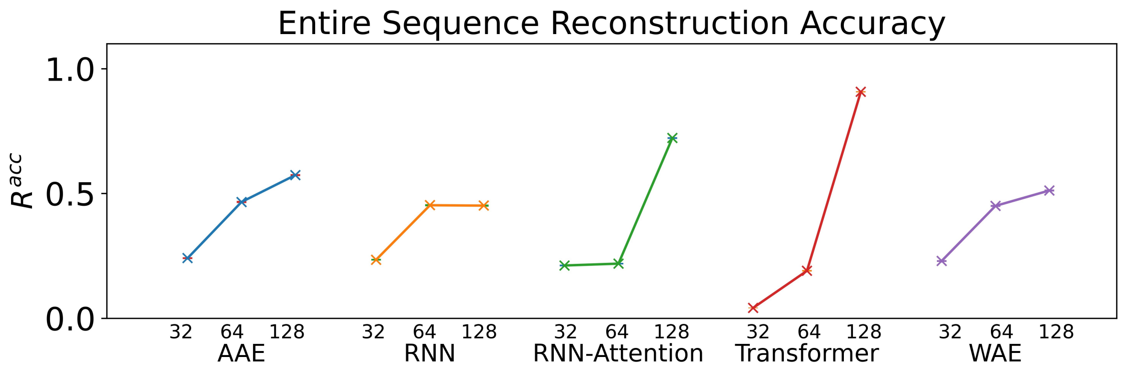

Figure 4.6: The entire-sequence reconstruction accuracies, R^{acc} for the five models and their respective three latent space size variants. Error bars are 95% confidence intervals on the mean computed with the assumption of Bernoulli sampling. All error bars are less than 0.5% from sampling 20,000 sequences.

identify this "ordering" in a linear projection rather than in the full latent space.

After model training with the binary classifier, we perform principal components analysis on the associated latent space of each model (32-dimensional, 64-dimensional, or 128-dimensional, respectively) and project into the top five highest-variance components of the PCA decomposition. Based on previous studies that employed PCA for two-dimensional projections, we expected to observe latent space ordering–that is a partitioning of the data points according to the label predicted by the property predictor–in the top one or two principal components of each model. To our surprise, this was not the case (Fig. 4.7). Further investigation revealed at least some ordering in one or two of the top five principal components for 4/5 of the models. The fifth model, the RNN-Attn, performs so poorly we do not expect to observe ordering no matter how many PCs we analyze. This indicates that despite being explicitly "instructed"–via the loss function–of the importance of the verified-AMP/non-verified-AMP labeling, most models do not "pay attention" to this to the exclusion of other characteristics of the sequences. We may quantify this idea further by investigating the fractional variance explained by different PCs as a proxy for the amount of information contained. When the extent to which these different components capture the overall latent space variance is assessed (Fig. 4.8), we observe that the models differ significantly in how much fractional variance explained (FVE) is contained within the top five PCs, ranging from only about 10% in the Transformer-128 model to over 40% in the AAE-128 model. The more variance is contained within the top five PCs, the more it seems to be concentrated in the first PC (AAE, RNN demonstrate a much more significant drop from PC1 to PC2 compared to the other three models–overall we observe 12% (min) and 35% (max), for the AAE, 4.5% (min) and 5.5% (max), for the RNN FVE respectively, and closer to 1-2% for the other models. Trends are largely independent of latent space dimensionality. In general, the Transformer and WAE models of various sizes contain the least FVE in their top five components, whereas the AAE models contain the most. We observe that in all models other than the Transformer, PC1 is correlated with molecular weight or length (see Sec. 4.3.5), with all other variances being of roughly equivalent magnitudes. Overall, we may interpret this as telling us that the models place greatest importance on length, while other descriptive variables, even those we attempt to emphasize *a priori*, are assigned similar levels of importance.

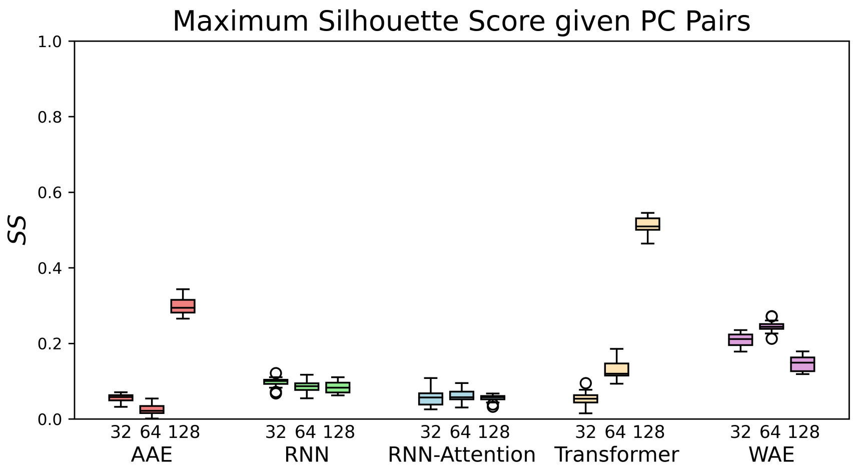

Figure 4.7: The maximal Silhouette scores, SS, for the five models with corresponding three latent sizes. Silhouette scores are shown for the principal components pairs featuring the largest score of the top five PCs calculated with PCA. The PC combinations from left to right are, AAE: [2,3], [1,5], [3,4], RNN: [4,5], [1,2], [1,2], RNN-Attention: [3,4], [3,4], [2,3], Transformer: [1,5], [2,4], [1,3], WAE: [2,5], [2,4], [2,5]. The error bars are generated by bootstrapped sampling of the latent space and calculating a 95% confidence interval computed with the assumption of Bernoulli sampling. The flier points indicate outliers from the interquartile range of the whiskers.

We quantitatively assess the extent to which the clusters of AMPs and non-AMPs are separated from one another in the model latent spaces and pairs of the top five principal components (Fig. 4.7). We do so through the use of the Silhouette score [62] applied to the known labeling of the dataset peptides, which is a well-known metric for assessing cluster distinctness.

$$SS = \frac{\langle r_{\text{out}\rangle} - \langle r_{\text{in}} \rangle}{\max\left(\langle r_{\text{out}} \rangle, \langle r_{\text{in}} \rangle\right)}, \tag{27}$$

We use the Silhouette score as our metric because its boundedness gives it a natural interpretation in this context. Scores that are too close to zero indicate that no clustering has occured, whereas scores too close to one indicate a disconnected latent space, which might mean the model had failed at satisfying the Gaussian prior. Ideally, therefore, we would observe moderate Silhouette scores, and indeed we do. The silhouette score for each PC pair is calculated by taking subsamples of PC pair vectors and computing their Silhouette Score relative to the known verified-AMP/non-verified-AMP properties. Once this has been computed for all pairs of PC's $[1-5]$ we then find the pair that has the highest silhouette score.

If we rank the models by maximum Silhouette score in all pairs of the top five PCs, the Transformer and AAE perform the best, reaching a maximum of $SS = 0.51 \pm 0.02$ for the Transformer at 128 dimensions, and a maximum of $SS = 0.3 \pm 0.03$ for the AAE at 128 dimensions, in PC's [1,3] and PC's [2,3] respectively, with both of them demonstrating moderate maximal clustering ability and an increase in clustering ability with latent space dimensionality. The WAE demonstrates poorer performance, displaying a maximum of $SS = 0.24 \pm 0.02$ in PC's [2,4] at 64 dimensions this performance is largely independent of latent space dimensionality. The RNN and

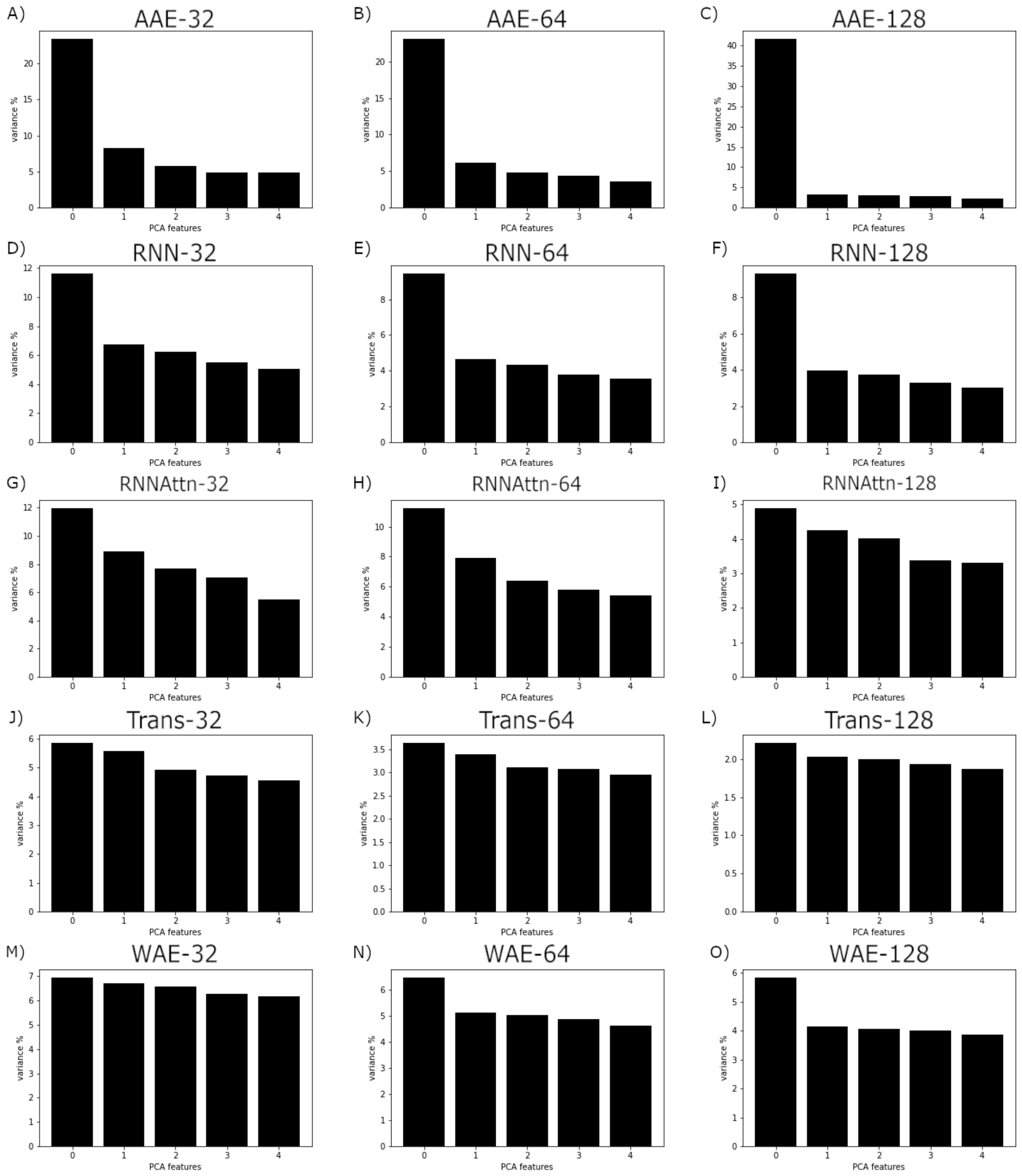

Figure 4.8: The percentage variance explained by each respective PC computed with PCA on the test set data.

RNN-Attention demonstrate the worst abilities as measured by the Silhouette score, with a maximum of $SS = 0.1 \pm 0.02$ in any of the top five PCs for the RNN and $SS = 0.02 \pm 0.02$ for the RNN-Attention.

To better visualize the latent space, in Fig. 4.9, we plot the test set of sequences embedded in the reduced latent representation for each model and color the scatter points according to whether they feature verified-AMPs (orange), or non-verified-AMPs (blue), characteristics. We observe an agreement between the Silhouette scores previously discussed and the visual clustering in the scatter plots. The AAE-128 and Transformer-128 models form more distinct clusters than the other models, though all clusters still overlap as desired.

In summary, the latent spaces of the models were found to exhibit ordering. We quantified the ordering with the Silhouette score. Surprisingly, we found that AMP correlations are not always present in the first or second PC pairs and as such it is important to investigate downstream PCs before ruling out potential correlations in the model latent space. Once a correlation has been identified, the variance explained per PC can serve as an approximate weight factor indicating the importance attributed to this ordering by the model. For the 128-dimensional AAE, summing the variance of the two top PCs, this is about 10%; for the 128-dimensional Transformer, it is about 4%.

4.3.3 Latent Space Sampling and Generative Performance Evaluation

Reconstruction accuracy and clustering of test set embeddings are both useful metrics to ensure that model training is proceeding as expected; however, a more important question for a generative model is whether it is capable of accurately recapitulating the underlying distribution of the data and whether the resultant overall distribution associated with the latent space is appropriate for novel data generation. In this section, we demonstrate the capacity of the model to generate unique and diverse AMP sequences in localized portions of the latent space.

In contrast to the previous section, where we focused on PC pairs, in this section, for ease of calculation and visualization, we employ only a single variable per model variant. We choose the principal component (PC) having the largest Silhouette Score with verified-AMP/non-verified-AMP labeling (see Fig. 4.14A and further discussion of PC identification in Sec 4.3.5). We identify the "width" of the latent space by calculating the mean and standard deviation over the embedded sequences. Then we sample from ten evenly spaced regions that form a "line across" the latent space, limiting the min and max values of the region centers to ± 4 times the standard deviation. The random sampling performed in each region follows a normal distribution with the mean determined by the region center and the standard deviation set to 20% of the standard deviation of the latent space. For external validation on a large number of generated sequences–difficult to perform experimentally–we assess in each neighborhood the probability of generating AMPs ρ_{AMP} (Fig. 4.10) by employing a previously published QSAR model. AMPlify is a deep neural network AMP classifier that has been shown to achieve 93.71% accuracy on the task of AMP classification [3].

As in the previous section, where we demonstrated clustering of the test set embedded in the latent space, we expected to observe partitioning of each PC into areas of low and high AMP probability, and indeed in most cases we did. Because the task given to the models by the property predictor was essentially one of partitioning but there was no constraint on the manner in which to perform the partitioning, a partitioning of the latent space in any direction is equally expected; that

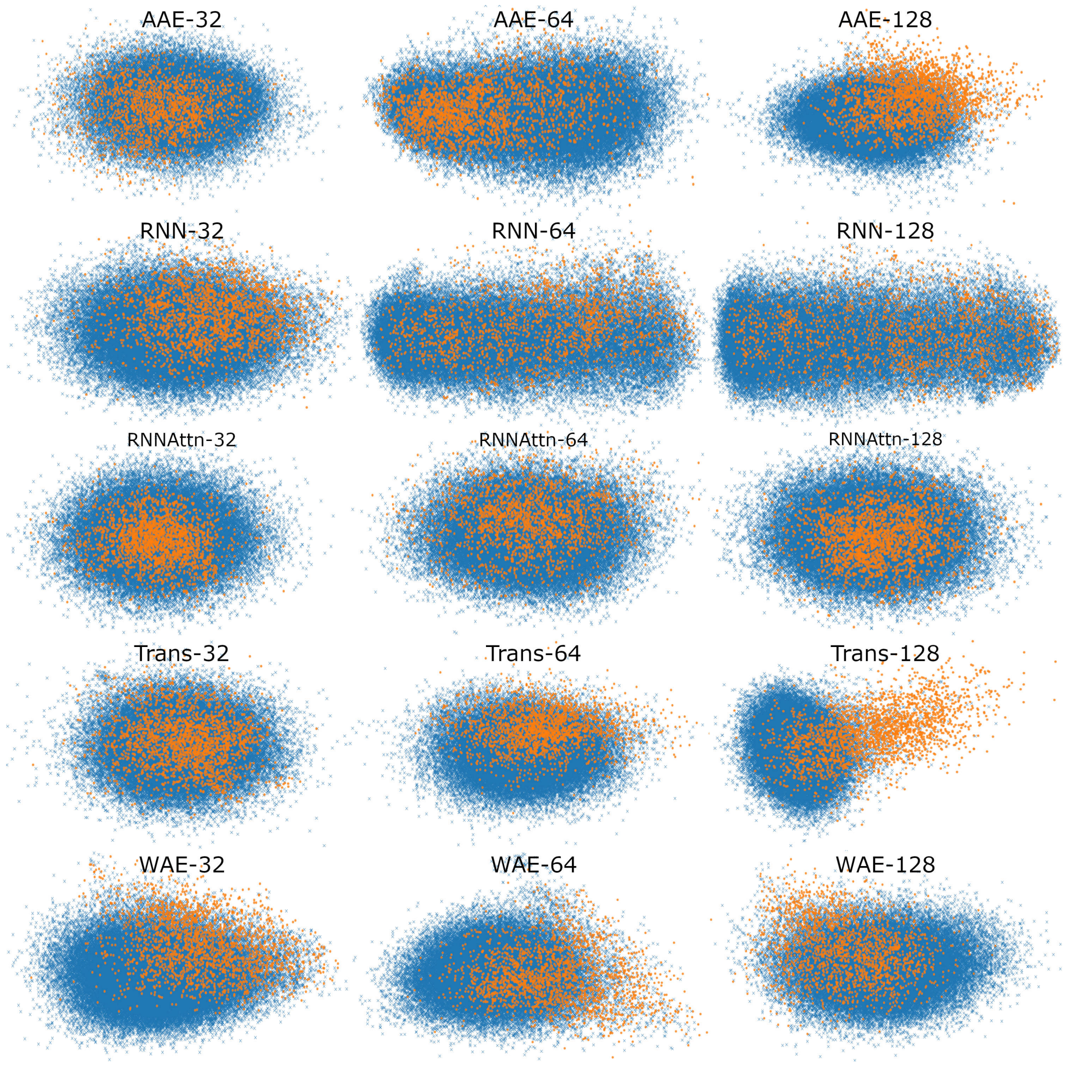

Figure 4.9: Scatter plot of maximally AMP-separating PCs presented as a visualization of the different latent spaces. We embed the test sequences into the latent space, perform PCA, and illustrate the PC pair corresponding to the largest Silhouette score (cf. Fig. 4.7). The PC combinations are as follows, AAE: [2,3], [1,5], [3,4], RNN: [4,5], [1,2], [1,2], RNN-Attention: [3,4], [3,4], [2,3], Trans-former: [1,5], [2,4], [1,3], WAE: [2,5], [2,4], [2,5]. Orange points denote verified-AMP sequences, and blue points denote known non-verified-AMP sequences. Different models correspond to different visible separations in the latent space, though most models demonstrate at least some separation of AMPs and non-AMPs.

is, higher probability of AMPs could be localized in various different areas across the PC.

We observe little or no ($\rho_{\text{AMP}} < 0.4$ for all tested values) partitioning for the 32- and 64-dimensional AAE and RNN-Attention. We observe a semi-linear partitioning–as the PC is increased or decreased, an area of zero AMP probability is followed by a monotonic, almost linear rise in probability–for all other models except the 128 dimension RNN-Attention. Interestingly, the 64 dimensional RNN-Attention model shows a non-linear partitioning of its space with a region of high AMP probability, even though the PCA visualisation and the silhouette score for this model did not demonstrate a clustering of AMPs in its latent space. Thus, we show that predicting experimentally-verified antimicrobial properties, even though many of the non-hits may also be AMP-like in nature, is in many cases sufficient to partition the space into regions of high and low antimicrobial peptide probability in general.

We next evaluate the local properties of the latent space as we interpolate across the individual PCs. In addition to demonstrating generative partitioning in the latent space, we assess the following quantities in each neighborhood and take a global average across the neighborhoods to gain a general understanding of latent space properties in the vicinity of the AMPs: (i) sequence similarity ρ_{sim}, and (ii) Jaccard similarity of 2-mers (J_2) and 3-mers (J_3) in the sequence.

We assess the ability of the model to generate sequences that are dissimilar to one another through the uniqueness [33], $\langle \mu \rangle$, which we compute as an average probability by generating a random sample s_i of 100 sequences and counting the number that are distinct from one another,

$$\langle \mu \rangle = \frac{\sum_n (S_{gen_n} \notin S_{gen_{m \neq n}})}{N_{gen}}. \tag{28}$$

We compute the pairwise similarity between all generated sequences in the neighborhood using the `pairwise2` command from the Biopython [63] package, with the Blosum62 [64] substitution matrix as the scoring function. Pairwise similarity is measured by identifying an optimal sequence alignment from bioinformatics arguments and then computing the scores of amino acids that exhibit matching physicochemical classes.

The Jaccard similarity of two sets is defined as follows,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{29}$$

We assess the Jaccard similarity between all pairs of peptides in the local neighborhood by considering the sets of overlapping k-mers that describe the sequence, where a k-mer of a peptide is a subsequence of length k, and the spectrum of k-mers has previously been demonstrated to contain significant information about AMP properties [65]. We consider k-mers of lengths two and three in this analysis. $0 \leq J(A, B) \leq 1$ by design, and a Jaccard similarity of zero implies that two sequences share no subsequences of length k whereas a Jaccard similarity of one implies that two sequences are composed of identical subsequences.

The sequence similarity score is a relative evaluation based on the Blosum62 matrix scores, whereas the Jaccard similarity score is an absolute measure of the similarity between two sequences where identical sequences return a score of one. It should be noted, however, that the number of matches grows more quickly than the size of the set–i.e. the numerator of the Jaccard score grows more quickly than the denominator. This means that a comparatively low Jaccard score can still indicate a reasonable amount of similarity. Very roughly, for the 2-mers, a Jaccard score of 0.06

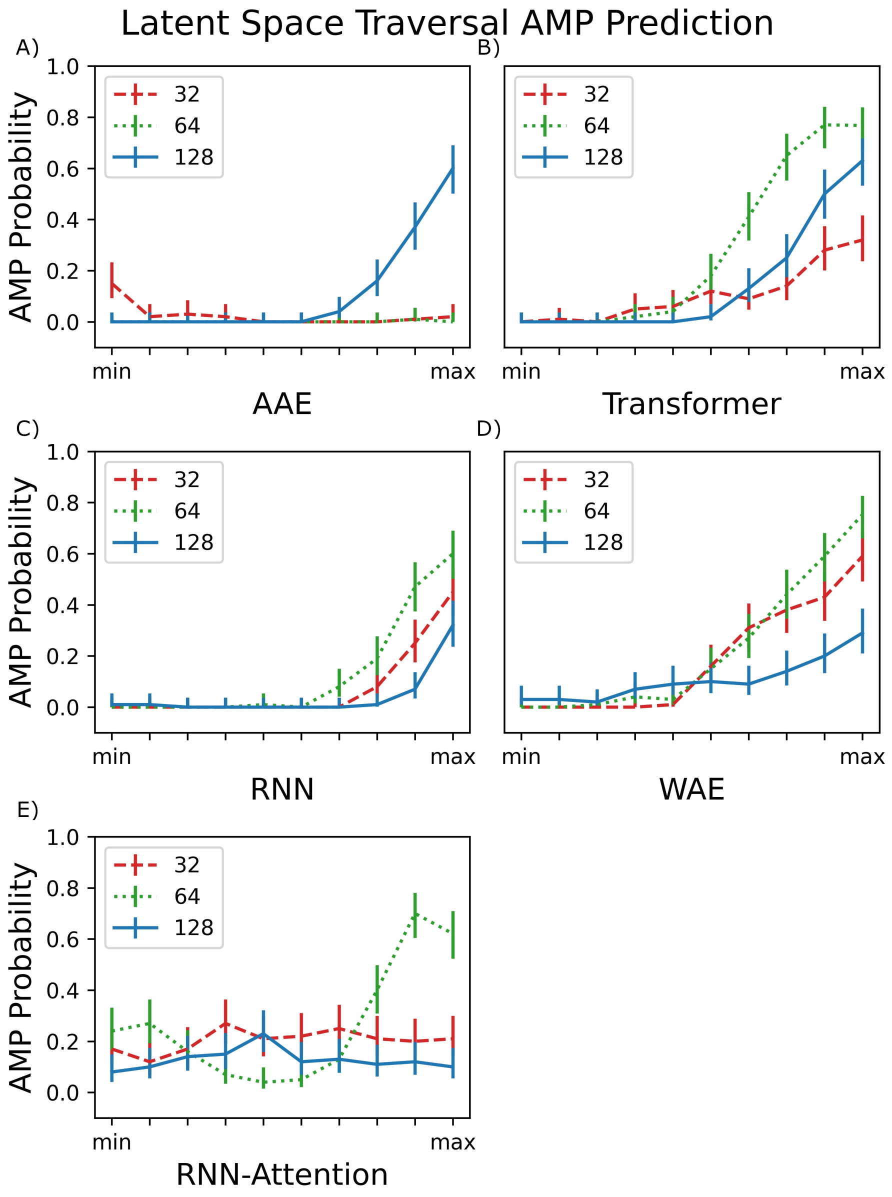

Figure 4.10: Traversing the PC's with highest AMP correlation from the smallest embedded value of each PC (min) to the largest embedded value (max), scaled to the same axis for comparison. The AMPlify QSAR 48 classifier [3] is a binary classifier that returns True or False and an average AMP probability is taken over the 100 local samples. All 5 models and their respective latent dimension variants are plotted. The error bars are generated by calculating a 95% confidence interval on the AMP probabilities computed with the assumption of Bernoulli sampling.

is the expected value for two random sequences of length 50, and it is significantly lower than that for the 3-mers.

The size of the latent space is not seen to have a drastic impact on the sequence similarity of peptides found in a local neighbourhood for any of the models. An interesting dynamic is displayed for the some models, in which as the center of the latent space is approached the samples become slightly more diverse. This is seen by the "U" shape seen especially in the RNN, and WAE models. This is likely due to the Gaussian nature of the space packing more peptides near the center thus increasing sampling diversity.

The Jaccard similarities for the 2-mers and 3-mers are rather low, but still significantly higher than would be predicted by random chance. Due to the nature of the Jaccard score, this indicates a balance between shared and dissimilar fragments, particularly for the Transformer model. The 128 dimensional Transformer has significantly higher 2-mer and 3-mer similarity scores, from 0.24 to 0.34 for 2-mer, and 0.11 to 0.26 for 3-mer, than all other models < 0.22. This is a good indication that local points in the space share similar constituent sequence fragments. Interestingly, we observe heightened similarity for the RNN and WAE models in the region corresponding to heightened AMP probability (i.e. towards the PC maximum value), whereas for the Transformer-128, the trend is opposed: there is a region of heightened similarity near the minimum corresponding to a low probability of AMP-like sequence generation, and a region of somewhat lower probability near the maximum corresponding approximately to the region of heightened probability of AMP-like sequence generation.

Overall, most models perform similarly when reconstructing peptides in a local neighbourhood and this applies to both entire sequences and local k-mer segments. In Sec. 4.3.4, we also show that the models generate disparate peptides not found in the testing or training sets, as desired.

4.3.4 Novelty and Uniqueness of the Entire Generator

We assess the ability of the models to generate AMPs distinct from the training set through the novelty $\langle\eta\rangle$, which we compute as an average probability by generating $N_{\text{gen}} = 1000$ new peptide sequences s_i from random points in the latent space and counting the number that are not also found in the entire training set $\{S_{\text{train}}\}$,

$$\langle\eta\rangle = \frac{\sum_{i=1}^{N_{\text{gen}}} \left(s_i \notin \{S_{\text{train}}\}\right)}{N_{\text{gen}}}. \tag{30}$$

In principle, we expect that the novelty should be high in the absence of overfitting, and that is what we observe (cf. Fig. 4.12).

We assess the ability of the model to generate sequences that are dissimilar to one another through the uniqueness, $\langle\mu\rangle$, which we compute as an average probability by generating a random sample s_i of 1000 sequences and counting the number that are distinct from one another,

$$\langle\mu\rangle = \frac{\sum_n (S_{gen_n} \notin S_{gen_{m\neq n}})}{N_{gen}}. \tag{31}$$

All of the models show a strong uniqueness,
($\langle\eta\rangle \approx 1$) demonstrating that, when the latent space is randomly sampled, they are capable of generating novel peptides not found in the training set. The latent space is also found to be diverse

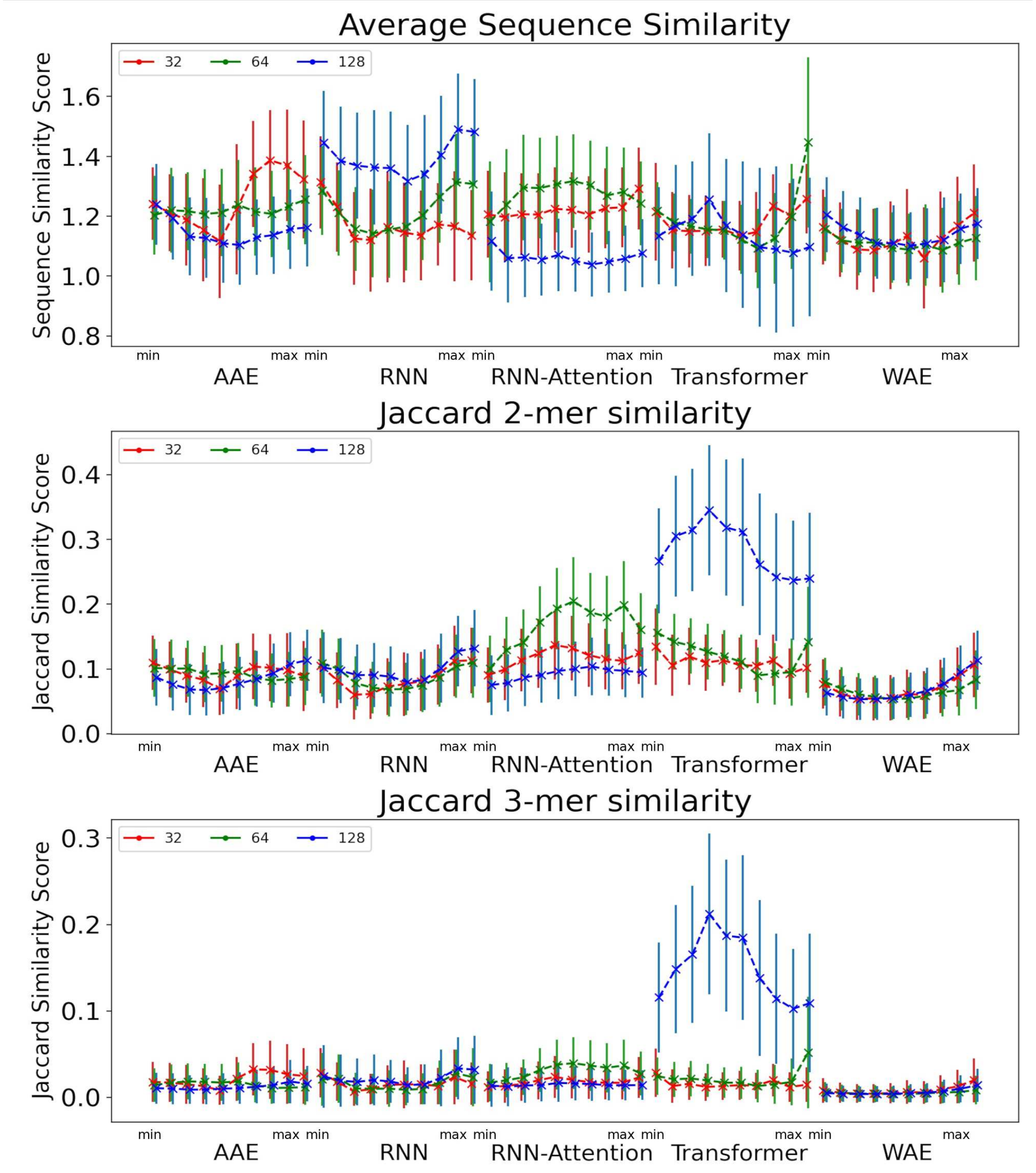

Figure 4.11: Metrics evaluating the generative capacities of models near AMPs in the latent space. The average sequence similarity (top), the Jaccard similarity of the k-mers for k=2 (middle) and the Jaccard similarity of the k-mers for k=3. The error bars are one standard deviation from the mean. Peptides are generated by sampling from the PC with highest AMP correlation from minimum to maximum value.

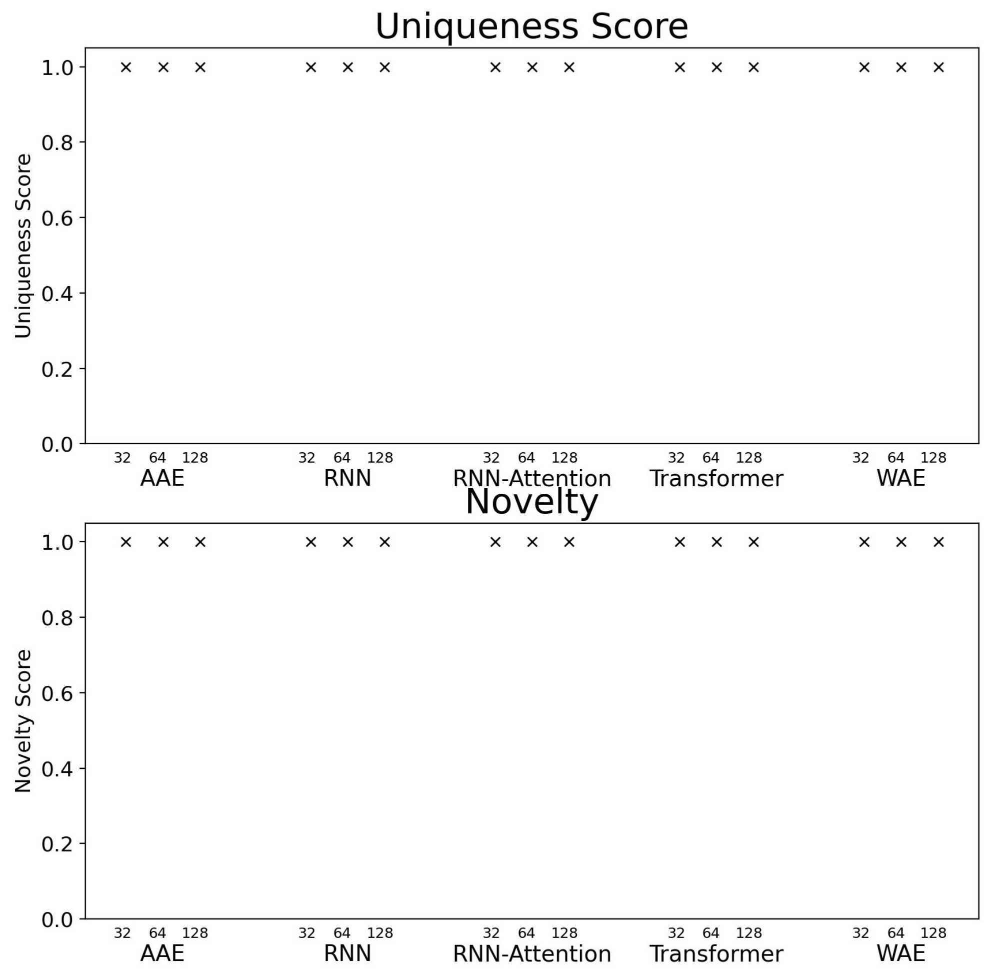

Figure 4.12: The metrics evaluating the uniqueness, novelty and sequence similarity of random samples from the latent space. Error bars for uniqueness and novelty are 95% confidence intervals on the mean. Error bars are not visible because they are below 1% from sampling 1000 latent points.

with high uniqueness of the generated sets, ($\langle \mu \rangle \approx 1$) meaning the models do not produce repetitive samples across the latent space.

4.3.5 Model Interpretability through Linearity and Bridge Variables

A perennial and very difficult problem to solve in deep learning is the question of how to interpret what the model is learning. We address this issue in the following manner: by recognizing that PCA performed on the latent space both creates a lower-dimensional representation which is more easily visualized and also creates a ranking of linear superpositions of the latent dimensions, we argue that a latent space that is less distorted by a PCA analysis is a more interpretable one, and we assess the models on this basis. This also addresses the question of how meaningful the observation of ordering in a linear projection of the latent space is, which, as far as we are aware, has not previously been addressed in the literature. Finally, we consider a set of physicochemical variables of interest and identify potential "bridge variables" associated with different PCs, to aid in their interpretation.

We measure PCA distortion of the latent space with four metrics: trustworthiness $\mathcal{T}$, continuity $\mathcal{C}$, steadiness $\mathcal{S}$, and cohesiveness $\mathcal{H}$ [66]. Trustworthiness is a measure of the introduction of false neighbors and the loss of true neighbors by a point k when projected from high to low dimensions,

$$\mathcal{T}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in \mathcal{U}_i} \max\left(0, r(i,j) - k\right), \tag{32}$$

where N is the number of samples, k is the size of the neighborhood in the reduced space, $\mathcal{U}_i$ is the set of points that are not neighbours in the original space but are now neighbors in the reduced space and $r(i, j)$ is the rank of the sample j according to its distance from i in the original space .

Continuity is a measure of the introduction of false members into a group–a cluster of points–or the loss of true members from a group when projected from high to low dimensions,

$$\mathcal{C}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in \mathcal{V}_i} \max\left(0, r(i,j) - k\right), \tag{33}$$

where N, k and $r(i, j)$ are as previously defined for Trustworthiness, $\mathcal{V}_i$ is the set of points that are neighbours in the original space but are no longer neighbors in the reduced space.

Steadiness is a measure of the loss of existing groups, and cohesiveness is a measure of the introduction of false groups. Briefly, the steps to calculate the steadiness and cohesiveness are are outlined below. We direct the interested reader to Jeon et al. [66] for further details. First, one computes the shared-nearest neighbor distance between points in the original space and in the projected space and constructs a dissimilarity matrix identifying compression and stretching of point pairs. Then average partial distortions are computed by randomly extracting clusters from one space and evaluating their dispersion in the opposite space. Once such partial distortions are known one can aggregate the results into steadiness and cohesiveness,

$$\mathcal{S} = \frac{1 - \sum_{i=1}^{n_{St}} w_i m_i^{compress}}{\sum_{i=1}^{n_{St}} w_i}, \tag{34}$$

$$\mathcal{H} = \frac{1 - \sum_{i=1}^{n_{Co}} w_i m_i^{stretch}}{\sum_{i=1}^{n_{Co}} w_i}, \tag{35}$$

where $w_i m_i^{compress|stretch}$ denote the iterative partial distortion measurements and their corresponding weights. Steadiness and Cohesiveness differ from trustworthiness and continuity in that they evaluate the authentic transformation of clusters of points in the reduced space as opposed to evaluating neighbor-by-neighbor values. All scores run from 0 to 1, where 1 indicates maximum authenticity or minimum spatial distortion.

For all models except the Transformer. all scores are ¿0.5 for all four metrics and the scores tend to hover around 0.75, with some exceptions. While the 128-dimensional Transformer performs best according to the steadiness score(~ 0.8), the Transformer model in general performs worse on all other metrics than the other models, particularly cohesiveness, for which all three Transformer models perform worse than any other model, and in particular the 128-dimensional Transformer displays an exceedingly poor performance of just over 0.25.

Overall we observe it is possible to construct linear projections of model latent spaces with comparatively low overall distortions; however, the model that has thus far performed particularly highly on other metrics (the 128-dimensional Transformer) has the greatest distortion, particularly in terms of cohesiveness, meaning it potentially introduces a significant number of false groupings. The Transformer's comparatively high distortion overall underscores one of the traditional trade-offs of machine learning: with greater power comes lessened interpretability.

Now that we have an idea of how confident we may be in the representation of the PCs for the model latent space, we investigate the identification of "bridge variables" for further interpretation of the latent space. Bridge variables are known quantities of relevance in a scientific problem that show correlations with unknown variables, allowing heightened interpretability of many nonlinear problems [67]. We consider a series of physicochemical properties of peptides that are measurable from sequence alone: Aliphatic Index, Boman Index, Isoelectric Point, Charge pH=3, Charge pH=7, Charge pH=11, Hydrophobicity, Instability Index, and Molecular Weight. We measure each of these properties for the test set sequences using the peptides python package [68]. We employ the Silhouette Score in one dimension to measure correlation between individual PCs and verified-AMP/non-verified-AMP labeling, and we use the Pearson correlation coefficient (r) to measure correlation between individual PCs and the continuous physicochemical properties.

The Aliphatic Index [69] is defined as the relative volume occupied by the aliphatic side chains,

$$Aliphatic\ Index = X(Ala) + aX(Val) + b(X(Ile) + X(Leu)), \tag{36}$$

where X(Ala), X(Val), X(Ile), X(Leu) are mole percent of alanine, valine, isoleucine and leucine. The constants *a* and *b* are the relative volume of aliphatic side chains to that of alanine side chain where, $a \approx 2.9$ and $b \approx 3.9$. The Boman Index [70] is equal to the sum of the solubility values for all residues in a sequence. It gives an estimate of the peptides' likelihood to bind to membranes or other receptors. Peptides with Boman Index > 2.48 are said to exhibit high binding potential. Isoelectric point [71] is the pH at which the net charge of a protein becomes zero. The charges at various pH levels [71] are determined according to the known isoelectric points of the amino acids.

The hydrophobicity is a measure of the degree to which the peptide is hydrophobic, calculated by averaging the hydrophobicity values of each residue by using the scale proposed in Kyte and Doolittle [72]. The instability index predicts whether a peptide will be stable in a test tube as

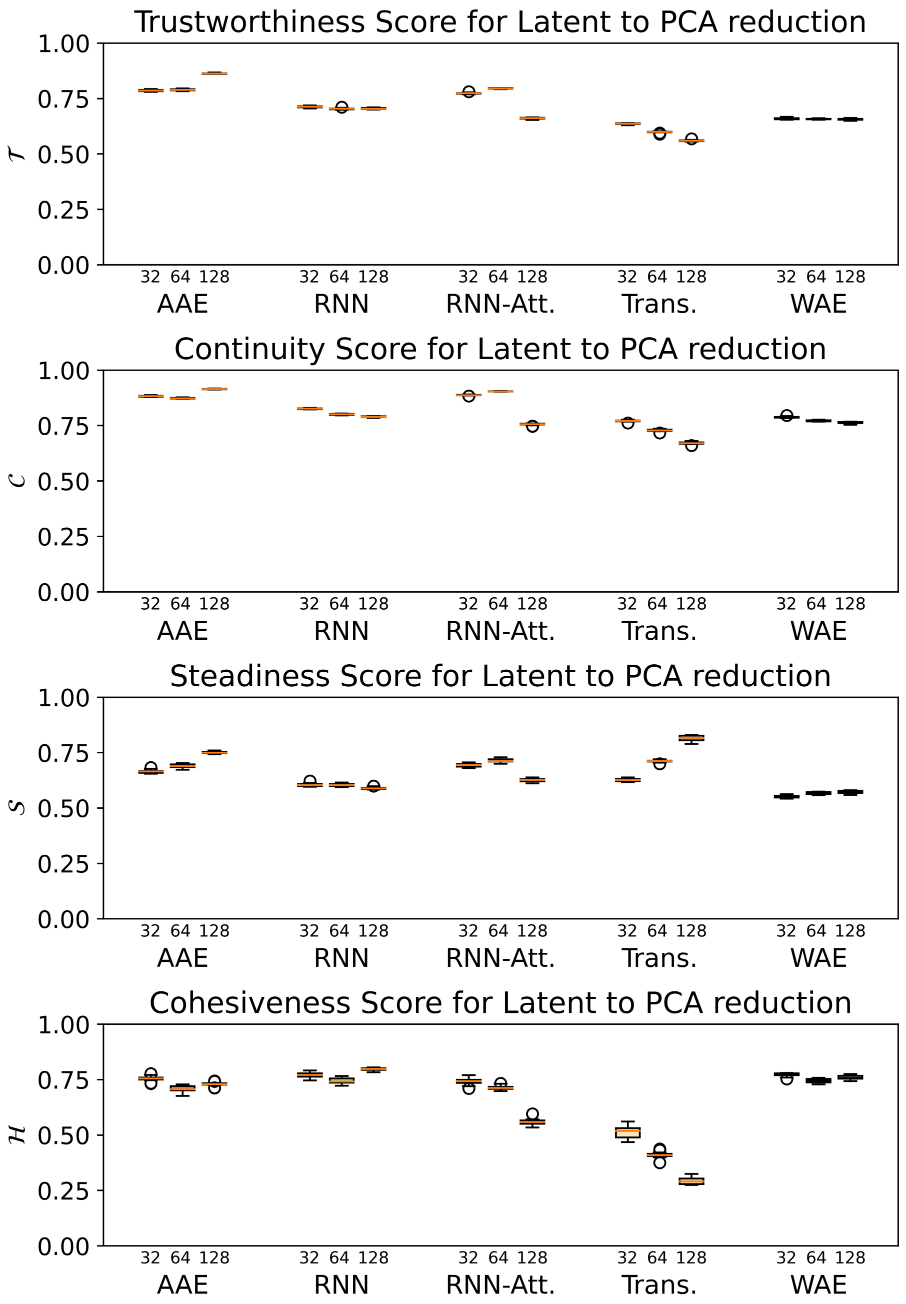

Figure 4.13: The Trustworthiness $\mathcal{T}$, Continuity $\mathcal{C}$, Steadiness $\mathcal{S}$ and Cohesiveness $\mathcal{H}$ projection scores for the five models and three corresponding latent space size variants. The error bars are generated by bootstrapped sampling of the latent space and calculating a 95% confidence interval. The flier points indicate outliers from the interquartile range of the whiskers.

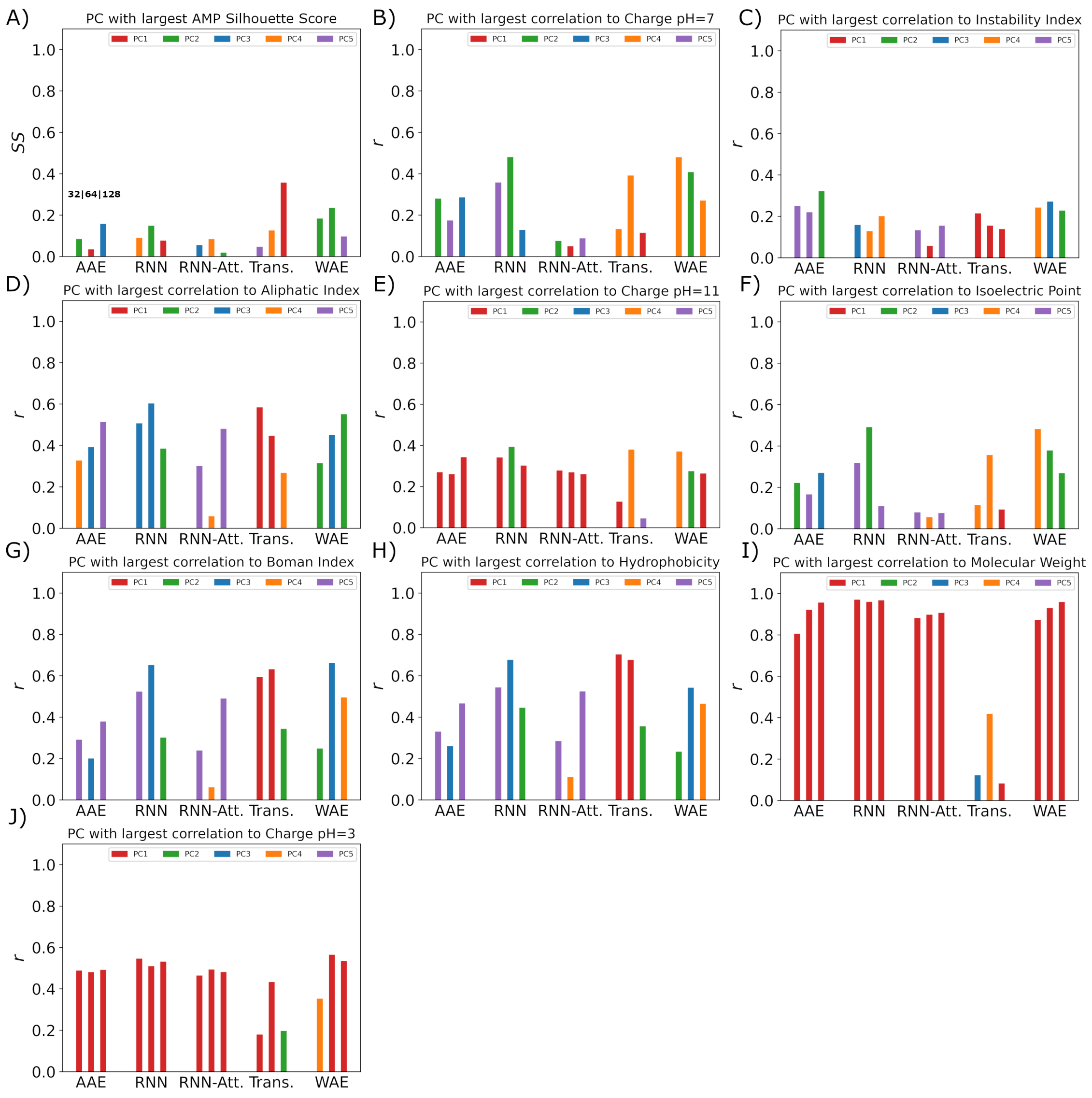

Figure 4.14: Top PC bridge variable correlation scores for all five models and their three latent sizes. (A) The AMP Spearman correlation coefficients (ρ). (B)-(J) The Pearson correlation coefficients (r) for the (B) Charge pH=7, (C) Instability Index, (D) Aliphatic Index, (E) Charge pH=11, (F) Isoelectric Point, (G) Boman Index, (H) Hydrophobicity, (I) Molecular Weight and (J) Charge pH=3. Shown in order for latent sizes 32-128 for each model.

presented in Guruprasad et al. [73]. The molecular weight [74] is the average molecular weight of the peptide found by summing the individual masses of its amino acids and is directly correlated with sequence length.

In Fig. 4.14, we identify the single PC with the highest correlation to each one of the potential bridge variables and illustrate the value of that maximal correlation. We note that although there is a comparatively low Silhouette score in one dimension indicating that the score in two dimensions is a more appropriate quantification of ordering in the system, we may use it to indicate which PC the models consider "most" relevant for verified-AMP ordering, and thus identify whether those PC's simultaneously correlate with physicochemical properties. For example, The 32-dimensional AAE, 64-dimensional WAE, RNN, Transformer and 128-dimensional AAE and RNN models employ one of the top 5 PC's most strongly for AMP ordering (Fig. 4.14A), and the same PC is also correlated with Charge pH=7 (Fig. 4.14B) and Isoelectric Point (Fig. 4.14F) for this model (Fig. 4.14H).

For all models but the Transformer, the first principal component is highly correlated with molecular weight (Fig. 4.14I), which makes sense for the RNN, AAE, and WAE, as all are length-dependent models. That the RNN-Attention model also exhibits this behavior demonstrates the need to commit fully to a Transformer model to avoid a significant component of the model's variance being devoted to sequence length. The comparative performance of the AAE with the Transformer shows, however, that it is not necessary to remove the length dependence to enforce ordering capability in the model, as long as more than the first and second PC are considered. In a more general sense, any model with a high-variance component built in as part of the architecture will likely demonstrate this behavior.

In these plots we find that the models will distribute the correlation across all of the first 5 PC's. Generally the transformer model is observed to make use of the first PC more often and this is expected to be because it does not feature a length based correlation in the first PC.

Using these correlation coefficients it is possible to further characterise the ordering of the latent space and define directions along which interpolation should occur such that desirable characteristics will emerge from generated peptides. Each PC can be interpreted as a linear combination of the respective model's latent dimensions (32, 64 or 128). From this interpretation it is possible to construct a direct relation between the latent dimensions of the model and the physicochemical properties investigated through the PCA mapping; we have made the PCA mappings themselves available on the project Github repository, https://github.com/Mansbach-Lab/latent-spaces-amps.

4.4 Case study: Pipeline for denovo AMP discovery using PCA components

Our pipeline consists of three of the best performing models. We suggest the simultaneous exploration of the Transformer-128 (high reconstruction accuracy, good clustering separation in two dimensions, linear generated sequence partitioning and low interpretability), AAE-128 (moderate reconstruction accuracy, good clustering separation in two dimensions, non-distorted PCA space linear generated sequence partitioning, moderate linearity, medium interpretability–bridge variables identified for AMP PCs), and WAE-128 models (moderate reconstruction accuracy, moderate clustering separation in two dimensions, moderate distortion in PCA space, linear generated sequence partitioning, moderate linearity, high interpretability).

The pipeline takes an AMP of interest as input. The AMP is passed through the encoders of the different models and transformed into a model-specific latent representation. The AMP latent vector is appended to the embedded training data and is transformed into linear representations with PCA (Fig. 4.15).

We then proceed to choose variables of interest to hold constant in the latent space and perturb the others. This will allow us to find new peptides with the properties of interest maintained. We choose to maintain the verified-AMP-correlated PC and the hydrophobicity-correlated PC, to produce the sequences likelier to share experimental AMP-likeness and hydrophobicity– an important AMP property– with GL13K. We note that this procedure could be done with any identified property near any input point of interest. We use the previously mentioned correlation coefficients (Sect. 4.3.5), to find the verified-AMP correlating PC and the top hydrophobicity correlating PC. Having kept track of the PCA vector for our AMP of interest, we perform sampling by keeping our two PC's of interest, AMP and hydrophobicity correlated, fixed and adding gaussian noise to all other PC's. The added noise variance must be tuned individually for each model as the latent space organization is different for each model. The variance tuning is performed by identifying the mean and standard deviation of all training dataset points in PCA space and then sampling Gaussian noise centered at the mean with 1/5 the standard deviation. This noise vector is then summed with the PCA vector of our AMP of interest, thus shifting the sampling location to be near the AMP of interest. Once we have the noisy nearby PCA samples we perform an inverse PCA transformation returning to our latent space vector representation, demonstrating one particular valuable property of the PCA approach. We then pass the noisy latent samples to the decoder which will generate the desired candidate peptide sequences.

We demonstrate the use of the pipeline on GL13K. For each model, we generate five different sequences and display their properties (Figs. 4.16,4.17,4.18). In the ideal case we should observe little change in Hydrophobicity of the samples while the other physicochemical properties should vary and in general this is what we observe. We observe that certain properties of random sequences generated in the neighborhoods tend to remain more constant irrespective of model (Aliphatic Index, Charge at pH=3, Molecular Weight, Isoelectric Point) and certain properties tend to be more variable (Boman index, Charge at pH=11, Instability Index), while certain properties are more dependent on the model. Hydrophobicity varies the most in the AAE model, and the least in the WAE model. One benefit of employing multiple models is the ability to sample different local neighborhoods of GL13K, with potentially different properties; another benefit is including both more interpretable and more high-performing models to generate samples.

The results from the pipeline for the AAE-128, Transformer-128 and WAE-128 presented in the figures (Figs. 4.16,4.17,4.18) show that by locking PC's of interest and sampling nearby points along other PC's we can generate novel peptides that have similar properties to the original GL13K. While some generated samples feature drastic changes in certain properties such as the fourth peptide from the AAE-128 with a charge of 1 at pH 11 or both generated peptides with -1 charges at pH 11 for the Transformer-128 and WAE-128 models, most properties fall near the original GL13K sequence properties.

This general pipeline can be extended to any peptide function by identifying the top PC's correlating with the desired function and keeping those fixed as noise is added to the remaining

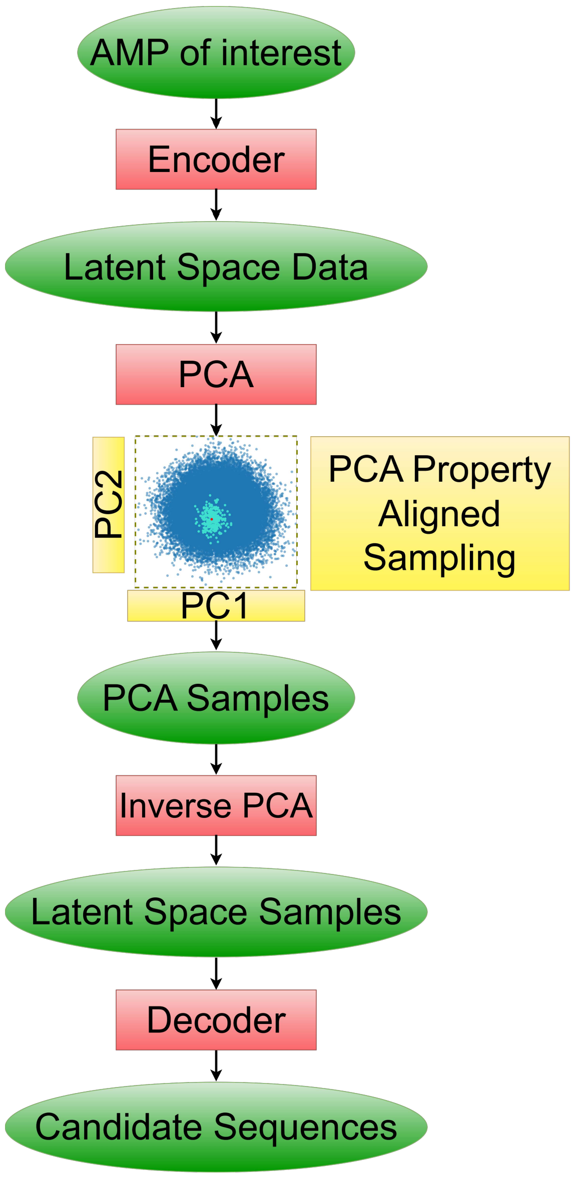

Figure 4.15: Schematic of the suggested pipeline for generation of sequences of interest.

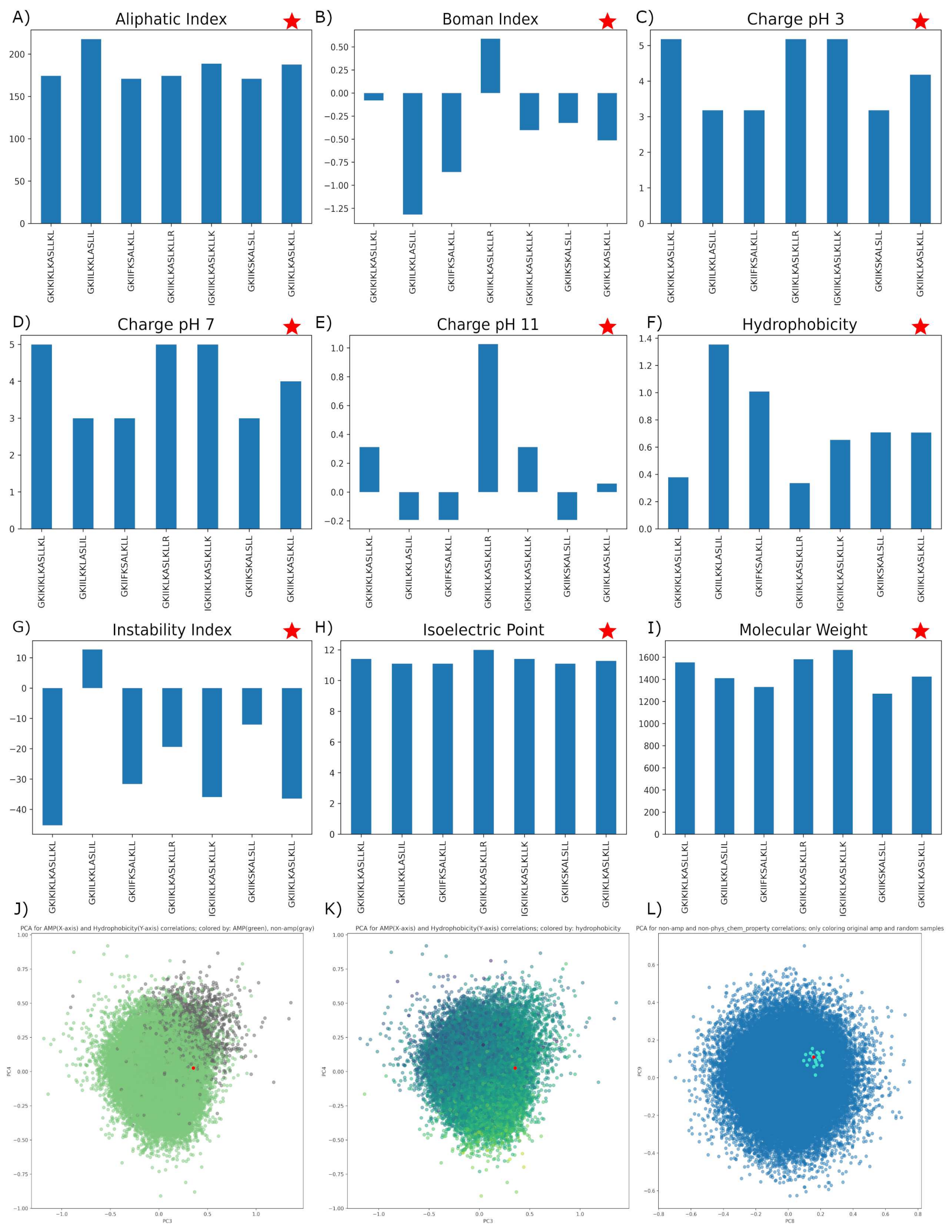

Figure 4.16: Pipeline sampling for the AAE-128 model. The top three rows plot the physicochemical properties of the six generated sequences and the properties of GL13K (right-most bar). The bottom row first plots the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the NON-AMP points as green, AMP points as grey and GL13K as a red point (left). Then the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the points according to their respective hydrophobicities and GL13K as a red point (center). The final rightmost plot shows two different PC's and colors the test set a dark blue, the randomly sampled points in light blue and GL13K as a red point. Having locked the PC's of interest we plot different PC's such that sampled points do not overlap GL13K.

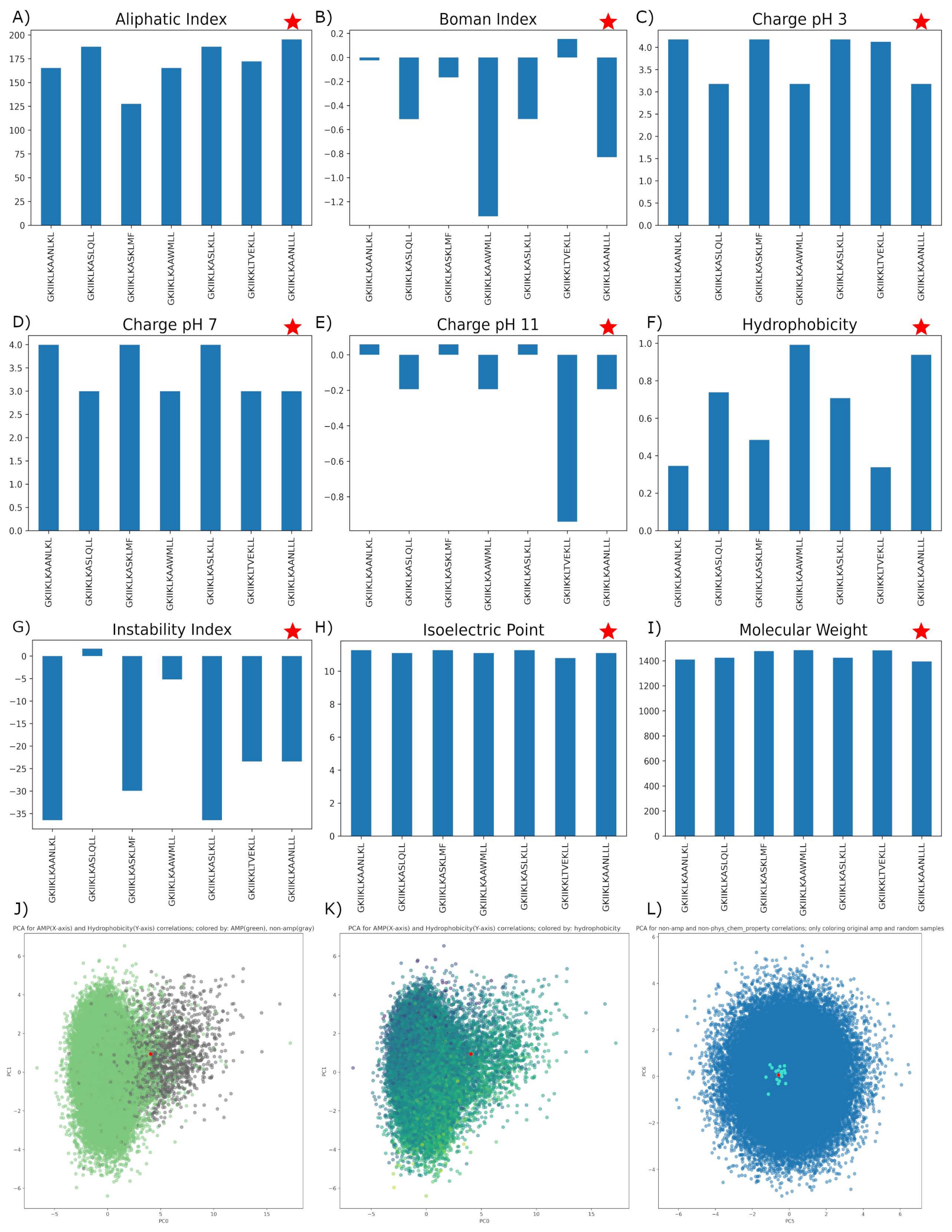

Figure 4.17: Pipeline sampling for the Transformer-128 model. The top three rows plot the physicochemical properties of the six generated sequences and the properties of GL13K (right-most bar). The bottom row first plots the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the NON-AMP points as green, AMP points as grey and GL13K as a red point (left). Then the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the points according to their respective hydrophobicities and GL13K as a red point (center). The final rightmost plot shows two different PC's and colors the test set a dark blue, the randomly sampled points in light blue and GL13K as a red point.Having locked the PC's of interest we plot different PC's such that sampled points do not overlap GL13K.

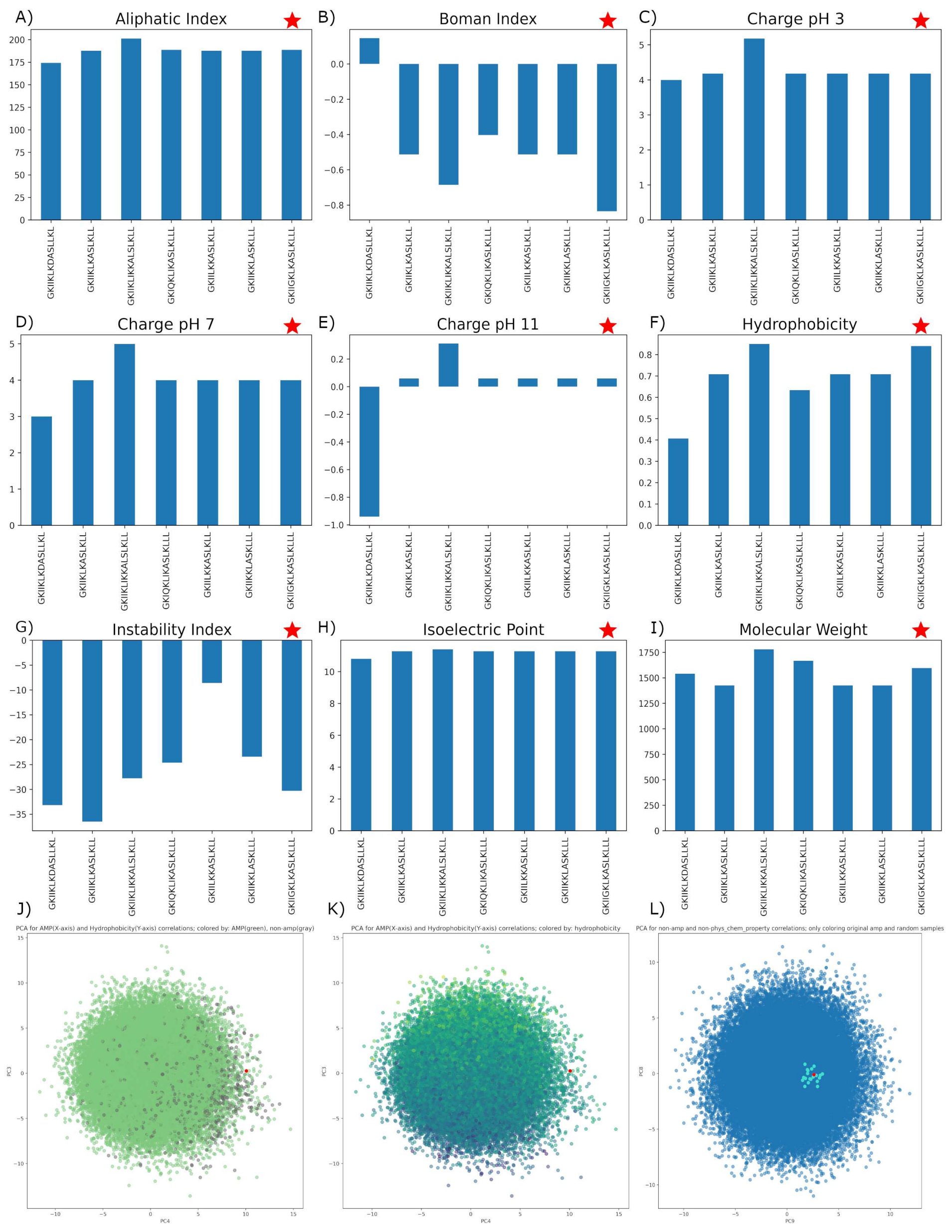

Figure 4.18: Pipeline sampling for the WAE-128 model. The top three rows plot the physicochemical properties of the six generated sequences and the properties of GL13K (right-most bar). The bottom row first plots the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the NON-AMP points as green, AMP points as grey and GL13K as a red point (left). Then the PCA reduced latents of the test set for the AMP correlating PC and the Hydrophobicity correlating PC and colors the points according to their respective hydrophobicities and GL13K as a red point (center). The final rightmost plot shows two different PC's and colors the test set a dark blue, the randomly sampled points in light blue and GL13K as a red point.Having locked the PC's of interest we plot different PC's such that sampled points do not overlap GL13K.

PCA vectors in order to sample in the functional neighbourhood it could be employed for.

4.5 Discussion and Conclusions

We have trained five deep learning models with VAE-like latent spaces on a library of sequences of short proteins and assessed the characteristics of the resultant models, including a careful examination of the properties of the latent spaces associated with them. We show that the models have the capability to create smooth, continuous latent spaces that can be sampled for *de novo* AMP generation and optimization. The inclusion of a simultaneously-trained property prediction network disentangles verified-AMP and non-verified-AMP sequences; however, to our surprise we find that, unlike in previous studies, this is not always apparent in the first two principal components derived by PCA. Different models associate different information with highly-varying PCs, and different models vary drastically in the ways in which they encode variance, and, hence, information. It is important to note that this presents a challenge for the incorporation of domain knowledge, since we see that the model does not necessarily place greater emphasis on user-provided information. Furthermore, it sounds a note of caution in the interpretation of latent space orderings, since observed orderings may occupy only a small fraction of the informational content in the model latent space. We have also addressed the question of how meaningful the use of PCA is as a tool for indicating properties of VAE-like latent spaces and argued that the less distortion imposed by PCA upon the different neighborhoods and interactions between points in the manifold, the more clearly interpretable the latent space is. The analysis that we present here may be applied to other short peptides of interest, such as anti-cancer peptides. Based on our results, we would suggest retraining our Transformer, AAE, and WAE models in conjunction with a new cancer/anti-cancer property predictor that relates to the property of interest. One could then employ a similar analysis and pipeline to assess the quality of the resultant latent spaces and generate sequences of interest.

We have investigated the use of the principal components with the highest clustering of verified AMP properties for *de novo* AMP generation and showed that our models generate highly diverse and unique sequences, with comparatively low sequence similarity in local neighborhoods. Despite the low similarity and the use of a predictor trained on experimentally-verified AMP properties rather than direct knowledge of AMP-like-ness, all models but the 32 and 128-dimensional RNN-Attention and 64-dimensional AAE are capable of successfully partitioning a single coordinate in the latent space into regions that generate AMP-like sequences with high probability and regions that generate non-AMP-like sequences with high probability. We observe that the capacity of the model to reconstruct input sequences is not clearly linked to its ability to partition the space, and we add our voices to a number of cautions against the over-use of reconstruction accuracy as a metric for generative models.

We have evaluated the extent to which models order their latent spaces according to standard peptide physicochemical properties that they are not trained on and identify the principal components most strongly correlated to given properties. We find that the models will order any of the first five principal components investigated according to physicochemical properties but when a model needs to assign a larger proportion of the variance to learning peptide length the first component is usually correlated with length / molecular weight. Indeed, only the 128-dimensional Transformer eschews length as a consideration almost entirely (no length dependence observed in the reconstruction accuracy, small correlation between any top five PC and molecular weight). The

Transformer in general clearly has the capacity to function independent of the length but demonstrates a more rapid drop in performance as the latent space dimensionality is decreased than the RNN, WAE, or AAE. We speculate that this is due to the nature of the task being that discriminating between lengths can actually discourage models from overfitting; that is, from simply "memorizing" the answers, and may encourage a more meaningful lower-dimensional representation, although we also note that the 128-dimensional Transformer shows by far the most heightened local fragment-based similarity in its verified-AMP-relevant PC. This could suggest a model relying on fundamentally different information from the others.

In terms of other relevant physicochemical variables, we observe moderate correlations ($0.35 \lesssim r$Pearson $\lesssim 0.65$) between at least one PC and isoelectric point / charge at pH 7 in the AAE-64, RNN-32, RNN-64, Transformer-64 and all WAE models and moderate correlation between at least one PC and hydrophobicity in all but the RNN-Attention-32. As these are traditional hallmarks employed for AMP design, this is desirable behavior, and in particular aids in the interpretability of the models through a linear mapping of the latent space variables to straightforward "bridge variables" for most models. It also shows that the models are capable of identifying relevant properties from sequences alone, despite being trained only with a binary property predictor.

We may further employ the bridge variables in conjunction with the probability of AMP generation in a single PC (Fig. 4.10) to aid in the interpretability of the models. In our case models demonstrating a monotonic linear increase in probability, especially those reaching a prediction probability of > 0.6 (128-dimensional AAE, 64-dimensional RNN, 64-dimensional RNN-Attention, 64 and 128-dimensional Transformer and 32 and 64-dimensional WAE models) are arguably the easiest to interpret, since we now essentially have a single linear mapping from the latent space to AMP probability, which for the RNNs and WAEs are comparatively non-distorted from the original space (cf. Sec. 4.3.5).

We demonstrate a trade-off between model complexity and model interpretability under this paradigm and suggest that for optimized design of AMPs in a continuous latent space, it may be appropriate to perform the optimization in multiple different latent spaces, using a similar philosophy to that of ensemble voting. We do a short case study to show one way this might be implemented, and indeed, in future work we plan to use this as a starting point for an active learning procedure to traverse these spaces and perform multi-scale molecular dynamics simulations upon relevant points.

In the future, we plan to investigate a phenomenon termed selective latent memories due to Kullback-Leibler Divergence constraining. This effect is observed during training and causes a drop in the entropy of certain latent dimensions when the KLD is minimized. In addition, we will generalize from a binary AMP/non-AMP classifier to a multi-class predictor capable of grouping sequences by expected mechanism of action. Finally, as a prelude to the previously mentioned active learning traversal of the space, we plan to investigate the incorporation of structural data into the models, perhaps leveraging the recent success of AlphaFold2 [30] and similar structural prediction algorithms.

The models developed in this research used deep learning to discover embeddings for sequences of amino acids but future work should investigate other peptide representations such as protein structure distance graphs which can embed structural information and SMILES strings used in the world of small drug design. SMILES strings encode chemical information into a sequence of characters thus allowing the models to learn chemical distinctions between the amino acids.

Overall, we have performed a thorough qualitative and quantitative analysis of the latent spaces of five different generative models for AMP design, identifying strengths and weaknesses of each, as well as developing a suite of analysis methods for latent space design and sampling in the context of generative deep learning of AMP sequences. We provide a much-needed set of benchmarking protocols in this nascent area of research.

Chapter 5

Final Remarks and Future Work

The research presented in Chapter 4 demonstrated the power of generative models on the task of de novo antimicrobial peptide generation as well as their limitations, especially in terms of interpretability. While it is still early days for generative deep learning in the past few months alone the sub-field of deep learning has grown incredibly fast with significant progress in many biophysics topics including, large-language-models as biology experts [75], protein structure prediction [30][23], protein simulation [76] and protein docking [77].

The recent release of the Alphafold2 model [30], with code made publicly available on github, has served as a basis for further developments relating to protein structure prediction. Alongside Alphafold2 the release and open sourcing of Stable Diffusion a text-to-image generative model has inspired biophysicists to look into "diffusion" models as powerful generative tools [76], [77].

Graph neural networks are models that use deep learning on graph data [78]. Throughout training a graph neural network will learn how to update graph nodes and edges to achieve the desired goal. Models that incorporate graph neural networks, often through Transformers, seem particularly promising as their representational capacity is not bound by the limitations of a sentence-based sequence, as found in the 5 models presented in this research. Instead each protein could be represented as a unique graph and the attention mechanism can now act over the entire graph. These models are particularly interesting in their structural representations of proteins given it is now possible to input one sequence with multiple corresponding structures, a feature relevant for analysis of different folding states as investigated through molecular dynamics simulations. A future project we would like to work on trains a graph neural network on simulation data from AMPs in order to predict folding states and relate the deep learning component to the simulation counterpart.

Overall the the research presented in this thesis is to serve as a stepping stone on the path to solving AMP design with generative deep learning models. Signs of the tremendous capability of the models involved joined with palpable excitement pushing the field forward fill me with hope that a day will come in the near future where we can work with these systems, in a cooperative and transparent manner, to help humanity guide biological processes in their interest.

Bibliography

[1] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge: MIT Press, 2016.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *ArXiv*, vol. preprint, p. ArXiv ID:1706.03762v5, 2017.

[3] W. Chen and A. L. Ferguson, "Molecular enhanced sampling with autoencoders: On-the-fly collective variable discovery and accelerated free energy landscape exploration," *Journal of Computational Chemistry*, vol. 39, no. 25, pp. 2079–2102, sep 2018.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, dec 1989.

[6] J. Schmidhuber, "Learning Complex, Extended Sequences Using the Principle of History Compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, mar 1992.

[7] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," *Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014.

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, no. Ml, dec 2013, p. ArXiv ID: 1312.6114.

[9] A. Van Den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv*, vol. preprint, p. arXiv ID:1711.00937, 2017.

[10] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," *arXiv*, vol. preprint, p. arXiv ID:2007.03898, 2021.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv*, vol. preprint, p. arXiv ID:1406.2661, jun 2014.

[12] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," *arXiv*, vol. preprint, p. ArXiv ID:1511.05644, 2015.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, sep 2014, pp. 1–15.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv*, vol. preprint, p. arXiv ID:1907.11692, 2019.

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, vol. preprint, p. arXiv ID:1810.04805, 2019.

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, may 2020.

[17] Centers for Disease Control and Prevention, "Antibiotic resistance threats in the United States," National Center for Emerging Zoonotic and Infectious Diseases (U.S.), Atlanta, Georgia, Tech. Rep., nov 2019.

[18] C. L. Ventola, "The antibiotic resistance crisis: part 1: causes and threats." *P and T : a peer-reviewed journal for formulary management*, vol. 40, no. 4, pp. 277–83, apr 2015.

[19] K. Bush, P. Courvalin, G. Dantas, J. Davies, B. Eisenstein, P. Huovinen, G. A. Jacoby, R. Kishony, B. N. Kreiswirth, E. Kutter, S. A. Lerner, S. Levy, K. Lewis, O. Lomovskaya, J. H. Miller, S. Mobashery, L. J. V. Piddock, S. Projan, C. M. Thomas, A. Tomasz, P. M. Tulkens, T. R. Walsh, J. D. Watson, J. Witkowski, W. Witte, G. Wright, P. Yeh, and H. I. Zgurskaya, "Tackling antibiotic resistance," *Nature Reviews Microbiology*, vol. 9, no. 12, pp. 894–896, dec 2011.

[20] J. Li, J. J. Koh, S. Liu, R. Lakshminarayanan, C. S. Verma, and R. W. Beuerman, "Membrane active antimicrobial peptides: Translating mechanistic insights to design," *Frontiers in Neuroscience*, vol. 11, p. Article 73, 2017.

[21] M. Magana, M. Pushpanathan, A. L. Santos, L. Leanse, M. Fernandez, A. Ioannidis, M. A. Giulianotti, Y. Apidianakis, S. Bradfute, A. L. Ferguson, A. Cherkasov, M. N. Seleem, C. Pinilla, C. de la Fuente-Nunez, T. Lazaridis, T. Dai, R. A. Houghten, R. E. Hancock, and G. P. Tegos, "The value of antimicrobial peptides in the age of resistance," *The Lancet Infectious Diseases*, vol. 20, no. 9, pp. e216–e230, 2020.

[22] J. J. Schneider, A. Unholzer, M. Schaller, M. Schäfer-Korting, and H. C. Korting, "Human defensins," *Journal of Molecular Medicine*, vol. 83, no. 8, pp. 587–595, aug 2005.

[23] J. Koehbach and D. J. Craik, "The Vast Structural Diversity of Antimicrobial Peptides," *Trends in Pharmacological Sciences*, vol. 40, no. 7, pp. 517–528, jul 2019.

[24] J. Lei, L. Sun, S. Huang, C. Zhu, P. Li, J. He, V. Mackey, D. H. Coy, and Q. He, "The antimicrobial peptides and their potential clinical applications," *Am J Transl Res*, vol. 11, no. 7, pp. 3919–3931, 2019.

[25] F. Guilhelmelli, N. Vilela, P. Albuquerque, L. d. S. Derengowski, I. Silva-Pereira, and C. M. Kyaw, "Antibiotic development challenges: The various mechanisms of action of antimicrobial peptides and of bacterial resistance," *Frontiers in Microbiology*, vol. 4, no. DEC, pp. 1–12, 2013.

[26] A. Bin Hafeez, X. Jiang, P. J. Bergen, and Y. Zhu, "Antimicrobial Peptides: An Update on Classifications and Databases," *International Journal of Molecular Sciences*, vol. 22, no. 21, p. 11691, oct 2021.

[27] G. Wang, "The antimicrobial peptide database provides a platform for decoding the design principles of naturally occurring antimicrobial peptides," *Protein Science*, vol. 29, no. 1, pp. 8–18, jan 2020.

[28] S. P. Piotto, L. Sessa, S. Concilio, and P. Iannelli, "YADAMP: yet another database of antimicrobial peptides," *International Journal of Antimicrobial Agents*, vol. 39, no. 4, pp. 346–351, apr 2012.

[29] F. H. Waghu and S. Idicula-Thomas, "Collection of antimicrobial peptides database and its derivatives: Applications and beyond," *Protein Science*, vol. 29, no. 1, pp. 36–42, jan 2020.

[30] P. Cramer, "AlphaFold2 and the future of structural biology," *Nature Structural and Molecular Biology*, vol. 28, no. 9, pp. 704–705, sep 2021.

[31] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, jun 2019.

[32] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, feb 2018.

[33] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Johansson, H. Chen, S. Nikolenko, A. Aspuru-Guzik, and A. Zhavoronkov, "Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models," *Frontiers in Pharmacology*, vol. 11, pp. 1–19, 2020.

[34] A. Tucs, D. P. Tran, A. Yumoto, Y. Ito, T. Uzawa, and K. Tsuda, “Generating Ampicillin-Level Antimicrobial Peptides with Activity-Aware Generative Adversarial Networks,” *ACS Omega*, vol. 5, no. 36, pp. 22 847–22 851, sep 2020.

[35] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, “The rise of deep learning in drug discovery,” *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[36] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, “Generating focused molecule libraries for drug discovery with recurrent neural networks,” *ACS Central Science*, vol. 4, no. 1, pp. 120–131, 2018.

[37] P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. dos Santos, P.-Y. Chen, Y. Y. Yang, J. P. K. Tan, J. Hedrick, J. Crain, and A. Mojsilovic, “Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations,” *Nature Biomedical Engineering*, vol. 5, no. 6, pp. 613–623, jun 2021.

[38] T. Sercu, S. Gehrmann, H. Strobelt, P. Das, I. Padhi, C. D. Santos, K. Wadhawan, and V. Chenthamarakshan, “Interactive Visual Exploration of Latent Space (IVELS) for peptide auto-encoder model selection,” in *Deep Generative Models for Highly Structured Data, DGS@ICLR 2019 Workshop*. International Conference on Learning Representations, ICLR, 2019.

[39] C. M. Van Oort, J. B. Ferrell, J. M. Remington, S. Wshah, and J. Li, “AMPGAN v2: Machine Learning-Guided Design of Antimicrobial Peptides,” *Journal of Chemical Information and Modeling*, vol. 61, no. 5, pp. 2198–2207, 2021.

[40] D. Nagarajan, T. Nagarajan, N. Roy, O. Kulkarni, S. Ravichandran, M. Mishra, D. Chakravortty, and N. Chandra, “Computational antimicrobial peptide design and evaluation against multidrug-resistant clinical isolates of bacteria,” *Journal of Biological Chemistry*, vol. 293, no. 10, pp. 3492–3509, 2018.

[41] A. Capecchi, X. Cai, H. Personne, T. Köhler, C. van Delden, and J. L. Reymond, “Machine learning designs non-hemolytic antimicrobial peptides,” *Chemical Science*, vol. 12, no. 26, pp. 9221–9232, 2021.

[42] O. Dollar, N. Joshi, D. A. C. Beck, and J. Pfaendtner, “Attention-based generative models for de novo molecular design,” *Chemical Science*, vol. 12, no. 24, pp. 8362–8372, 2021.

[43] O. Prykhodko, S. V. Johansson, P.-C. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist, and H. Chen, “A de novo molecular generation method using latent vector based generative adversarial network,” *Journal of Cheminformatics*, vol. 11, no. 1, p. 74, dec 2019.

[44] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, “Grammar Variational Autoencoder,” in *34th International Conference on Machine Learning, ICML 2017*, mar 2017, p. ArXiv ID: 1703.01925.

[45] F. Grisoni, M. Moret, R. Lingwood, and G. Schneider, "Bidirectional Molecule Generation with Recurrent Neural Networks," *Journal of Chemical Information and Modeling*, vol. 60, no. 3, pp. 1175–1183, mar 2020.

[46] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song, "Syntax-directed variational autoencoder for structured data," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, p. ArXiv ID: 1802.08786.

[47] B. Dai, Z. Wang, and D. Wipf, "The usual suspects? Reassessing blame for VAE posterior collapse," *arXiv*, p. ArXiv ID: 1912.10702, 2019.

[48] R. Chowdhury, N. Bouatta, S. Biswas, C. Floristean, A. Kharkar, K. Roy, C. Rochereau, G. Ahdritz, J. Zhang, G. M. Church, P. K. Sorger, and M. AlQuraishi, "Single-sequence protein structure prediction using a language model and deep learning," *Nature Biotechnology*, vol. 40, no. 11, pp. 1617–1623, nov 2022.

[49] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, "Unified rational protein engineering with sequence-based deep representation learning," *Nature Methods*, vol. 16, no. 12, pp. 1315–1322, 2019.

[50] H. J. Kim, S. E. Hong, and K. J. Cha, "seq2vec: Analyzing sequential data using multi-rank embedding vectors," *Electronic Commerce Research and Applications*, vol. 43, no. August 2019, p. 101003, 2020.

[51] A. Bateman, "UniProt: A worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, no. D1, pp. D506–D515, jan 2019.

[52] S. A. Pinacho-Castellanos, C. R. García-Jacas, M. K. Gilson, and C. A. Brizuela, "Alignment-Free Antimicrobial Peptide Predictors: Improving Performance by a Thorough Analysis of the Largest Available Data Set," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 3141–3157, 2021.

[53] S. Ramazi, N. Mohammadi, A. Allahverdi, E. Khalili, and P. Abdolmaleki, "A review on antimicrobial peptides databases and the computational tools," *Database*, vol. 2022, no. February, pp. 1–17, mar 2022.

[54] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick," *arXiv*, vol. preprint, p. ArXiv ID:1506.02557, jun 2015.

[55] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems*, no. Nips, jun 2016, p. ArXiv ID: 1606.04934.

[56] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, "Wasserstein auto-encoders," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 1–20.

[57] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, “Alias-Free Generative Adversarial Networks,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. NeurIPS, jun 2021, pp. 438–448.

[58] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, dec 2019, pp. 8107–8116.

[59] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012.

[60] J. Vig, “A multiscale visualization of attention in the transformer model,” in *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, 2019, pp. 37–42.

[61] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding,” *arXiv*, vol. preprint, p. arXiv ID:2205.11487, 2022.

[62] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, no. C, pp. 53–65, 1987.

[63] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. De Hoon, “Biopython: Freely available Python tools for computational molecular biology and bioinformatics,” *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[64] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, pp. 10 915–10 919, 1992.

[65] F. Cipcigan, A. P. Carrieri, E. O. Pyzer-Knapp, R. Krishna, Y.-W. Hsiao, M. Winn, M. G. Ryadnov, C. Edge, G. Martyna, and J. Crain, “Accelerating molecular discovery through data and physical sciences: Applications to peptide-membrane interactions,” *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241744, jun 2018.

[66] H. Jeon, H. K. Ko, J. Jo, Y. Kim, and J. Seo, “Measuring and Explaining the Inter-Cluster Reliability of Multidimensional Projections,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 551–561, 2022.

[67] R. A. Mansbach and A. L. Ferguson, “Machine learning of single molecule free energy surfaces and the impact of chemistry and environment upon structure and dynamics,” *The Journal of Chemical Physics*, vol. 142, no. 10, p. 105101, mar 2015.

[68] M. Larralde, “Peptides,” p. github.com/althonos/peptides.py, 2021.

[69] A. Ikai, “Thermostability and aliphatic index of globular proteins,” *Journal of Biochemistry*, vol. 88, no. 6, pp. 1895–1898, 1980.

[70] H. G. Boman, “Antibacterial peptides: Basic facts and emerging concepts,” *Journal of Internal Medicine*, vol. 254, no. 3, pp. 197–215, 2003.

[71] Compiled by A. D. McNaught and A. Wilkinson., *Compendium of Chemical Terminology, 2nd ed. (the ”Gold Book”).* Oxford: Blackwell Scientific Publications, 1997.

[72] J. Kyte and R. F. Doolittle, “A simple method for displaying the hydropathic character of a protein,” *Journal of Molecular Biology*, vol. 157, no. 1, pp. 105–132, may 1982.

[73] K. Guruprasad, B. V. Reddy, and M. W. Pandit, “Correlation between stability of a protein and its dipeptide composition: A novel approach for predicting in vivo stability of a protein from its primary sequence,” *Protein Engineering, Design and Selection*, vol. 4, no. 2, pp. 155–161, 1990.

[74] P. J. Mohr and B. N. Taylor, “CODATA recommended values of the fundamental physical constants: 1998,” *Reviews of Modern Physics*, vol. 72, no. 2, pp. 351–495, apr 2000.

[75] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T. Y. Liu, “BioGPT: generative pre-trained transformer for biomedical text generation and mining,” *arXiv*, vol. preprint, p. arXiv ID:2210.10341, 2022.

[76] M. Arts, V. G. Satorras, C.-W. Huang, D. Zuegner, M. Federici, C. Clementi, F. Noé, R. Pinsler, and R. van den Berg, “Two for One: Diffusion Models and Force Fields for Coarse-Grained Molecular Dynamics,” *arXiv*, vol. preprint, p. arXiv ID:2302.00600, 2023.

[77] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola, “DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking,” *arXiv*, vol. preprint, p. arXiv ID:2210.01776, 2022.

[78] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, “The Graph Neural Network Model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, jan 2009.