AMPGAN v2: Machine Learning Guided Discovery of Anti-Microbial Peptides

Colin M. Van Oort,[†] Jonathon B. Ferrell,[‡] Jacob M. Remington,[‡] Safwan Wshah,[*,†] and Jianing Li[*,‡]

†*Department of Computer Science, University of Vermont, Burlington Vermont*
‡*Department of Chemistry, University of Vermont, Burlington Vermont*

E-mail: Safwan.Wshah@uvm.edu; Jianing.Li@uvm.edu

Abstract

Antibiotic resistance is a critical public health problem. Each year ∼2.8 million resistant infections lead to more than 35,000 deaths in the U.S. alone. Anti-microbial peptides (AMPs) show promise in treating resistant infections. But, applications of known AMPs have encountered issues in development, production, and shelf-life. To drive the development of AMP-based treatments it is necessary to create design approaches with higher precision and selectivity towards resistant targets.

In this paper we present AMPGAN v2, a generative adversarial network (GAN) based approach for rational AMP design. Like AMPGAN,[1] AMPGAN v2 combines data driven priors and controlled generation. These elements allow for the generation of AMP candidates tailored for specific applications. AMPGAN v2 is able to generate AMP candidates that are novel and diverse, making it an efficient AMP design tool.

Introduction

AMPs are a part of the natural immune response in all classes of life and are active against a broad spectrum of microbes.[2,3] Some AMPs are less likely to induce bacterial resistance, relative to traditional small molecule antibiotics.[4,5] Additionally, AMPs can have synergistic effects when used in combination with traditional antibiotics[6–8] or other AMPs.[9,10]

Over 15,000 antimicrobial peptides (AMPs) have been identified,[11] but few have been advanced to clinical trials despite their promise as treatments for antibiotic resistant pathogens. Many known AMPs have limitations that have prevented effective therapeutic application, such as relatively low half-lives,[12,13] undesirable or unknown toxicity to human cells,[14,15] and high production costs relative to traditional antibiotics.[14,16,17]

Designing AMP candidates that mitigate these short comings is a difficult problem. AMPs are made of amino acids arranged in a chain of arbitrary length, thus their chemical search space is massive. There are approximately 4.5×10^{41} unique peptides with 32 or less residues, if we consider only the 20 standard proteinogenic amino acids. Since the number of confirmed AMPs is low in comparison, less than 1.6×10^4 in DBAASP,[11] it seems that the density of AMPs in the space of all peptides is low. Efficient methods are required to effectively develop AMP-based therapeutics.

Many recent approaches to AMP discovery have relied on predictive models,[18–27] using a naive candidate generation method and a rejection sampling approach to identify promising candidates. Since the candidate generation method is naive, often many samples must be drawn before a promising peptide is found. Excessive sampling requirements inhibit the util-

ity of these models in discovery and design tasks. Reliance on engineered features constructed with domain expertise can further restrict the ability of these models to generate candidates that are qualitatively distinct from known AMPs.

In this work we present AMPGAN v2, a conditional GAN model for AMP design. We show that AMPGAN v2 can generate novel AMP candidates with similar physio-chemical properties to the training data and simultaneously incorporate designer constraints.

Related Work

Machine learning has been used to aid in the discovery and development of AMPs, with increasing adoption as both fields have matured. A large body of work has been developed based on predictive modeling of AMP properties,[18–27] and is generally labelled as quantitative structure-activity relationship (QSAR) modeling. The basic QSAR approach is to select a property of interest (e.g. antimicrobial activity), train a machine learning model to predict that property using relatively easily obtained features (e.g. primary peptide structure), then apply the trained model to unlabelled samples to estimate the property of interest. In particular, such a model can be used to identify properties of peptides present in a database that have yet to be experimentally validated.

The predictive approach can be extended to a generative one by adding an uninformed candidate generator (e.g. select a random peptide with length no more than 32). The randomly generated candidates can then be sorted and selected based on the property predicted by the QSAR models. Since the candidate generator is uninformed, it can take a massive number of samples before an appropriate candidate is identified, depending on the selection criteria. Explicitly generative models that are better informed by the data can reduce the amount of sampling required to identify promising candidates.

Proving this point, several studies have successfully applied VAEs[28] to AMP design and discovery.[29–33] If we expand our scope to the more general case of molecular design then we find several more applications of VAEs[34,35] and some of adversarial autoencoders (AAEs).[36,37]

Despite fairly broad adoption of machine learning techniques in this domain and growing interest in generative models, there is relatively little work investigating the use of GANs for AMP design and discovery. GANs have served as core components in several creative image manipulation tools,[38–40] allowing for the generation of realistic looking images that satisfy user imposed constraints. Considering the success of machine learning in QSAR modeling, better generative modeling tools could improve AMP design and discovery processes.

Generative Adversarial Networks

GANs have seen great success in learning to generate images following an explosion of research interest in 2014.[41–46] More relevant to the AMP design task, GANs have also been successful at text generation.[47–49] These results indicate the potential for application of GANs to AMP discovery.

GANs are generative models that learn to produce samples from arbitrary data distributions by pitting a pair of artificial neural networks, dubbed the generator and discriminator, against each other in a zero-sum game.[41,50,51] The generator maps samples from a latent distribution—like a multi-variate normal distribution—to samples that appear to be drawn from the real data distribution. The discriminator (or critic) is given samples drawn from an authentic data distribution or produced by the generator and must identify their source. During training the discriminator minimizes a classification error, while the generator maximizes the error of the discriminator.

GANs create samples with arbitrary features that appear to be drawn from the authentic data distribution. However, in some applications it may be useful to direct the generative process to create samples with certain features. This control can be realized through the use of Conditional GANs (CGANs),[42] where the generator and discriminator are provided an additional input, the conditioning vector, that con-

tains meta-data for the current sample. The conditioning vector is often constructed as a binary vector that indicates the presence or absence of the features of interest. For example, in an image generation context, the conditioning vector could indicate whether the generated image should contain certain objects.

During training GANs learn a semantically meaningful latent representation of the data distribution.[52,53] Exploring this latent space can provide useful information about the relationship between AMPs and accelerate the discovery of new sequences with specific properties.

Latent space interpolations are a useful method for exploring the learned latent space of a GAN. To perform a latent space interpolation, two or more points are selected from the latent space, an interpolation is constructed between those points, and the generator is applied to some subset of the interpolated points. This allows for the paths of the latent interpolation to be analyzed using tools and techniques from the target domain, in our case bio-statistical tools used to analyze peptide sequences. Latent space interpolations have been used to identify overfitting in GANs[54] and as important components of GAN-aided design tools.[38–40]

Bidirectional GANs (BiGANs) were developed to ease latent space interpolations.[55,56] The primary mechanism for this is a third network, the encoder, which maps data samples (e.g. AMP sequences) into the latent space of the generator. This allows real data samples to be projected into the latent space and serve as landmarks for mapping or exploration.

Bidirectional conditional GANs (BiCGANs) are ideal for the AMP design task since they provide designer control through the conditioning variables and fast latent space interpolations through the bidirectional component. We discuss our BiCGAN-based model, AMPGAN v2, after covering the training data and associated pre-processing in the following section.

Methods and Models

Data

We constructed our training set by combining the Database of Antimicrobial Activity and Structure of Peptides (DBAASP),[11,57] Antiviral Peptide database (AVPdb),[58] and UniProt[59] databases. We extracted the FASTA formatted sequence information, microbe targets (e.g. gram+ bacteria, gram- bacteria, viruses), mechanism targets (e.g. cell membrane, cytoplasmic protein, cell replication), and activity measures (primarily MIC50 measured in μg/ml) from each database as available. We filtered sequences that contain non-FASTA characters (e.g. tail modifications, lower case characters, etc.), and only consider sequences with a length of 32 amino acids or less for practical reasons.

After removing duplicated sequences between DBAASP and AVPdb, as well as "false negative" sequences from UniProt that also appear in DBAASP or AVPdb, we obtained 6238 sequences from DBAASP, 312 sequences from AVPdb, and 490341 sequences from UniProt. If a particular sequence has measured effectiveness against multiple microbe targets or mechanism targets, then we considered the superset of these. For sequences that have multiple activity measurements against one or more microbes, all measurements with compatible units are converted to μg/ml and the arithmetic mean was used to represent the general anti-microbial activity of the sequence.

Conditioning Data

We constructed conditioning vectors for our model using indicators for the target microbes, target mechanisms, MIC50 level, and sequence length. For the target microbes, we curated a reduced set that includes cancer, fungus, gram+ bacteria, gram- bacteria, insect, mammalian, mollicute, nematode, parasite, protista, and virus. Likewise, we consolidated the set of target mechanisms: lipid bilayer, replication, virus entry, DNA/RNA, cytoplasmic protein, assembly, virucidal, membrane protein, surface

glycoprotein, release, and unknown.

The conditioning vector is then constructed as a 64 bit binary vector. The target microbes are encoded with 11 bits indicating activity, or lack thereof, against each microbe group. Likewise, the target mechanisms are encoded with 11 bits indicating interaction with a particular cell process or element. The MIC50 values are discretized into deciles using the following bin edges: 0.0000037, 5.7557, 11, 17.9869, 27, 38.8498, 57.5996, 85.3173, 128, 232.4687, and 11240. Finally, the length of the sequence is represented using 32 digits, each indicating the presence or absence of a FASTA character. Figure 1 summarizes the contents of the conditioning vector.

We assumed that the sequences from UniProt did not have anti-microbial activity, since arbitrary peptides are unlikely to feature anti-microbial properties and we have already removed known AMPs. Thus when we constructed conditioning vectors for these sequences the only non-zero elements were the length component, which was set appropriately, and the MIC50 component, which was set to the highest bin (lowest activity).

The SI contains additional figures that show the distributions of values across the conditioning vector elements (i.e. target microbes, target mechanisms, MIC50, and sequence length).

AMPGAN v2 Design and Training

AMPGAN v2, whose architecture is shown in Figure 2, is a BiCGAN consisting of three primary components: the generator, discriminator, and encoder.

The generator architecture is composed of a dense layer that mixes the latent representation and conditioning vector, followed by a stack of exponentially dilated convolutions, and terminated by a single convolution that combines the multi-scale features extracted by the prior convolution stack. Global position information is added to the features in between the dense layer and the convolution stack to improve global sequence structure.[60] Additional details of the generator architecture can be found in Figure 3.

The discriminator architecture contains a stack of strided convolutions, followed by several dense layers. Spatial dropout is applied to each convolution and dropout is applied to each dense layer, excluding the output layer. Strided convolutions are used to quickly downsample the feature maps, while dropout increases the variance of the signal provided by the discriminator and can stabilize training.[46] Additional details of the discriminator architecture can be found in Figure 4. The AMPGAN v2 encoder shares the same architecture as the discriminator, with the only difference being a larger final layer with a linear activation function.

We trained AMPGAN v2 for 2000 epochs, where AMPGAN v2 was shown all 6550 AMP sequences along with a random sample of the 490341 Non-AMP sequences in each epoch. Training proceeded with a batch size of 128 samples, where half were drawn from the AMP set and half from the Non-AMP set. As shown in Figure 2, the binary crossentropy loss provides the training signal for the generator and discriminator, while the mean squared error is used for the encoder. The discriminator is regularized using a gradient penalty, which has been shown to improve training stability and generalization.[61,62]

AMPGAN v2 builds on the experience gained from Ferrell et al., though there are several differences in the implementation and evaluation procedure that make direct comparison of the two difficult. Full implementation details for AMPGAN v2 can be found in our GitLab repository.[63]

Results and Discussion

Training Stability

GANs can be difficult to train depending on properties of their architecture and training data. Poor training stability can involve generator mode collapse,[64–66] cyclic generator-discriminator dynamics,[61,62,64] and vanishing gradients caused by discriminator failures.[67,68]

To investigate the training stability of AMPGAN v2 we trained 30 replicates from scratch using different random initializations. We

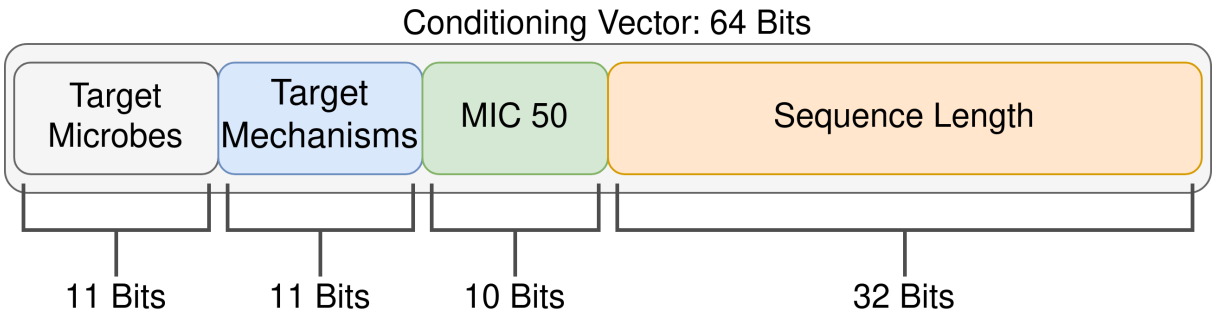

Figure 1: A visual summary of the contents and dimensions of a conditioning vector. All elements are binary encoded. For the target microbes and target mechanisms each element of the binary vector indicates activity against a particular microbe class or cellular mechanism. A one-hot encoding is used for the MIC 50 element, indicating a single MIC 50 decile. The sequence length is encoded as a bit mask, where 1 indicates the presence of a character and 0 indicates an empty slot.

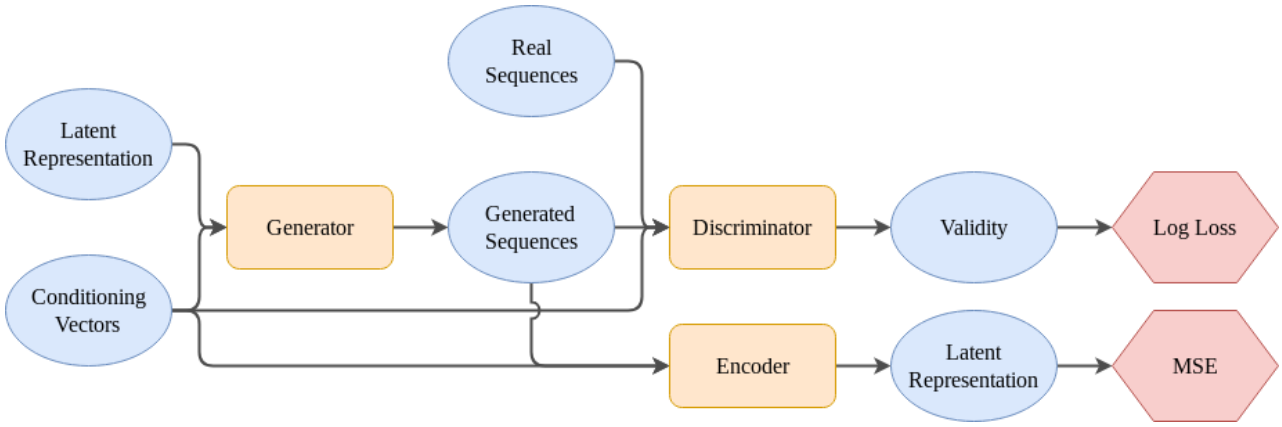

Figure 2: The macro-architecture of AMPGAN v2 consists of three networks: the generator, discriminator, and encoder. The discriminator predicts whether a sample is generated or not, and is updated using the log loss. The generator synthesizes samples, and is updated to maximize the loss of the discriminator. The encoder maps sequences into the latent space of the generator, and is trained using the mean squared error (MSE).

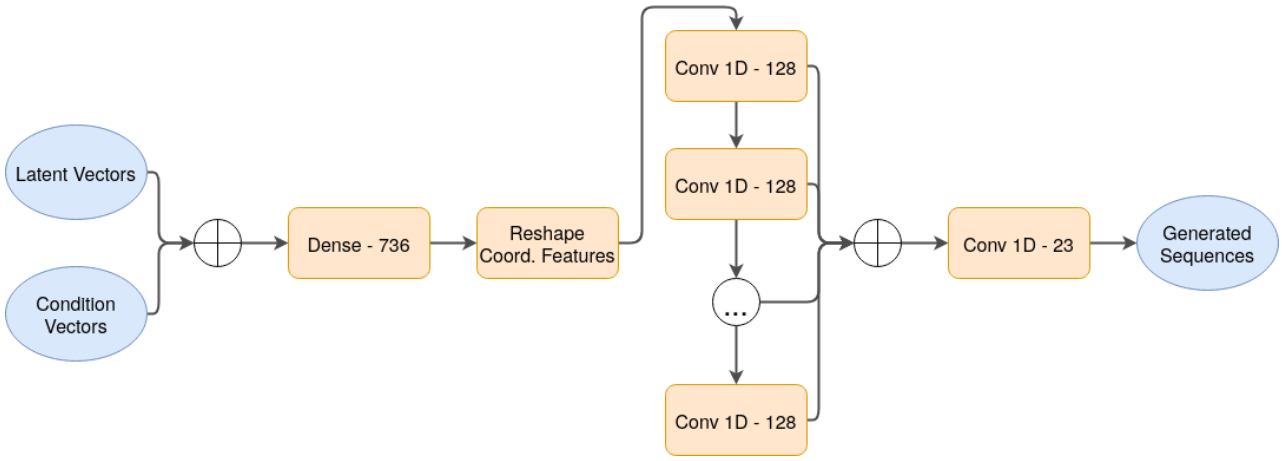

Figure 3: Details of the generator architecture used in AMPGAN v2. We use 6 convolution layers in the central stack, each with a kernel size of 3 and an exponential dilation rate. All dense and convolution layers are followed by a leaky ReLU activation, except the final convolution layer, which has a hyperbolic tangent activation. The final convolution has a kernel size of 1.

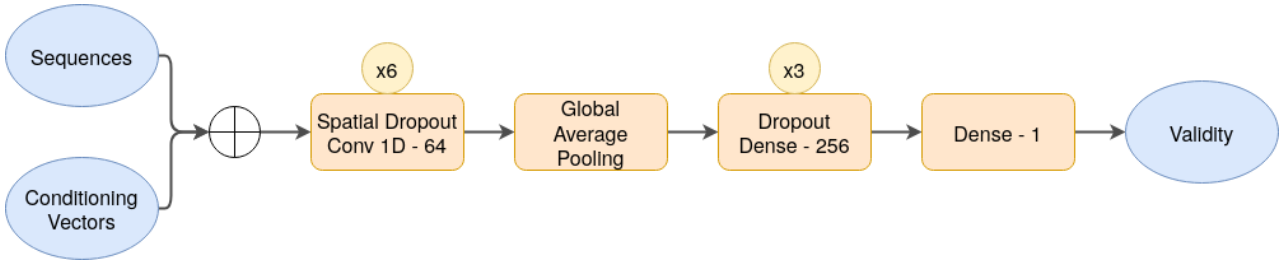

Figure 4: Details of the discriminator architecture used in AMPGAN v2. The convolutions use a filter size of 4 and a stride of 2. All applications of Dropout and Spatial Dropout use a drop rate of 25%. All dense and convolution layers are followed by a leaky ReLU activation, except the final dense layer, which has a sigmoid activation. The condition vectors are tiled and concatenated with the sequences along the features/channels dimension. The encoder uses the same architecture with a different output dimension on the final layer corresponding to the selected latent space dimension and a linear activation function.

used a heuristic criteria with two conditions to determine if a trial is successful. First, the model must generate sequences with a character-level entropy that falls between 2 and 4. This removes models that tend to generate sequences with unrealistically low or high FASTA character diversity. For reference, the average character-level entropy across our training AMPs, non-AMPs, and their combination was ∼2.6, ∼3.43, and ∼3.42 respectively. Second, the model must generate sequences whose length closely matches the value dictated by the conditioning vector. We quantified this by computing the R^2 score over batches of generated sequences, and consider values greater than 0.5 to be successful.

The results of this experiment are shown in Figure 5, where we see that three successful trials led to models with realistic sequence entropy and high correlation between the sequence length dictated by the conditioning vector and the length of the generated sequence. The other 27 trials failed to produce acceptable models, resulting in a ∼10% training success rate.

Our training success criterion requires that a successful generator account for the sequence length provided in the conditioning vector, but there is room for variation between the requirement of $R^2 = 0.5$ and the ideal value of $R^2 = 1.0$. Despite the allowed variance, all three successful trials resulted in models with high R^2 scores–specifically 0.9852, 0.9986, and 0.9975. Qualitatively, this means that almost all of the generated sequences have a sequence length that is within ±3 of the dictated sequence length.

Physio-chemical Similarity

Following the training stability experiment, we know that we can obtain models that can generate samples with realistic FASTA character diversity and are responsive to the sequence length provided in the conditioning vector. To be applicable to AMP design and discovery, we need to know more about the quality of the generator and the properties of the generated candidates. It is prohibitively expensive to experimentally validate the ability of the generator to create sequences that follow the target microbe, target mechanism, and MIC50 values provided in the conditioning vector—so instead we focus on comparisons between easily measurable physio-chemical properties of generated and authentic peptide sequences.

First we compared the distributions of amino acids present in training sequences and generated sequences, which is shown in Figure 6. From top to bottom, this figure shows comparisons between all sequences, AMP sequences, and non-AMP sequences. The left panel of each row shows the two distributions overlaid, while the right panel shows the difference between the distributions. The distributions compared in each row are remarkably similar, with the gen-

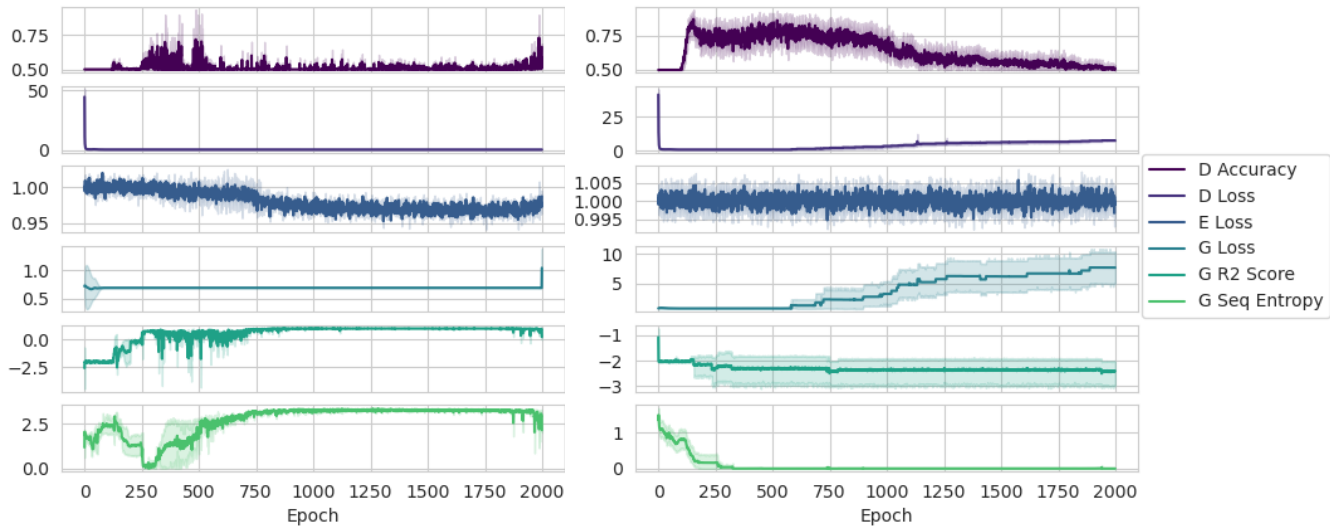

Figure 5: Investigation of training stability, summarizing the results of 30 independent trials. The left panel was constructed using the successful trials (3/30) and the right panel was constructed with the failed trials (27/30). From top to bottom the panels display the classification accuracy of the discriminator, the discriminator loss (log loss), the encoder loss (MSE), the generator loss (log loss), the R^2 score between the length dictated by the conditioning vector and generated sequences, and the average character-level entropy calculated over batches of generated sequences. This experiment highlights the relative instability of AMPGAN v2, with a success rate of ∼10%.

erated distribution falling within ±1% of the real distribution for almost all amino acids. Exceptions to this include K and R, which are more prevalent in generated sequences, along with A, I, and L, which are more prevalent in real sequences.

Figure 6 only compares real and generated distributions, however, the relationship between AMPs and non-AMPs within each group is also important. The generator may create AMP sequences that are similar to real AMP sequences and non-AMP sequences that are similar to real non-AMP sequences, but fail to adequately capture the relationship between AMP and non-AMP sequences. To investigate this we create two additional comparisons, shown in Figure 7, between AMP and non-AMP sequences in both real and generated groups. Though there are some slight deviations present in the individual distributions in the panels on the left, which were already identified in Figure 6, the differences shown in the panels on the right are nearly identical. This indicates that the generator has learned the relative relationship between AMP and non-AMP sequences, despite some slight biases in its understanding of those distributions individually. Additionally, the lower right panel of Figure 7 is directly comparable to Figure 3 from Das et al., which agrees with the signs of the relative changes shown here for all amino acids except F and G. This qualitative agreement may indicate that both models have accurately captured the qualities of the training data distribution, or at least that both models acquired a similar bias profile.

Figures 6 and 7 only investigate the appearance frequency of single amino acids, but there is a large body of research[69–73] that suggests peptides feature complex grammatical structure. We approached this using generalized word shifts,[74] which extend the simple analysis done at the character level to sub-sequences of arbitrary length. Word shifts measure the contributions of distinct sub-sequences to a divergence measure between two groups of sequences and highlight the largest contributors. In Figure 8 we provide word shifts between generated AMPs and real AMPs for sub-sequences of length 2 and 3. The largest contributors from generated peptides involve one or more instances of K or R and the largest contributors from real peptides involve L or A, which

Figure 6: Distribution of amino acids used in generated vs non-generated sequences. The rows show the distributions for all sequences (top), AMP sequences (middle), and non-AMP sequences (bottom). The columns show the distributions super-imposed (left) and the difference between the distributions (right), facilitating different comparison perspectives. The generated distribution was created using 4855 sequences with conditioning vectors drawn at random from the training set. 50% of the conditioning vectors were taken from AMP sequences and 50% from non-AMP sequences. The model used to generate these sequences was arbitrarily selected from the set of successfully trained models. The non-generated distribution was created using a sample of 5120 sequences that were randomly drawn from the training set with a 50%/50% split between AMP and non-AMP sequences. In all comparisons K is the largest outlier, appearing 4-6% more often in generated sequences than real sequences.

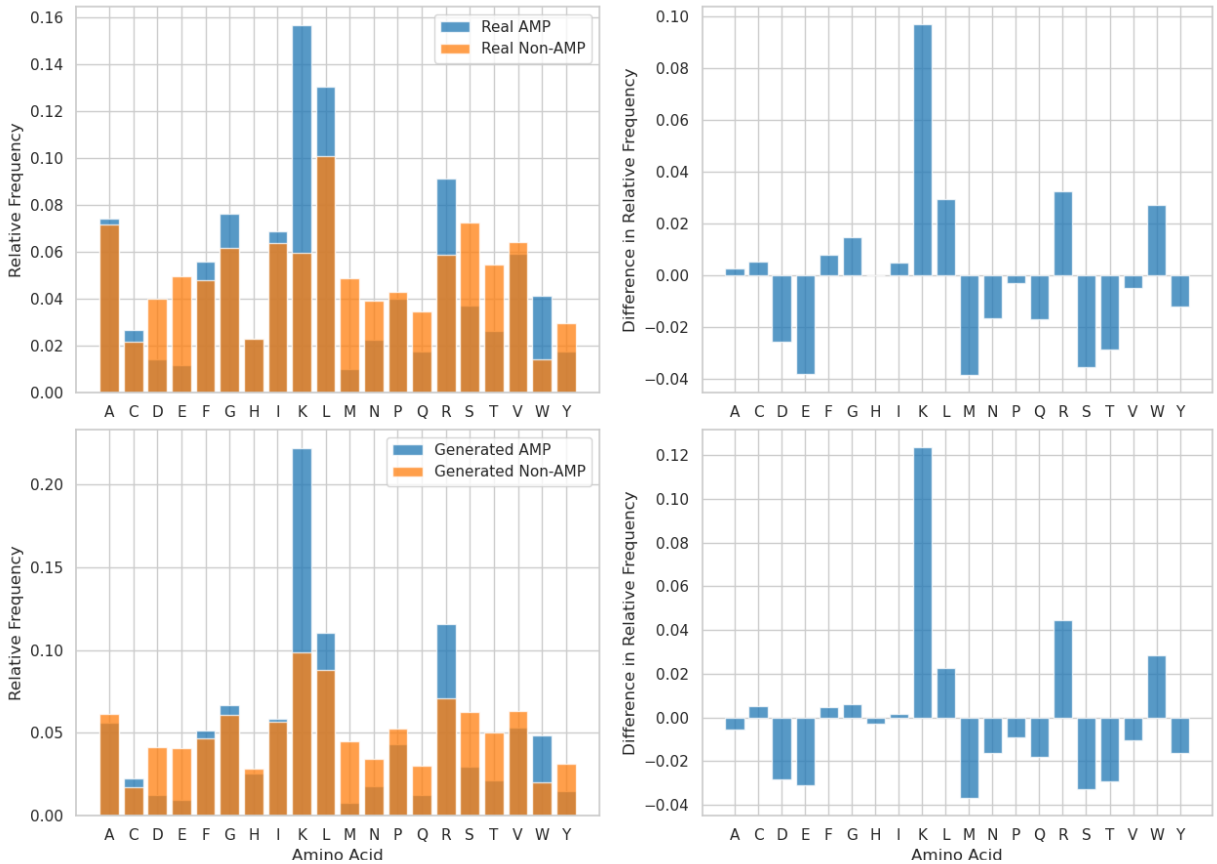

Figure 7: Amino acid usage frequency distributions for generated AMP and generated Non-AMP sequences (left) along with the difference between the two distributions (right). Comparisons are made between real (top) and generated (bottom) groups.

matches the results of the character level analysis. Many of the sub-sequences present in both plots feature positive charge or are hydrophobic, which corresponds well with known properties of a subset of AMPs. In the length 2 sub-sequence shift, the GP and PG motifs are of particular interest since they are often part of hinge-like structures near bends or kinks in proteins.

The supporting information contains a baseline analysis that compares two uniformly randomly constructed samples of sequences using the same tools, providing additional context for interpreting Figures 6 - 8.

Sequence Diversity

When proposing candidate AMPs it is important that the generated candidates are diverse as a population and novel relative to known AMPs. If the generator produces sequences with low diversity, it can run into the same sampling problems as the extended predictive models discussed earlier. A generative model will be less useful for discovering new AMPs if it does not produce sequences that are novel relative to known AMPs. We applied the Gotoh global alignment algorithm[75,76] to quantify the relative similarity of two bags of sequences. The distribution of alignment scores obtained between a pair of bags indicates the relative similarity of the bags, with more similar bags receiving higher scores.

Figure 9 contains letter-value plots[77] that summarize the scores obtained by comparing the training AMPs, generated sequences, generated AMPs, and generated non-AMPs to themselves (i.e. a measure of diversity). Additionally, the final letter-value plot shows the distribution of global scores obtained by comparing the generated and training AMP sequences.

The training AMP score distribution features much higher median and upper percentile scores than any other distribution under consideration, indicating that there is relatively low sequence diversity in the training AMP set. The median score of 16.55 and mean score of 16.49 indicate a low diversity, especially relative to the generated AMP sequences that feature a median score of 7.83 and a mean score of 7.95. The generated non-AMP sequences feature a similar level of diversity to the AMP sequences, reaching a median score of 7.8 and a mean score of 7.92. The combined set of generated sequences obtains slightly higher scores than either the AMPs or non-AMPs separately, with a median of 8.0 and a mean of 8.17, which may indicate a slight chemical overlap between the two groups or may be due to chance. Comparing the generated AMPs with the training AMPs results in the lowest scores observed, with a median of 5.24 and a mean of 5.54, indicating that the generated AMPs are novel relative to the training AMPs.

The SI contains additional information about the scoring function and baseline information that provides context for interpreting the global scores.

Conclusion

In this work we introduced AMPGAN v2, a BiCGAN that allows for the controlled generation of peptides with varying degrees of antimicrobial properties. We demonstrate that AMPGAN v2 can be trained successfully using a combination of AMP and non-AMP data. Additionally, AMPGAN v2 is able to generate sequences that are diverse, novel relative to the training data, yet still structurally similar in many ways. AMPGAN v2 is responsive to changes in the conditioning vector, allowing for effective control of the generative process.

Though AMPGAN v2 has many positive features, it also has limitations. Future work should address the low training stability of the current system. Furthermore, little validation was done to ensure that AMPGAN v2 is responsive to manipulations of the target microbe and target mechanism conditioning elements, or that the generated sequences feature those attributes. A previous version of AMPGAN v2 was able to generate novel AMPs,[1] but that still leaves the target microbe and target mechanism conditioning elements fairly untested.

AMPGAN v2 contributes a GAN-based model to an area where non-generative models

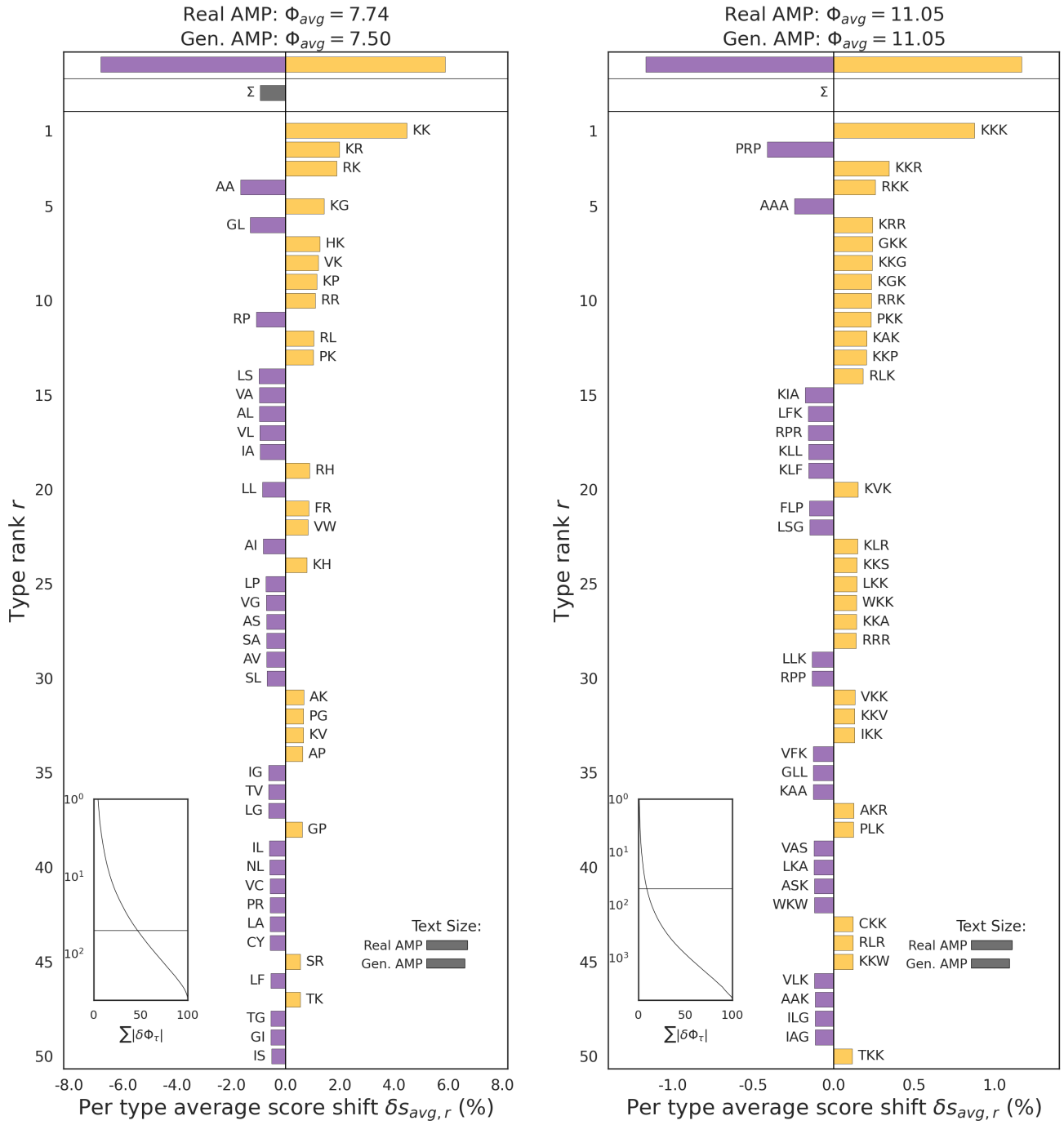

Figure 8: Shannon's entropy divergence between the distributions of length 2 (left) and length 3 (right) sub-sequences of FASTA characters in AMPs from the training set (real) or AMPs created by the generator (generated). The two values in the title of each panel indicate the average entropy of each group. For reference, the distribution of sub-sequences drawn from uniformly random sequences results in a maximum entropy of ∼8.64 for length 2 sub-sequences and ∼12.97 for length 3 sub-sequences. Both groups in both plots feature a lower entropy than the maximum, thus we should expect to see meaningful structures in each group. The CDF plot in the lower left corner of each panel indicates that the top 50 contributors to the divergence only account for ∼50% (left) and ∼10% (right) of the total divergence, thus both distributions are extremely flat.

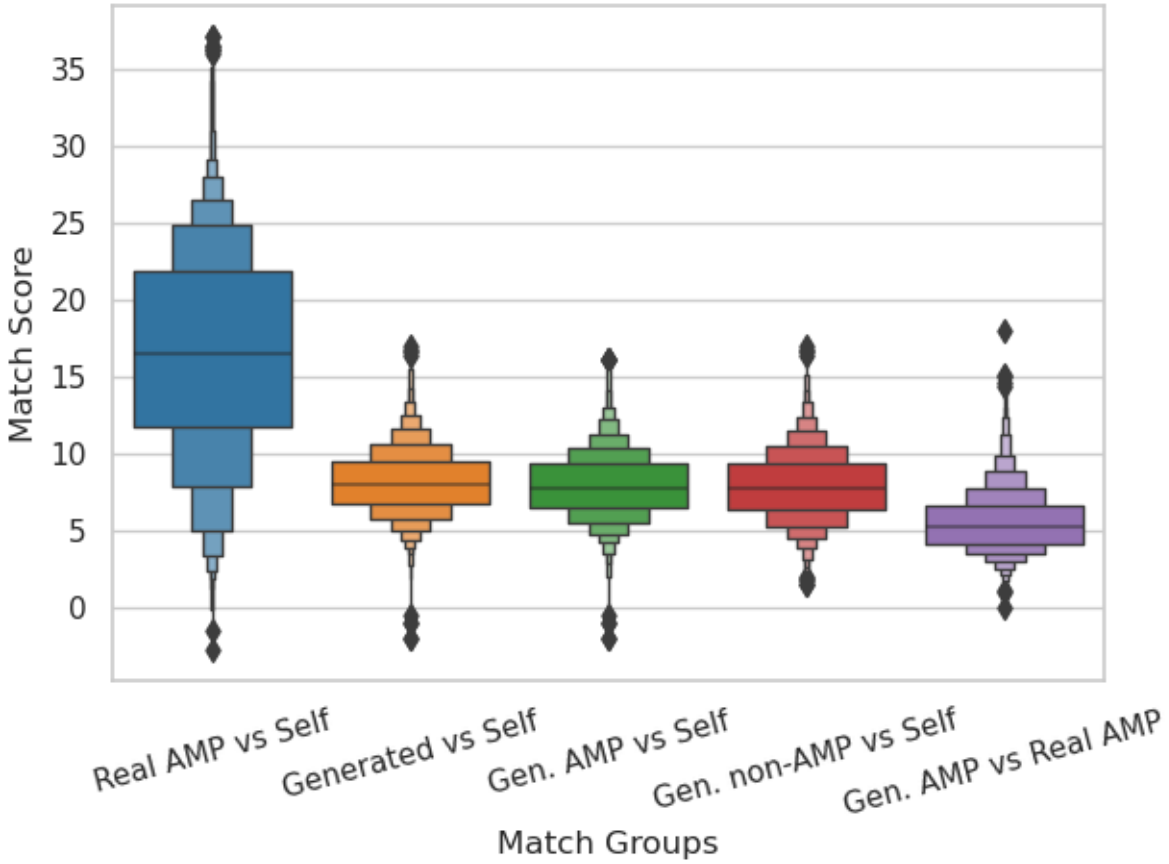

Figure 9: Letter-value plots showing distributions of match scores obtained from comparisons between different groups of sequences. The central horizontal line in each column denotes the median value. Each box extending from the median line indicates a percentile that is a half step between the starting percentile and the terminal percentile in that direction. For example, starting from the median line, the first box above is terminated at the 75th percentile, halfway between the 50th percentile and the 100th percentile. The diamonds in the tails indicate outliers, which in this case are approximately 5 to 8 of the most extreme values in each tail. The first distribution shows the match scores obtained when comparing the set of training AMPs with itself. The distribution of match scores for training AMPs has a median value that is approximately double that of the distribution for generated AMPs. This indicates that the set of generated AMPs is more diverse than the set of training AMPs. If we compare the generated AMPs directly with the training AMPs, which is shown in the final distribution, we find the lowest median match score observed so far. A low median match score here shows that the generated AMPs are novel relative to the training AMPs.

or VAEs are most prevalent. Additionally, we open source AMPGAN v2, allowing the community to interact with and deploy our tool to design and discover AMPs.

Acknowledgement The authors would like to thank the following for helpful discussion and suggestions: Thayer Alshaabi, Lapo Frati, Gabriel Meyer-Lee, Ollin Demian Langle Chimal. Computations were performed on the Vermont Advanced Computing Core supported in part by NSF award No. OAC-1827314.

References

(1) Ferrell, J. B.; Remington, J. M.; Van Oort, C. M.; Sharafi, M.; Aboushousha, R.; Janssen-Heininger, Y.; Schneebeli, S. T.; Wargo, M. J.; Wshah, S.; Li, J. A Generative Approach toward Precision Antimicrobial Peptide Design. *BioRxiv* **2020**,

(2) Reddy, K.; Yedery, R.; Aranha, C. Antimicrobial peptides: premises and promises. *International Journal of Antimicrobial Agents* **2004**, *24*, 536–547.

(3) Ageitos, J.; Sánchez-Pérez, A.; Calo-Mata, P.; Villa, T. Antimicrobial peptides (AMPs): Ancient compounds that represent novel weapons in the fight against bacteria. *Biochemical Pharmacology* **2017**, *133*, 117–138.

(4) Lázár, V.; Martins, A.; Spohn, R.; Daruka, L.; Grézal, G.; Fekete, G.; Számel, M.; Jangir, P. K.; Kintses, B.; Csörgő, B., et al. Antibiotic-resistant bacteria show widespread collateral sensitivity to antimicrobial peptides. *Nature Microbiology* **2018**, *3*, 718.

(5) Spohn, R.; Daruka, L.; Lázár, V.; Martins, A.; Vidovics, F.; Grézal, G.; Méhi, O.; Kintses, B.; Számel, M.; Jangir, P. K., et al. Integrated evolutionary analysis reveals antimicrobial peptides with limited resistance. *Nature Communications* **2019**, *10*, 1–13.

(6) Hollmann, A.; Martinez, M.; Maturana, P.; Semorile, L. C.; Maffia, P. C. Antimicrobial peptides: interaction with model and biological membranes and synergism with chemical antibiotics. *Frontiers in chemistry* **2018**, *6*, 204.

(7) Ruden, S.; Rieder, A.; Schwartz, T.; Mikut, R.; Hilpert, K.; Chis Ster, I. Synergy pattern of short cationic antimicrobial peptides against multidrug-resistant Pseudomonas aeruginosa. *Frontiers in Microbiology* **2019**, *10*, 2740.

(8) Zharkova, M. S.; Orlov, D. S.; Golubeva, O. Y.; Chakchir, O. B.; Eliseev, I. E.; Grinchuk, T. M.; Shamova, O. V. Application of antimicrobial peptides of the innate immune system in combination with conventional antibiotics–a novel way to combat antibiotic resistance? *Frontiers in Cellular and Infection Microbiology* **2019**, *9*, 128.

(9) Yan, H.; Hancock, R. E. Synergistic interactions between mammalian antimicrobial defense peptides. *Antimicrobial Agents and Chemotherapy* **2001**, *45*, 1558–1560.

(10) Yu, G.; Baeder, D. Y.; Regoes, R. R.; Rolff, J. Combination effects of antimicrobial peptides. *Antimicrobial Agents and Chemotherapy* **2016**, *60*, 1717–1724.

(11) Pirtskhalava, M.; Gabrielian, A.; Cruz, P.; Griggs, H. L.; Squires, R. B.; Hurt, D. E.; Grigolava, M.; Chubinidze, M.; Gogoladze, G.; Vishnepolsky, B., et al. DBAASP v. 2: an enhanced database of structure and antimicrobial/cytotoxic activity of natural and synthetic peptides. *Nucleic Acids Research* **2016**, *44*, D1104–D1112.

(12) Mathur, D.; Singh, S.; Mehta, A.; Agrawal, P.; Raghava, G. P. In silico approaches for predicting the half-life of natural and modified peptides in blood. *PLOS ONE* **2018**, *13*, e0196829.

(13) Gomes, B.; Augusto, M. T.; Felício, M. R.; Hollmann, A.; Franco, O. L.;

Gonçalves, S.; Santos, N. C. Designing improved active peptides for therapeutic approaches against infectious diseases. *Biotechnology Advances* **2018**, *36*, 415–429.

(14) Bradshaw, J. P. Cationic antimicrobial peptides. *BioDrugs* **2003**, *17*, 233–240.

(15) Marr, A. K.; Gooderham, W. J.; Hancock, R. E. Antibacterial peptides for therapeutic use: obstacles and realistic outlook. *Current Opinion in Pharmacology* **2006**, *6*, 468–472.

(16) Hilpert, K.; Volkmer-Engert, R.; Walter, T.; Hancock, R. E. High-throughput generation of small antibacterial peptides with improved activity. *Nature Biotechnology* **2005**, *23*, 1008–1012.

(17) Mahlapuu, M.; Håkansson, J.; Ringstad, L.; Björn, C. Antimicrobial peptides: an emerging category of therapeutic agents. *Frontiers in cellular and infection microbiology* **2016**, *6*, 194.

(18) Kumar, M.; Verma, R.; Raghava, G. P. Prediction of mitochondrial proteins using support vector machine and hidden Markov model. *Journal of Biological Chemistry* **2006**, *281*, 5357–5363.

(19) Fjell, C. D.; Hancock, R. E.; Cherkasov, A. AMPer: a database and an automated discovery tool for antimicrobial peptides. *Bioinformatics* **2007**, *23*, 1148–1155.

(20) Thomas, S.; Karnik, S.; Barai, R. S.; Jayaraman, V. K.; Idicula-Thomas, S. CAMP: a useful resource for research on antimicrobial peptides. *Nucleic Acids Research* **2009**, *38*, D774–D780.

(21) Mooney, C.; Wang, Y.-H.; Pollastri, G. SCLpred: protein subcellular localization prediction by N-to-1 neural networks. *Bioinformatics* **2011**, *27*, 2812–2819.

(22) Mooney, C.; Haslam, N. J.; Pollastri, G.; Shields, D. C. Towards the improved discovery and design of functional peptides: common features of diverse classes permit generalized prediction of bioactivity. *PLOS ONE* **2012**, *7*, e45012.

(23) Mooney, C.; Haslam, N. J.; Holton, T. A.; Pollastri, G.; Shields, D. C. PeptideLocator: prediction of bioactive peptides in protein sequences. *Bioinformatics* **2013**, *29*, 1120–1126.

(24) Xiao, X.; Wang, P.; Lin, W.-Z.; Jia, J.-H.; Chou, K.-C. iAMP-2L: a two-level multi-label classifier for identifying antimicrobial peptides and their functional types. *Analytical Biochemistry* **2013**, *436*, 168–177.

(25) Agrawal, P.; Raghava, G. P. Prediction of antimicrobial potential of a chemically modified peptide from its tertiary structure. *Frontiers in Microbiology* **2018**, *9*, 2551.

(26) Barrett, R.; Jiang, S.; White, A. D. Classifying antimicrobial and multifunctional peptides with Bayesian network models. *Peptide Science* **2018**, *110*, e24079.

(27) Witten, J.; Witten, Z. Deep learning regression model for antimicrobial peptide design. *BioRxiv* **2019**, 692681.

(28) Kingma, D. P.; Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* **2013**,

(29) Das, P.; Wadhawan, K.; Chang, O.; Sercu, T.; Santos, C. D.; Riemer, M.; Chenthamarakshan, V.; Padhi, I.; Mojsilovic, A. Pepcvae: Semi-supervised targeted design of antimicrobial peptide sequences. *arXiv preprint arXiv:1810.07743* **2018**,

(30) Das, P.; Sercu, T.; Wadhawan, K.; Padhi, I.; Gehrmann, S.; Cipcigan, F.; Chenthamarakshan, V.; Strobelt, H.; Santos, C. d.; Chen, P.-Y., et al. Accelerating Antimicrobial Discovery with

Controllable Deep Generative Models and Molecular Dynamics. *arXiv preprint arXiv:2005.11248* **2020**,

(31) Sercu, T.; Gehrmann, S.; Strobelt, H.; Das, P.; Padhi, I.; Dos Santos, C.; Wadhawan, K.; Chenthamarakshan, V. Interactive Visual Exploration of Latent Space (IVELS) for peptide auto-encoder model selection. **2019**,

(32) Hawkins-Hooker, A.; Depardieu, F.; Baur, S.; Couairon, G.; Chen, A.; Bikard, D. Generating functional protein variants with variational autoencoders. *BioRxiv* **2020**,

(33) Chen, S.; Kim, H. U. Designing Novel Functional Peptides by Manipulating a Temperature in the Softmax Function Coupled with Variational Autoencoder. 2019; pp 6010–6012.

(34) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science* **2018**, *4*, 268–276.

(35) Jin, W.; Barzilay, R.; Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364* **2018**,

(36) Kadurin, A.; Nikolenko, S.; Khrabrov, K.; Aliper, A.; Zhavoronkov, A. druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics* **2017**, *14*, 3098–3104.

(37) Blaschke, T.; Olivecrona, M.; Engkvist, O.; Bajorath, J.; Chen, H. Application of generative autoencoder in de novo molecular design. *Molecular Informatics* **2018**, *37*, 1700123.

(38) Brock, A.; Lim, T.; Ritchie, J. M.; Weston, N. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093* **2016**,

(39) Zhu, J.-Y.; Krähenbühl, P.; Shechtman, E.; Efros, A. A. Generative visual manipulation on the natural image manifold. 2016; pp 597–613.

(40) Perarnau, G.; Van De Weijer, J.; Raducanu, B.; Álvarez, J. M. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355* **2016**,

(41) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. Advances in neural information processing systems. 2014; pp 2672–2680.

(42) Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* **2014**,

(43) Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. 2015.

(44) Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. Advances in neural information processing systems. 2016; pp 2172–2180.

(45) Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **2015**, *115*, 211–252.

(46) Brock, A.; Donahue, J.; Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* **2018**,

(47) Hu, Z.; Yang, Z.; Liang, X.; Salakhutdinov, R.; Xing, E. P. Toward controlled generation of text. 2017; pp 1587–1596.

(48) Fedus, W.; Goodfellow, I.; Dai, A. M. MaskGAN: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736* **2018**,

(49) Chen, J.; Wu, Y.; Jia, C.; Zheng, H.; Huang, G. Customizable Text Generation via Conditional Text Generative Adversarial Network. *Neurocomputing* **2019**,

(50) Schmidhuber, J. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. *Institut für Informatik, Technische Universität München. Technical Report FKI-126* **1990**, *90*.

(51) Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. 1991; pp 222–227.

(52) Bojanowski, P.; Joulin, A.; Lopez-Paz, D.; Szlam, A. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776* **2017**,

(53) Shen, Y.; Gu, J.; Tang, X.; Zhou, B. Interpreting the latent space of gans for semantic face editing. *arXiv preprint arXiv:1907.10786* **2019**,

(54) Odena, A.; Olah, C.; Shlens, J. Conditional image synthesis with auxiliary classifier gans. 2017; pp 2642–2651.

(55) Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; Courville, A. Adversarially learned inference. *arXiv preprint arXiv:1606.00704* **2016**,

(56) Donahue, J.; Krähenbühl, P.; Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782* **2016**,

(57) Gogoladze, G.; Grigolava, M.; Vishnepolsky, B.; Chubinidze, M.; Duroux, P.; Lefranc, M.-P.; Pirtskhalava, M. DBAASP: database of antimicrobial activity and structure of peptides. *FEMS microbiology letters* **2014**, *357*, 63–68.

(58) Qureshi, A.; Thakur, N.; Tandon, H.; Kumar, M. AVPdb: a database of experimentally validated antiviral peptides targeting medically important viruses. *Nucleic Acids Research* **2014**, *42*, D1147–D1153.

(59) Consortium, U. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research* **2019**, *47*, D506–D515.

(60) Liu, R.; Lehman, J.; Molino, P.; Such, F. P.; Frank, E.; Sergeev, A.; Yosinski, J. An intriguing failing of convolutional neural networks and the coordconv solution. Advances in Neural Information Processing Systems. 2018; pp 9605–9616.

(61) Roth, K.; Lucchi, A.; Nowozin, S.; Hofmann, T. Stabilizing training of generative adversarial networks through regularization. Advances in neural information processing systems. 2017; pp 2018–2028.

(62) Mescheder, L.; Geiger, A.; Nowozin, S. Which training methods for GANs do actually converge? *arXiv preprint arXiv:1801.04406* **2018**,

(63) Van Oort, C. M. AMP-GAN. https://gitlab.com/vail-uvm/amp-gan, Accessed 2020/08/30.

(64) Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. Advances in neural information processing systems. 2016; pp 2234–2242.

(65) Che, T.; Li, Y.; Jacob, A. P.; Bengio, Y.; Li, W. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136* **2016**,

(66) Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M. U.; Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. Advances in Neural Information Processing Systems. 2017; pp 3308–3318.

(67) Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. International conference on machine learning. 2017; pp 214–223.

(68) Nagarajan, V.; Kolter, J. Z. Gradient descent GAN optimization is locally stable. Advances in neural information processing systems. 2017; pp 5585–5595.

(69) Brendel, V.; Busse, H. Genome structure described by formal languages. *Nucleic Acids Research* **1984**, *12*, 2561–2568.

(70) Loose, C.; Jensen, K.; Rigoutsos, I.; Stephanopoulos, G. A linguistic model for the rational design of antimicrobial peptides. *Nature* **2006**, *443*, 867–869.

(71) Gimona, M. Protein linguistics—a grammar for modular protein assembly? *Nature Reviews Molecular Cell Biology* **2006**, *7*, 68–73.

(72) Wang, Y.; You, Z.-H.; Yang, S.; Li, X.; Jiang, T.-H.; Zhou, X. A high efficient biological language model for predicting protein–protein interactions. *Cells* **2019**, *8*, 122.

(73) Rao, R.; Bhattacharya, N.; Thomas, N.; Duan, Y.; Chen, P.; Canny, J.; Abbeel, P.; Song, Y. Evaluating protein transfer learning with TAPE. Advances in Neural Information Processing Systems. 2019; pp 9689–9701.

(74) Gallagher, R. J.; Frank, M. R.; Mitchell, L.; Schwartz, A. J.; Reagan, A. J.; Danforth, C. M.; Dodds, P. S. Generalized Word Shift Graphs: A Method for Visualizing and Explaining Pairwise Comparisons Between Texts. *arXiv preprint arXiv:2008.02250* **2020**,

(75) Gotoh, O. An improved algorithm for matching biological sequences. *Journal of molecular biology* **1982**, *162*, 705–708.

(76) Cock, P. J.; Antao, T.; Chang, J. T.; Chapman, B. A.; Cox, C. J.; Dalke, A.; Friedberg, I.; Hamelryck, T.; Kauff, F.; Wilczynski, B., et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **2009**, *25*, 1422–1423.

(77) Hofmann, H.; Kafadar, K.; Wickham, H. Letter-value plots: Boxplots for large data. *The American Statistican* **2011**,

Supporting Information Available

Conditioning Information Distributions

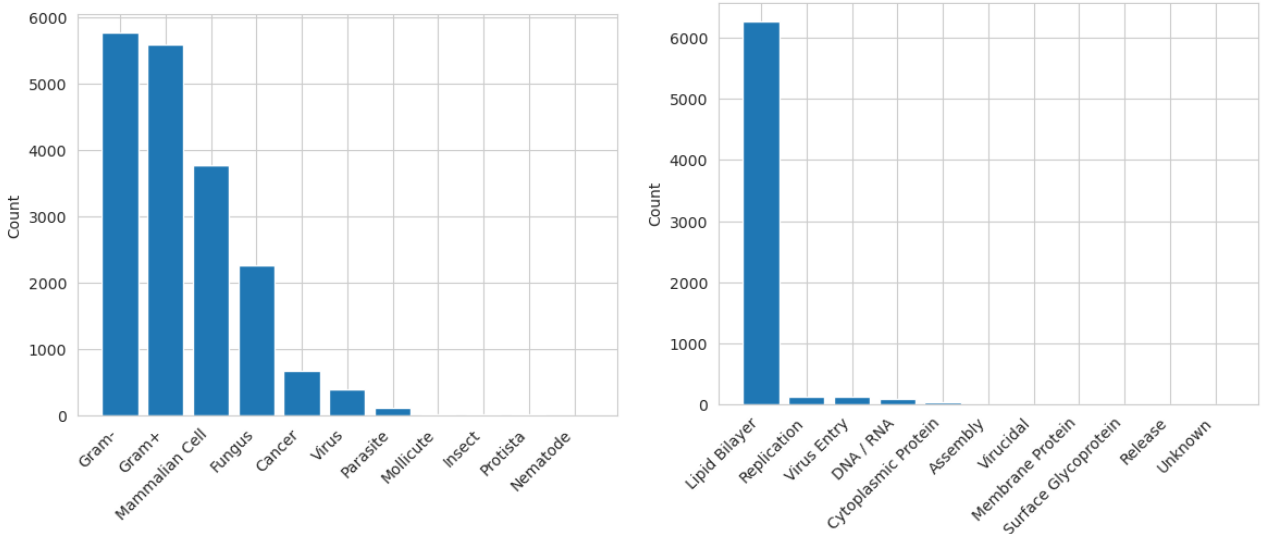

Figure 10: Label frequency for the target microbe (Left) and target mechanism (Right) conditioning variables.

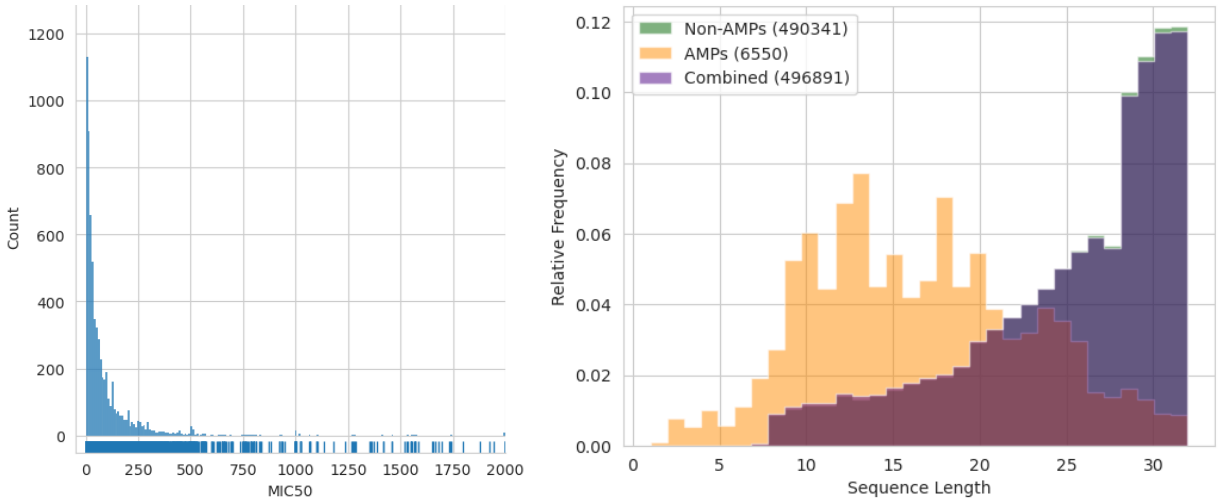

Figure 11: The distribution of MIC50 values before discretization (Left) and peptide sequence lengths (Right). 27 samples with MIC50 values greater than 2000 were truncated to ease inspection of the rest of the distribution.

Sequence Length Correlation

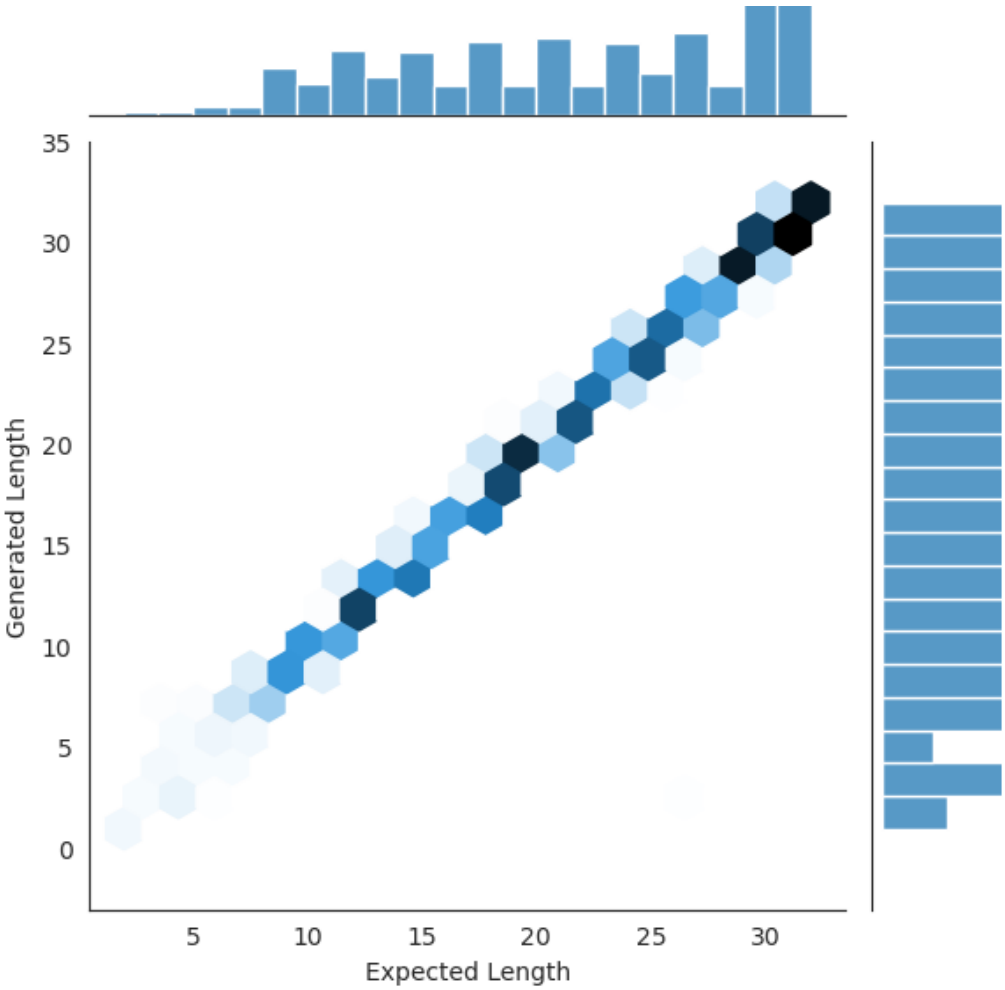

Figure 12: Agreement between the sequence length dictated by the conditioning vector and the length of sequences produced by the generator. This figure was created using 4855 sequences that were generated using conditioning vectors drawn at random from the training set. 50% of the conditioning vectors were taken from AMP sequences and 50% from non-AMP sequences. The model used to generate these sequences was arbitrarily selected from the set of successfully trained models. The generator pays close attention to the sequence length conditioning variable, resulting in an R^2 score of 0.9798.

Sequence Analysis Random Baselines

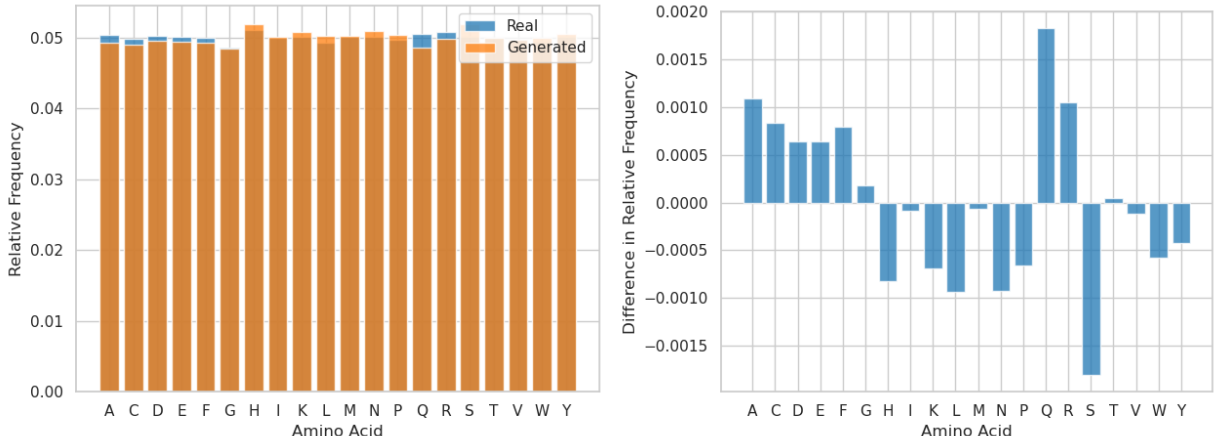

Figure 13: Amino acid frequency distribution comparison between two independent groups of 5000 uniformly randomly constructed sequences with a maximum length of 32. The distributions are flat, excluding a small amount of sampling noise. Additionally, the deviation between the two is extremely small, with the largest difference value being several orders of magnitude smaller than the largest value present in Figures 6 or 7.

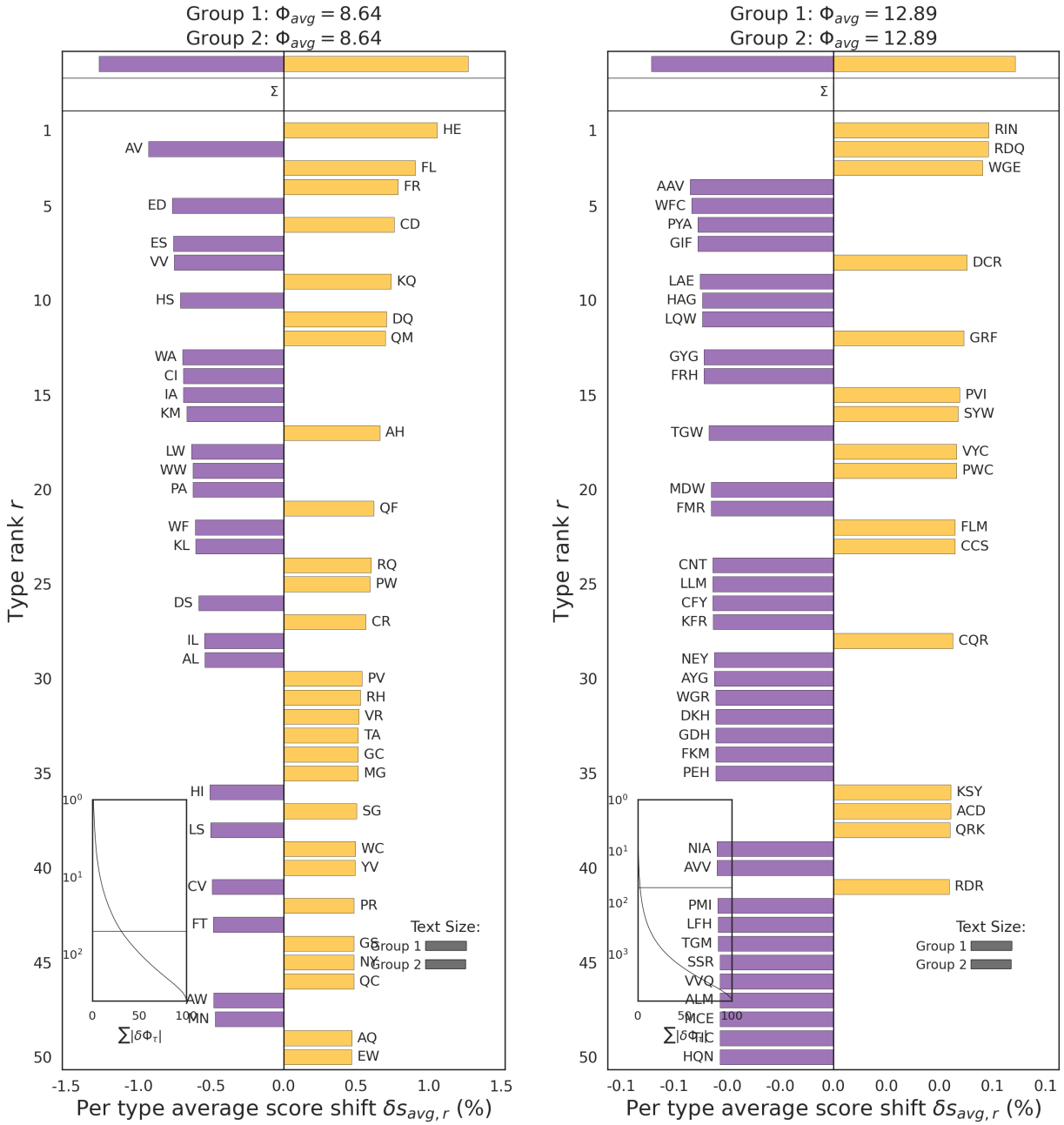

Figure 14: Word shift plots comparing two independent groups of 5000 uniformly randomly constructed sequences with a maximum length of 32. Similar to the character level analysis shown in Figure 13, these word shifts are extremely flat. However, since the number of distinct elements grows exponentially with the sub-sequence length, sampling error may have a larger impact here. The maximum entropy for length 2 sub-sequences constructed from the 20 common amino acids is ∼8.64, which is reliably obtained by a sample of this size. The maximum entropy for length 3 sub-sequences is ∼12.97, but is not reached due to sampling error. Approximately 1 to 5 length 3 sub-sequences are unobserved in a sample of this size. There are 400 unique length 2 and 8000 unique length 3 sub-sequences, thus a uniform distribution over those sets has an element-wise probability of 0.0025 and 0.000125 respectively.

Global Sequence Alignment Scores

To investigate the similarity of two bags of sequences we applied the Gotoh global alignment algorithm.[75] We use the implementation provided by Biopython's `PairwiseAligner` object,[76] configured with the `BLOSUM62` substitution matrix, an open gap score of -10, and an extend gap score of -1.

Interpreting global alignment scores can be difficult, so we performed a Monte Carlo experiment to uncover information about the distribution of scores in particular circumstances. Specifically, we construct two bags of random sequences, called S_1 and S_2, containing N and $N/2$ sequences respectively. These sequences have a uniformly random length selected from 1 to 32, and the elements of each sequence are uniformly randomly selected from the 20 common amino acids. Next, we construct a new bag, S_3, by combining S_2 with a duplicate. Thus, S_1 and S_3 both contain N sequences, where all sequences in S_1 are likely to be unique and S_3 contains two copies of every unique sequence in S_2. Next, we construct S_4 by randomly mixing the sequences of S_1 and S_3 using a control parameter $m \in [0, 1]$. This mixing is implemented by iterating pairwise over the sequences of S_1 and S_3, then iterating pairwise over the FASTA characters of those sequences. The characters from S_1 are selected with probability $1 - m$ and the characters from S_3 are selected with probability m. From this construction, the mixing parameter directly controls the diversity of S_4, providing a stochastic interpolation between relatively high and low diversity bags of sequences. Finally, pairwise scoring is computed between S_4 and itself using the system described above.

The full experiment then involved sampling the mixture parameter at 50 evenly spaced points that span the interval $[0, 1]$ and executing 30 replicates of the scoring procedure at each. All replicates use $N = 1000$. Figure 15 summarizes this experiment, where the orange line indicates the mean match score across the replicates and the shaded area covers plus or minus one standard deviation. Percentile information is shown by the blue lines, and extreme values are shown with grey markers. This indicates a roughly exponential scaling in the expected match score as the mixing parameter varies from 0 to 1, ranging from $\sim$2.5 to $\sim$20. The standard deviation indicates the existence of heteroscedasticity, where the tails of the score distribution spread out as the mixture parameter increases in value.

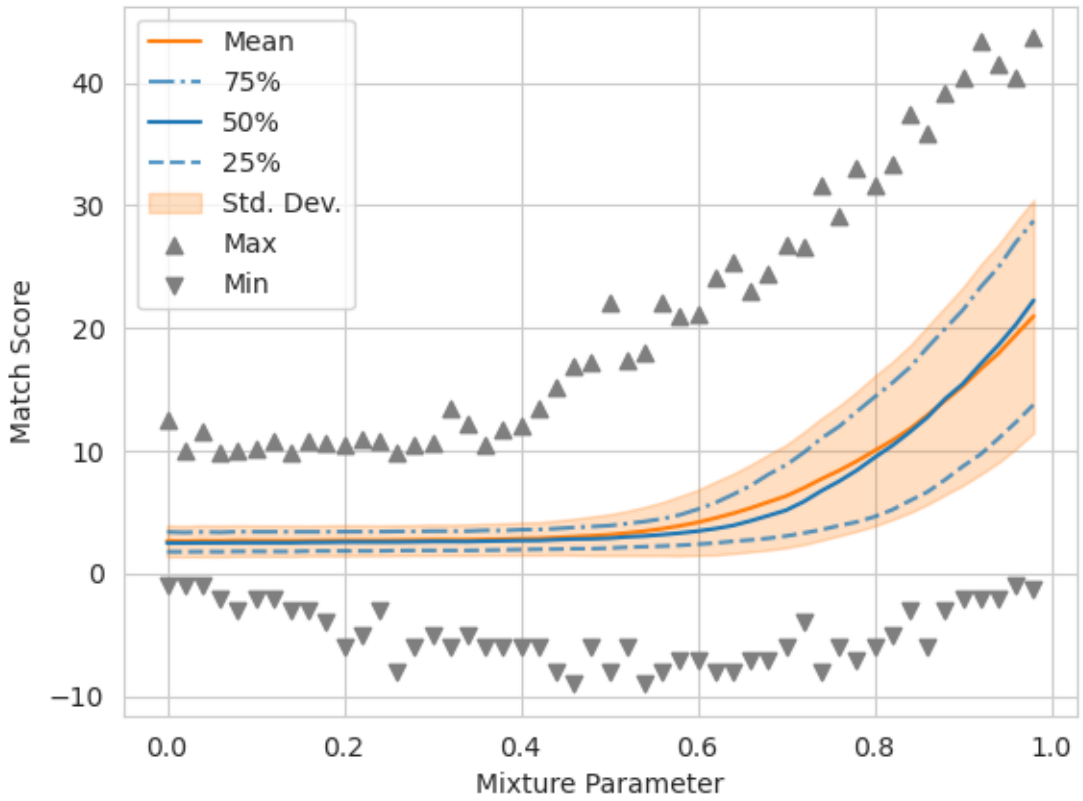

Figure 15: Distributions of global match scores between a bag of FASTA sequences and itself. The dark line indicates the mean match score and the shaded area indicates plus or minus one standard deviation. The horizontal axis corresponds with a mixture parameter that controls the level of diversity in the bag. For low values the bag of sequences is composed entirely of unique sequences, resulting in low match scores on average. As the value of the mixture parameter increases the level of diversity in the bag decreases. When the mixture parameter reaches a value of 1.0 the bag contains an exact duplicate for every sequence, resulting in match scores in the 30s.