Small Protein Design based on LSTM RNN

Haoyu Hua, Tianshu Wang, Yuxin Wang, Ridi Wen, Chongzhou Yang

Xiamen University

{23020211153893, 31520211154086, 23020211153971, 23020211153973, 31520211154103}@stu.xmu.edu.cn

Abstract

The three-dimensional structure, physicochemical properties and molecular functions of a protein are all determined by its amino acid sequence. A protein with a length of 200 amino acids has a possible sequence of 20^{200}, causing an overflow error in the computer. The current biological directed evolution technology can only involve a small part of the search space, while the deep learning method can greatly increase the design space, so as to design the required protein. In this project, we hope to use the RNN-based model to synthesize proteins with a given similar sequence, that is, to generate new protein sequences through a series of protein sequences with similar functions.

Introduction

Proteins are made from twenty-plus basic building blocks called amino acids and mediates the fundamental processes of life, having been the focus of much biomedical research for the past 50 years (Huang, Boyken, and Baker 2016). De novo protein design holds promise for creating small stable proteins with shapes and have the potential to solve a vast array of technical challenges in biomedicine and biological engineering. Although we can generate new proteins from scratch based on the principles of protein biophysics, there are still two challenges: one is that the system energy cannot be accurately calculated; the other is that the protein structure and sequence space is very large, which makes sampling difficult. Therefore, it is necessary for us to introduce artificial intelligence methods for protein design to explore the entire sequence space.

With the rapid development of deep learning, people have successively applied deep learning to design small protein molecules. (Colton, De Mántaras, and Stock 2009; Liu et al. 2016; White and Wilson 2010) The goal is to identify whether new examples (*e.g.*, small protein molecules) have realistic properties (*e.g.*, biological activity). General classifiers or classification models are classified for a given unlabeled domain instance, and one of their drawbacks is their inability to capture infinite or exponentially large combinatorial search space, which can be done by models generated from deep learning. Therefore, (Schneider and Schneider 2016; Miyao and Funatsu 2017) this feature of those models generated from deep learning can have potential relevance for the de novo design of small protein molecules and reverse QSAR modeling. So there is currently a great interest in how to automatically generate the required drug molecules (*e.g.*, small protein molecules) using deep learning methods.

LSTM (Long Short-Term Memory) is a long and short-term memory network, which belongs to temporal recurrent neural network (RNNs). (Hochreiter and Schmidhuber 1997) proposed that the LSTM system could solve the problem of gradient disappearance and gradient explosion problems during long sequence training to some extent. Briefly, the LSTM with memory cells, (Sutskever, Martens, and Hinton 2011; Segler et al. 2018), performs better in longer sequences than the ordinary RNNs.

We hypothesize that the syntax of sequence amino acid representation of proteins and peptides can be learned by the cyclic LSTM network to enable the learned model to generate de novo sequence designs. There are two main applications of RNNs, one is used to model the representation sentence and get a complete sentence representation; the other is used to represent the current context of a sentence. In a certain word, the hidden layer state of the RNNs can be expressed as a sentence context representation from the beginning to the word (Qian and Sejnowski 1988). The RNNs learn the most appropriate internal representation of a given problem directly from the amino acid sequence, so they no longer strictly require explicit feature selection in meaningful molecular descriptors when we have a sufficiently diverse and representative training pool.

Currently, one of the systems successfully applied for protein sequence analysis is recurrent LSTM systems, such as recurrent neural networks with or without LSTM memory units (Heffernan et al. 2017; Baldi and Pollastri 2003; Sønderby and Winther 2014). They can predict the protein secondary structure, and (Hochreiter, Heusel, and Obermayer 2007) detect protein homology. These systems are classifiers, which are not used for de novo sequence generation.

In practice problems, we often encounter the problem of trying to design proteins with different structures but similar functions. The goal is to design the most stable proteins with specific functions, or proteins with the least toxic side effects among antibodies and antimicrobial peptides. This

project attempts to explore how to design similar proteins based on protein sequences that have the same function, so as to achieve amplification of the protein database.

Therefore, we mainly use LSTM RNNs to design amino acid sequences, and chose antimicrobial peptides (AMPs) to train the model. Once trained, we use this system to generate new, potentially amphipathic AMPs and allow RNNs to capture the amphipathicity (Fjell et al. 2012). Amphipathicity is a related property of the antimicrobial activity of AMPs. On the other hand, we train the network by using small protein sequences with specific three-dimensional structures. After the model training, we use the model to produce new small protein sequences with the same 3D structure.

Related Work

In recent years, sequence-based protein design is a research hotspot. (Hawkins-Hooker et al. 2021) proposed a model based on autoencoder and used VAE for sequence generation. (Repecka et al. 2021) proposed a sequence generation model based on GAN to generate apple dehydrogenase. (Wu et al. 2020) proposed the use of the Transformer encoder-decoder model to generate signal peptides for industrial enzymes. (Angermueller et al. 2019) used reinforcement learning (RL) to construct state transition functions and reward mechanisms to generate protein sequences. (Ingraham et al. 2019) used a graph-based model based on structure to devo design protein sequences. (Jing et al. 2020) introduce geometric vector perceptrons, which extend standard dense layers to operate on collections of Euclidean vectors into the graph-based model. (Cao et al. 2021) create a novel transformer-based generative framework for designing protein sequences conditioned on a specific target fold. (Qi and Zhang 2020) developed a deep neural network named DenseCPD that considers the three-dimensional density distribution of protein backbone atoms and predicts the probability of 20 natural amino acids for each residue in a protein. Although these complex models can achieve excellent results on typical natural language sequence processing tasks, the results are not optimistic in the field of protein design where the data set is not rich enough. In view of this, we narrowed the scope of the model to only target small proteins. Design, and try to use a simple RNN neural network to explore this problem.

Proposed Solution

Training Data

We studied many data sets, among them, in order to train CPD, we used CATH4.2, and finally we selected our database. We have collected a large number of peptide sequences, they are from three publicly accessible databases. At the same time, we selected the complete DADP peptide database. Most AMPs that do not contain Cys amino acids (C) that may form disulfide bonds form linear helical structures. In order to facilitate future synthesis, we removed the Cys-containing sequence and incorporated the unnatural amino acid sequence. Our final training data set (1554 peptides) contains 7-48 amino acids, with an average length of 20.8 ± 7.7 (mean ± SD), median = 21 residues

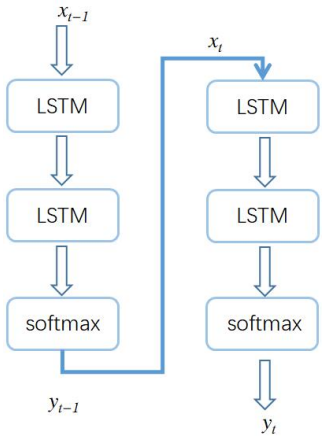

Figure 1: Structure of net.

Model Structure and Training

We trained a two-layer unidirectional LSTM RNN with 256 memory units in each layer. The output of the second LSTM is sent to the feed forward layer, which has 22 output neurons and combines the output signal with the softmax function. We add a constant deviation to the LSTM forget gate, and make the LSTM layer perform 20% and 40% dropout regularization on the first and second layers, respectively, to reduce the possibility of overfitting. We used the Adam optimizer with a learning rate of 1%. Perform one-hot encoding for each residue in the K amino acid symbol sequence, and calculate the cross-entropy loss L of the classification between the output of the network and the actual expectation as

$$L = -\sum_{i=1}^{K} y_i \log(y_i) \tag{1}$$

where y_i represents the i-th one-hot residue vector calculated in the training data. In order to obtain when the hyperparameters, we cross-validated different models five times, each time more than 200 epochs.

The number of one-layer and two-layer shield units of LSTM is selected from [24, 32, 48, 64, 128, 256, 512], and the regularization uses a ratio of 0.1 or 0.2 times the number of dropout layers to all layers. We identified each architecture for training where the smallest validation loss was observed. The criterion for selecting the best overall performance architecture is the verification loss of each network in this state. In each cross-validation folding, the model weights are reinitialized with different random seeds to reduce the residual knowledge from the previous folding.

Data Processing

In order to randomize and arbitrate the generated new sequence, we added the mark B to the N-terminus of all amino acid sequences. In order to simplify data processing and construct sequences of the same length, we also use trailing space characters to fill the sequence to the length of the longest sequence in the training set (48 residues). All sequences are represented by one-hot encoding based on bi-

nary encoding, and the length is the same as the vocabulary. In the end, 1154 sequences, 48 padding lengths and 22 feature vector lengths constitute the final sequence data matrix. The goal of network training (solving the loss function) is the amino acid at the next residue position of each position in the input.

Sequence Generation

We sampled 2000 cycles and used fifinal to train a new sequence of the network. Each loop is called from the character B and continues sampling, unless it reaches the filled space character or the maximum sequence length (48). In order to control the variability of the sequence during sampling, we introduced a temperature-related factor in the softmax function. At the sequence position i, the temperature occurrence probability of amino acid y is defined as

$$P(y_i) = \exp(y_i/T)(\sum_{j=1}^{n} \exp(y_i/T)). \quad (2)$$

Evaluation of Generated Sequences

We analyzed the sequence generated from scratch, according to the following criteria:

1. Compare with the training data. We determined the percentage of the effective sequence without B; and the peptide we created, and compared it with the training sequence. Then we statistically compared the global peptide descriptor value between the effective sequence and the training set, and then analyzed the difference. In addition, we calculated the Euclidean distance of the training set in the global peptide descriptor space.

2. Predicted antibacterial activity. We used the CAMP AMP prediction tool to evaluate the sequence we designed. We compared the predicted pseudo-probability and the prediction of the training set through the effective sequence of AMP.

In the above two standards, the manually generated sequence may be an amphipathic helical sequence. Amphiphilicity is created by setting a basic residue every three to four amino acids and filling the spaces with hydrophobic amino acids.

Experiments

We train the LSTM model, and we use two data sets when training the model. One is the open-source antimicrobial peptide data set, and the other is the laboratory's private small protein data set: 1,000 small protein sequences with HHH, EEHE, EHEE, and HEEH (where H stands for helix and E stands for folding) structure. For the antimicrobial peptide data set, we use the antimicrobial peptide data set (AMPs) in the training model. After the model is trained, we use the model to generate protein sequences to test whether the generated protein sequences are amphipathic for antimicrobial peptides. For the small protein data set in the laboratory, we used HHH, EEHE, EHEE, and HEEH to train the model, respectively, to generate 1000 protein sequences.

We set the network as a two-layer network architecture, each layer containing 256 neurons for production operation.

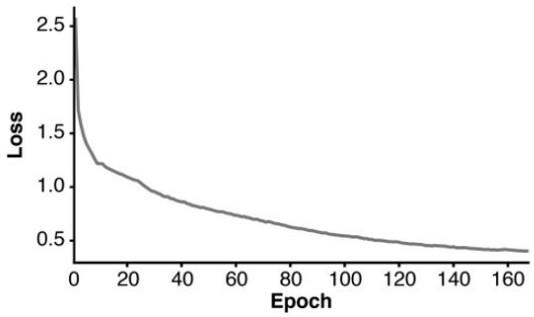

Figure 2: Loss evolution in training.

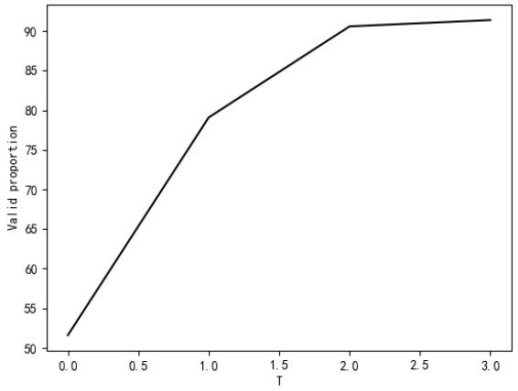

Figure 3: Percentages of valid unique sequences obtained from sampling 1000 sequences at different temperatures.

Figure 2 summarizes training performance of the network, measured in the form of classification cross-entropy loss. The average verification loss obtained after 167 periods is 0.56 ± 0.06 (mean ± SD). We chose this number of epochs to train the network on the complete data set. The evolution of the training loss of the final network is shown in the figure. The final state of the model is saved and used to sample the new sequence. During the design process of antimicrobial peptides, we tested the effect of different sampling temperatures (softmax function, equation) on the effectiveness of the constructed sequence (Table 1 and Figure 3). Setting a high temperature will result in more different sequences, and setting a low temperature will result in limited peptide sequence diversity. At a temperature of 1.25, the sequence validity begins to converge, and we decided to perform efficient sampling at this temperature. The 2000 sequences constructed through the network model produced 1747 unique peptides that were different from the training data ("valid"). In the newly generated sequence, 88% are between 7 and 48 residues (19 ± 8 (mean ± SD)).

Corresponds to the extreme case of the training data distribution. In order to allow direct comparison of the designed sequence with the training set, we randomly selected the same number of newly generated sequences as in the training set (1554) for further study. Our analysis of the main features of the generated sequence allows us to make a more

T	0.75	1	1.25	1.5
Valid(%)	51.6	79.1	90.6	91.4

Table 1: Percentages of valid unique sequences obtained from sampling 1000 sequences at different temperatures

detailed comparison with the training data set. In order to check whether the system does not just generate a sequence with a specific amino acid distribution by default, we also created a set of 1554 pseudorandom peptide sequences (7-48 residues) with the same amino acid distribution as the training set. we calculate the Euclidean distance from the sampling sequence to the training set in the combined descriptor space. The sampling sequence shows that the Euclidean distance is 0.5 ± 0.3 (mean ± SD). For comparison, the same distance calculation was performed on a random data set (0.7 ± 0.3). The calculated distance shows that the sequence generated by our model is significantly more similar to the training data compared with the comparison set (p-value < 0.05, Welch t test). The model sampling sequence is visually similar to the amino acid distribution of the training set and the random set, and the proportion of hydrophobic amino acids is slightly increased. When comparing the three groups, this increase is also reflected in the global hydrophobicity distribution (Figure 4(a)). However, the spatial orientation of hydrophobic and polar amino acids is different, resulting in a significant increase in the hydrophobic moment (p value < 0.001, two-sided Welch t test) compared with the random group (Figure 4(b)). The average is 0.38 ± 0.13 (mean ± SD), and the hydrophobic moment of the sampling sequence is also higher than that of the training set (0.36 ± 0.14). The histogram describing the distribution of the two groups of hydrophobic moments is shown in Figure S3. To quantify the antibacterial potential of the generated peptides, we used the publicly accessible AMP prediction tool from the CAMP44 server to predict their activity. We chose the CAMP random forest classifier because it performed better than other tools in a recent benchmark study. The pseudo probability of the antibacterial (P(AMP)) peptide is obtained from this random forest classifier model (Figure 4(c)). Of all the generated sequences, 82% were predicted to be"active" (P(AMP) ≥ 0.5), and 18% of the generated sequences were predicted to be"inactive" (P(AMP) < 0.5). Welch's t-test rejected the hypothesis that the random data set and the sampled data set have the same activity prediction mean (p-value < 0.01). According to AMP predictions, the new peptides generated by our LSTM RNN model have a higher probability of becoming active AMPs than random sequences with amino acid frequencies in the training set.

Similarly, we trained on a private data set and efficiently sampled at a temperature of 1.25. We use HHH, EEHE, EHEE, and HEEH to train the model, respectively, to generate 1000 protein sequences. We hope that the generated protein sequences can have the same structure. These peptides are different from the training data ("valid"). In the newly generated sequence, the length of the sequence is between 7 and 48 residues. The result is showed in Table 2 and Figure 5.

Discussion

This method uses LTSM RNN for de novo sequence design. This method is conceptually different from template based peptide design strategy and random sequence generation method, and amino acids are extracted from predefined distributions and then connected. Because LSTM RNN model depends on its internal high-dimensional sequence representation, it should not only copy specific sequence templates, but may construct "fuzzy" versions of sequences in training data. This concept is related to the "simulated molecular evolution" method based on random sequence evolution process.

The results show that the internal representation of helical amps can be generated by using LSTM RNN model. Although we call each generation process through the same initial 'B' token, the network model not only reproduces the training sequence, but also reveals its own internal interpretation of the data. This is reflected in the fact that no generated sequence is the same training peptide. Although some general peptide characteristics are different from the distribution of training data, RNN sequence generation performs better in approaching amp sequence space than random or rule-based peptide design in global peptide description subspace. The overall charge of the generated sequence is mainly positive charge, which confirms the early discovery that amp obtains its bacterial membrane targeting ability through the delicate balance of charge interaction and hydrophobicity.

The main difference between network generated sequence and random sequence is their amphiphilicity, and the hydrophobic moment of network generated set is higher. It proves this. The hydrophobic moment switches back to the regular pattern of charged and hydrophobic residues at a specific position in the sequence, which must be learned by the network. We assume that this also leads to better prediction of camp model. To clarify this hypothesis, we manually created a hypothetical amphiphilic helix sequence with alternating positive ionizable and hydrophobic amino acids. These sequences obtained higher scores than the training data through camp prediction. This result confirms our hypothesis that LSTM RNN actually learned to recognize amphiphilic grammar in peptides.

It has been shown that high hydrophobic moment alone is not a sufficient standard for strong and selective antibacterial activity. It can be reasonably assumed that a certain degree of conformational flexibility is required to obtain targeted membrane degradation activity. Therefore, generating a "fuzzy" version of the training sequence will prove to be useful for peptide design and optimization. Importantly, the sequence changes introduced by the network model are not random, but follow the changes of learning training data. Obviously, LSTM RNN is limited to the applicability covered by the training data. From the advantages of pure sequence orientation, extrapolation is difficult because peptide sequences are discrete points in chemical space and there is no smooth intermediate transition. This fact highlights the advantages of continuous descriptor space, which allows continuous transitions between molecules and supports extrapolation to a limited extent. Therefore, combining LSTM

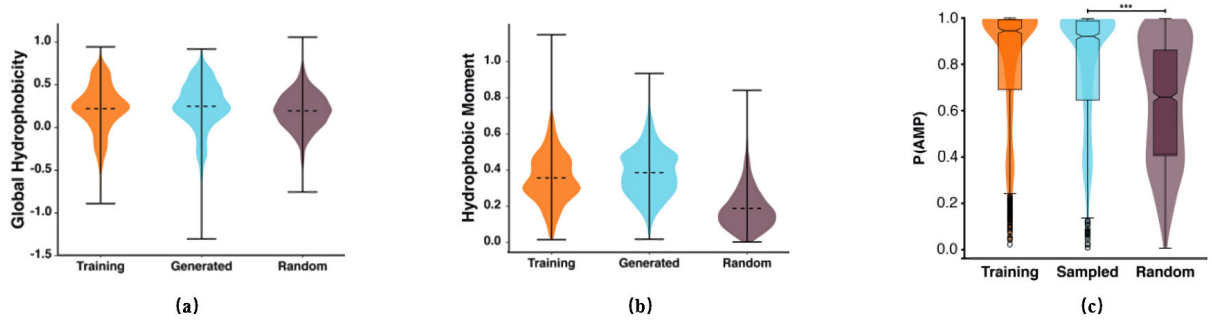

Figure 4: Comparison of the main peptide features between the training data (Training, orange), the generated sequences (Generated, blue), the pseudo-random sequences with the same amino acid distribution as the training set (Random, purple). The horizontal dashed lines represent the mean (violin plots) and median (box plots) values, and the whiskers extend to the most extreme nonoutlier data points. (a) Eisenberg hydrophobicity, (b) Eisenberg hydrophobic moment, (c) Pseudo-probabilities of sequences predicted to be AMPs.

	HHH	EEHE	EHEE	HEEH
Training sequences	1000	1000	1000	1000
Generated sequences	1000	1000	1000	1000
Valid sequences	927	946	962	897
Same structure	846	685	752	704
Valid sequences proportion	92.70%	94.60%	96.20%	89.70%
Same structure proportion	91.26%	72.41%	78.17%	78.48%

Table 2: Percentages of valid unique sequences and same structure sequences of each type.

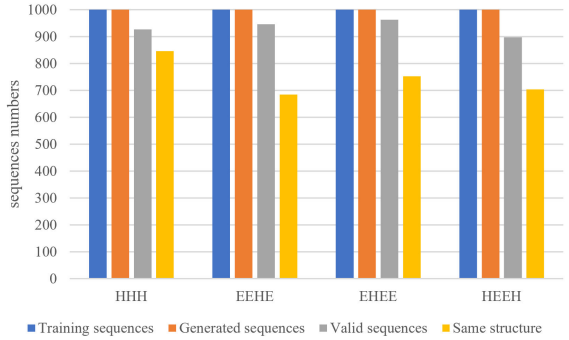

Figure 5: Valid unique sequences and same structure sequences of each type.

RNN with the prediction model trained by molecular descriptors can be used to design a customized activity focused amino acid sequence library.

Conclusion

We combined RNN with LSTM to generate amino acid sequence data and design peptides from scratch. The experimental results show that the network meets the requirement of generating peptide library of specified size. This positive result proves that LSTM and RNN can be widely used in a variety of peptide design tasks. From the results of this study, we conclude that the method may be best suited for use in conjunction with predictive models to assess the quality of the generated sequences. We have proved the effectiveness of the model by generating antimicrobial peptides and generating proteins with the same structure as the training set. This model has the advantages of simple training and strong scalability. It improves the long-term dependence problem in RNN; The performance of LSTM is usually better than time recurrent neural network and hidden Markov model (HMM); as a nonlinear model, LSTM can be used as a complex nonlinear element to construct larger depth neural network. But at the same time, because the model is simple, it has the shortcomings of over-fitting and incomplete feature extraction. Another disadvantage is that the gradient problem of RNN has been solved to a certain extent in LSTM and its variants, but it is still not enough. It can handle sequences of 100 orders of magnitude, but it will still be difficult for sequences of 1000 orders of magnitude or longer; Another disadvantage is that each LSTM cell means that there are four full connection layers (MLPs). If the LSTM has a large time span and the network is deep, the amount of calculation will be large and time-consuming. There is still a long way to go before putting the model into actual production.

References

Angermueller, C.; Dohan, D.; Belanger, D.; Deshpande, R.; Murphy, K.; and Colwell, L. 2019. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*.

Baldi, P.; and Pollastri, G. 2003. The principled design of large-scale recursive neural network architectures–dag-rnns

and the protein structure prediction problem. *The Journal of Machine Learning Research*, 4: 575–602.

Cao, Y.; Das, P.; Chenthamarakshan, V.; Chen, P.-Y.; Melnyk, I.; and Shen, Y. 2021. Fold2Seq: A Joint Sequence (1D)-Fold (3D) Embedding-based Generative Model for Protein Design. In *International Conference on Machine Learning*, 1261–1271. PMLR.

Colton, S.; De Mántaras, R. L.; and Stock, O. 2009. Computational creativity: Coming of age. *AI Magazine*, 30(3): 11–11.

Fjell, C. D.; Hiss, J. A.; Hancock, R. E.; and Schneider, G. 2012. Designing antimicrobial peptides: form follows function. *Nature reviews Drug discovery*, 11(1): 37–51.

Hawkins-Hooker, A.; Depardieu, F.; Baur, S.; Couairon, G.; Chen, A.; and Bikard, D. 2021. Generating functional protein variants with variational autoencoders. *PLoS computational biology*, 17(2): e1008736.

Heffernan, R.; Yang, Y.; Paliwal, K.; and Zhou, Y. 2017. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. *Bioinformatics*, 33(18): 2842–2849.

Hochreiter, S.; Heusel, M.; and Obermayer, K. 2007. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14): 1728–1736.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Huang, P.-S.; Boyken, S. E.; and Baker, D. 2016. The coming of age of de novo protein design. *Nature*, 537(7620): 320–327.

Ingraham, J.; Garg, V. K.; Barzilay, R.; and Jaakkola, T. 2019. Generative models for graph-based protein design.

Jing, B.; Eismann, S.; Suriana, P.; Townshend, R. J.; and Dror, R. 2020. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*.

Liu, L.; Tang, L.; Dong, W.; Yao, S.; and Zhou, W. 2016. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, 5(1): 1–22.

Miyao, T.; and Funatsu, K. 2017. Finding chemical structures corresponding to a set of coordinates in chemical descriptor space. *Molecular informatics*, 36(8): 1700030.

Qi, Y.; and Zhang, J. Z. 2020. DenseCPD: improving the accuracy of neural-network-based computational protein sequence design with DenseNet. *Journal of chemical information and modeling*, 60(3): 1245–1252.

Qian, N.; and Sejnowski, T. J. 1988. Predicting the secondary structure of globular proteins using neural network models. *Journal of molecular biology*, 202(4): 865–884.

Repecka, D.; Jauniskis, V.; Karpus, L.; Rembeza, E.; Rokaitis, I.; Zrimec, J.; Poviloniene, S.; Laurynenas, A.; Viknander, S.; Abuajwa, W.; et al. 2021. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3(4): 324–333.

Schneider, P.; and Schneider, G. 2016. De novo design at the edge of chaos: Miniperspective. *Journal of medicinal chemistry*, 59(9): 4077–4086.

Segler, M. H.; Kogej, T.; Tyrchan, C.; and Waller, M. P. 2018. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1): 120–131.

Sønderby, S. K.; and Winther, O. 2014. Protein secondary structure prediction with long short term memory networks. *arXiv preprint arXiv:1412.7828*.

Sutskever, I.; Martens, J.; and Hinton, G. E. 2011. Generating text with recurrent neural networks. In *ICML*.

White, D.; and Wilson, R. C. 2010. Generative models for chemical structures. *Journal of chemical information and modeling*, 50(7): 1257–1274.

Wu, Z.; Yang, K. K.; Liszka, M. J.; Lee, A.; Batzilla, A.; Wernick, D.; Weiner, D. P.; and Arnold, F. H. 2020. Signal peptides generated by attention-based neural networks. *ACS Synthetic Biology*, 9(8): 2154–2161.