EPiC Series in Computing

Volume 70, 2020, Pages 182–191

Proceedings of the 12th International Conference on Bioinformatics and Computational Biology

Exploring Deep Neural Network Architectures: A Case Study on Improving Antimicrobial Peptide Recognition

Manpriya Dua[1], Daniel Barbará[1], and Amarda Shehu[1,2,3,4]

[1] Department of Computer Science, George Mason University, Fairfax, VA
[2] Center for Advancing Human-Machine Partnerships, George Mason University, Fairfax, VA
[3] Department of Bioengineering, George Mason University, Fairfax, VA
[4] School of Systems Biology, George Mason University, Manassas, VA
amarda@gmu.edu

Abstract

With antibiotic resistance on the rise, health organizations are urging for the design of new drug templates. Naturally-occurring antimicrobial peptides (AMPs) promise to serve as such templates, as they show lower likelihood for bacteria to form resistance. This has motivated wet and dry laboratories to seek novel AMPs. The sequence diversity of these peptides, however, renders systematic wet-lab screening studies either infeasible or too narrow in scope. Dry laboratories have focused instead on machine learning approaches. In this paper, we explore various deep neural network architectures aimed at improving antimicrobial peptide recognition. Our enquiry results in several architectures with comparable or better performance than existing, state-of-the-art discriminative models.

1 Introduction

Bacterial resistance to one of more antibiotics has become a serious health concern [17, 24]. A recent study estimates that three million common surgical procedures can potentially become life-threatening due to such resistance [2]. Many health organizations are calling for the development of novel antibacterial drugs [23].

Naturally-occurring antimicrobial peptides (AMPs), natural components of innate immunity, are popular targets for developing new antibacterial drugs, as they have shown a lower likelihood for bacteria to form resistance compared to conventional drugs [12]. However, in doses required to inhibit bacterial growth, many AMPs are toxic to the host [14]. These findings have motivated wet and dry laboratories to expand their search for novel, laboratory-designed AMPs.

A significant challenge to these efforts is our lack of understanding on what determines antimicrobial activity, or, at a minimum, an expedient, and yet general model via which we can predict such activity. Many wet-laboratory efforts have focused on understanding the physicochemical properties that govern antimicrobial activity. However, these efforts face outstanding challenges due to the high diversity of AMPs on sequence, structure, and mechanism of action. For instance, AMPs comprise diverse sequence families, such as, cathelicidins, defensins, cecropins, and others; they have diverse secondary and tertiary structure, and they kill their targets through various mechanisms, such as, cell membrane damage, DNA interference, or signaling for adaptive immune responses [22]. Given such diversity, current wet-laboratory studies that rely on systematic screening are invariably narrow in scope [6]; we note that some focused studies have had success in revealing novel, naturally-occurring AMPs or designing synthetic AMPs by optimizing known ones [1,3,8].

Dry-laboratory approaches, on the other hand, have primarily focused on training machine learning models to recognize antimicrobial activity in given peptide sequences. These approaches have been aided by increasing deposition of experimentally-known AMPs in public databases and have resulted in discriminative models of increasing accuracy [10,11,18,20,25].

Motivated by the most recent research on discriminative models based on deep neural networks [18] and the overall increasing popularity of deep learning in bioinformatics, in this paper we carry out a systematic study on deep neural network architectures for the problem of antimicrobial activity recognition. We investigate choices with regards to the low-level representation layer, the number of layers, and the choice on dense versus convolution versus recurrent layer(s). These choices result in several architectures whose performance we contextualize in the body of machine learning methods for antimicrobial activity recognition. Our evaluation yields several interesting observations. For instance, we demonstrate that the recurrent layers do not aid performance and instead a simpler neural network without such layers performs comparably to the one proposed in [18]. We also show that convolution layers are useful and utilizing several layers aids performance. In addition, we demonstrate that the choice of the low-level representation matters, particularly when not very deep architectures can be afforded in the presence of constraints regarding the size of the labeled data.

The rest of this paper is organized as follows. A brief summary of related work on discriminative models for AMP recognition is provided in Section 1.1. The deep neural network architectures used for AMP recognition and the role of each layer in these architectures are described in detail in Section 2. Evaluation of these architectures is then related in Section 3. The paper concludes in Section 4.

1.1 Related Work

Published discriminative models largely focus on a binary classification setting and include artificial neural networks (ANN) [16], discriminant analysis (DA) [15], fuzzy k-nearest neighbor [25], hidden Markov models [5], logistic regression [13,20], random forests (RF) [19], support vector machines (SVM) [10,11], and deep neural networks (DNNs) [18]. Popular features are based on sequence composition, order [11,25], physico-chemical properties, such as charge, hydrophobicity, and more [13,15,16]. Complex features that encode distal interactions are built via evolutionary algorithms [20] or DNNs [18].

Some models are provided as web servers, including iAMPpred [11], iAMP-2L [25], AntiBP2 [9], CAMP [15], AMPer [4], AMP Scanner [18], and others. In particular, AMP Scanner is designed to support high-throughput screening experiments for wet-laboratory researchers to conduct systematic virtual screenings of peptide libraries to identify promising peptides for further characterization and optimization.

Comparative surveys report that state-of-the-art methods (and servers), including CAMP and AntiBP2, miss many true positives [1,19]. Others, such as the Antimicrobial Peptide Database (APD) AMP predictor [21], only accept individual query sequences, which limit applications for high-throughput recognition experiments by wet-laboratory researchers. The DNN proposed in [18] improves upon these methods and is considered one of the top performers.

The DNN model is trained over naturally-occurring AMP sequences (positive training dataset) and carefully-constructed, non-AMP sequences (negative training dataset). In [18], the positive and negative classes are balanced for the model training. It is worth noting that the construction of the negative data can be particularly challenging in bioinformatics applications. We direct the reader to the work in [18] for details on the process followed to construct the negative training dataset.

The DNN model contains convolutional and recurrent layers, followed by a single-node sigmoid unit that outputs the probability of a peptide sequence to possess antimicrobial activity. The convolutional layer allows capturing position-invariant patterns along the amino-acid sequence of a peptide. The recurrent layer is a long short term memory (LSTM) layer that allows recognizing and forgetting gap-separated patterns. Further details on the model's architecture, training, and its evaluation can be found in [18].

2 Methods

We treat the DNN model [18] described in Section 1.1 as the baseline architecture over which we remove or add layers and instigate other changes. In what follows, we proceed systematically. We first analyze the significance (and contribution) of each of the layers prior to further modifying promising resulting architectures via convolutional and/or recurrent layers. We do so in the context of several choices regarding the low-level features input to the model in the embedding layer.

2.1 Investigating the Low-level Representation

Let us recall that there are 20 classic amino acids found in nature, represented by 20 letters of the English alphabet. Lengths of known AMPs vary from a few to over a few 100 amino acids. In [18], due to uneven lengths, all AMP sequences are first padded (to a maximum length) with "X", which encodes the undetermined amino acid. Each thus-encoded constitutive amino acid is represented with a vector of chosen fixed length. This is done automatically by using Keras's *Embedding Layer*. This layer plays an important role in the classification model; without it, the amino acids which are represented using letters from the English alphabet need to be converted to either integers or to other numerical representations so that they can be processed.

We explored several alternative ways of representing the training sequences. We utilize a short AMP sequence "KIIFLIAI" to illustrate these representations below.

a) Integer Representation: Each amino acid is represented by a corresponding integer in the range [0,20]. The AMP sequence "KIIFLIAI" is then represented as "[9 8 8 10 8 1 8]" in integer representation. Further, the representation is padded with zeros (representing the undetermined amino acid) until maximum length. Making only this choice (removing the embedding layer) in the DNN proposed in [18] results in a model to which we refer as DNN-Integer.

b) One-Hot-Encoded Representation: Each amino acid is represented as a vector of length 21 (due to 21 different amino acids including the undetermined amino acid 'X'), instead of an integer as above. All positions in the vector are 0 except for a 1 in the integer position that represents this amino acid. The AMP sequence "KIIFLIAI" is then represented as "[[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0] [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]]." Further, the representation is padded with the vector "[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]" representing X until maximum length. Making this choice instead of utilizing the Embedding Layer as in [18] results in a model to which we refer as DNN-One-Hot.

c) Binary Representation: Each amino acid is represented with a vector of length 5. Each vector represents its corresponding integer value's binary representation. The AMP sequence"KIIFLIAI" is represented as "[[0 1 0 0 1] [0 1 0 0 0] [0 1 0 0 0] [0 0 1 0 1] [0 1 0 1 0] [0 1 0 0 0] [0 0 0 0 1] [0 1 0 0 0]]". Further, the representation is padded with [0 0 0 0 0], representing X until maximum length. Making this choice instead of utilizing the Embedding Layer as in [18] results in a model to which we refer as DNN-Binary.

K-mer Count Representation In contrast to the above representations, which represent each single amino acid, we now turn to utilizing counts of k-mers present in known AMP sequences from the training set. We consider all k-mer sub-sequences that are present in the training set at least n times, where the value of n is chosen based on experimental results. Each sequence is represented by the number of times the chosen k-mers appear in it, and each position in the sequence representation provides a count of the corresponding k-mer. We also experiment with a few different values of k for the k-mer count representation. We refer to the resulting DNN models as DNN-k-mer, with the value of k indicated. These models are much simpler, as they do not have to deal with entire sequences; they are much faster to train and have fewer layers for processing the input. Therefore, we utilize dense layers in the DNN–mer models rather than the rest of the DNN architecture (a convolutional and LSTM layer). We experiment with varying the number of layers, and the number of nodes in each layer. We use ReLU activation for the input and middle layers and Sigmoid activation for the output layer.

Byte Pair Encoding with AMPScanner We utilize a byte pair encoding (BPE) representation. In BPE, the most frequent peptide sub-sequences of length 2 (most frequent 2-mers) are replaced by new symbols, and these symbols are stored in memory for reference. Once the first set of replacements is done, a new set of most frequent 2-mers is chosen, and they are again replaced with new symbols. This process is repeated until a certain number of new symbols has been introduced. The idea is that most frequent sub-sequences would now be represented with shorter sub-sequences involving symbols; instead of peptide sequences, we have sequences that also contain symbols (meta amino amino acids). This new representation is fed to the Keras' Embedding Layer. We refer to the resulting DNN models as DNN-BPE.

2.2 Investigating the Layers

2.2.1 Convolutional Layer

This layer should play the most important role in identification of AMPs, since it focuses on smaller sub-sequences of neighboring amino acids and uses these sub-sequences for feature extraction. We conduct the following anatomical analysis. We first remove the convolutional layer from the baseline DNN [18] and refer to this resulting model as DNN-Conv^0. We observe the implication of this decision in the performance of the model. Alternatively, we retain the convolutional layer of the baseline model and add another layer, as well as vary the number of filters and kernel size (filter length) of the convolutional layers. We refer these resulting models as DNN-Conv^2.

2.2.2 Max-Pooling Layer

We recall that the role of this layer is to condense the features explored by the convolutional layer by pooling them and assigning the condensed value as the maximum in a given pool. We observe the effects of eliminating the max-pooling layer, while keeping the rest of the model architecture as is. We refer to this resulting model as DNN-MaxPool^0.

2.2.3 LSTM Layer

The LSTM layer is a recurrent layer. Recurrent layers are used to identify dependencies among sequences. So, a recurrent layer would identify how previously looked at sequences affect the next sequences. We make two overall decisions, removing the LSTM layer or replacing it with another layer. Removing it results in a model to which we refer as DNN-LSTM^0. Alternatively, we replace the LSTM layer with a Gated Recurrent Unit (GRU), a SimpleRNN layer, and a bidirectional LSTM layer; the BiLSTM layer helps in identifying dependencies among sequential data in both directions. We refer to these resulting models as DNN-GRU, DNN-SimpleRNN, and DNN-BiLSTM, respectively.

2.3 Discriminative Deep Model with Attention Mechanism

Finally, we investigate an attention mechanism by adding it to the best performing model from the models described above. Our implementation is based on the model described in DeepHINT [7]. First, we perform one-hot encoding on the input sequences. These one-hot encoded sequences are passed through two 1-D convolution layers (placed one after the other) followed by max-pooling layers, respectively. The obtained feature representation is further sent to two separate units, the first being a single neuron combining the output received from the max-pooling layer, and the second being an attention mechanism. The attention mechanism computes an importance score indicating the amount of attention paid to sequence features at each position based on the input feature vectors received from the max-pooling layer. Each feature vector obtained from the convolution and max pooled filters is normalized by the computed importance scores as in DeepHINT [7]. The outputs from these two units are concatenated and used for prediction of sequences containing AMP properties.

2.4 Implementation Details

For the purpose of this case study, we obtain data from APD3 (Anti-microbial Peptide Database). The dataset is also freely available to the research community at `http://www.ampscanner.com`. The AMPs, 1778 in all, are active against Gram-positive and/or Gram-negative bacteria. As in [18], we assign 712 AMPs for training, 354 for tuning/evaluation, and 712 for testing, respectively.

Our implementations all the DNN models described above are in Python using Keras. For k-mer models, we use $k \in [1, 2, 3, 4, 5]$; based on experimentation, we limit to a total of 3 dense layers for these models. We choose the number of epochs, number of filters and their sizes for the convolution layers, and the number of recurrent units based on experimentation results with training and validation accuracy and loss plots; we utilize dropout in layers to avoid over-fitting.

All models are run on a MacBook Pro with 2.7 GHz Dual-Core Intel Core i5 base processor and 16GB of RAM. The baseline DNN takes 342 seconds to train and evaluate. Removing the embedding layer and using other encodings (integer, one-hot, or binary) takes $143 - 173$ seconds to train and evaluate. Removing the convolution and the max-pooling layer takes 135 seconds to train and evaluate. Removing the LSTM layer takes 210 seconds to train and evaluate. Replacing the LSTM layer with GRU takes 296 seconds; replacing the LSTM layer with SimpleRNN takes 166 seconds, and replacing the LSTM layer with the bidirectional LSTM layer takes 311 seconds to train and evaluate. Introducing a second convolution and max pooling

Table 1: Comparison of models with varying sequence representations.

Models	Max Accur	Accur	Prec	Rec	F1	AUROC
DNN	**0.924**	**0.917**	**0.927**	**0.926**	**0.926**	**0.972**
DNN-Integer	0.700	0.682	0.767	0.654	0.699	0.760
DNN-Binary	0.878	0.869	0.881	0.892	0.886	0.940
DNN-One-Hot	**0.926**	**0.917**	**0.930**	**0.923**	**0.926**	**0.972**
DNN-1-mer	**0.921**	**0.907**	0.912	**0.926**	**0.919**	**0.964**
DNN-2-mer	0.907	0.897	0.907	0.913	0.910	0.956
DNN-3-mer	0.897	0.883	0.903	0.890	0.896	0.942
DNN-4-mer	0.872	0.862	0.887	0.871	0.879	0.940
DNN-5-mer	0.854	0.843	**0.915**	0.821	0.866	0.910
DNN-BPE	0.878	0.873	0.892	0.876	0.888	0.942

layer takes 148 seconds to train and evaluate. The model with the attention mechanism takes 46 seconds to train and evaluate. The k-mer models with $k \in \{1, 2, 3, 4, 5\}$ take 21, 13, 90, 501, and 876 seconds, respectively. The BPE model takes 717 seconds to train and evaluate.

3 Results

We first compare the performance of the DNN variants with different sequence representations. We do so over the testing dataset. Table 1 lists averages of the accuracy, precision, recall, F1, and AUROC over 5 runs of each model. The maximum accuracy (over the 5 runs) is also shown. The top three AUROC values and the three top maxiumum accuracy values are highlighted in bold font.

The top three values under each metric in Table 1 are highlighted in bold font. For the most part, they belong to the same three models, DNN, DNN-One-Hot, and DNN-1-mer (with the exception being DNN-5-mer which reaches the third highest average precision). In particular, the DNN-One-Hot model performs comparably to DNN when considering all metrics. A decrease in accuracy is observed when using integer and binary encoding. This decrease is due to the uneven representation of different amino acids. Table 1 also shows that the DNN-k-mer models with $k > 1$ do not outperform DNN and DNN-One-Hot; DNN-1-mer is the only DNN-k-mer model reaching comparable performance to DNN and DNN-One-Hot. These results indicate that higher-level features based on compositional features are not particularly informative, possibly because of the scarcity of longer motifs on short AMPs. It is worth noting that, since DNN-1-mer outperforms DNN-2-mer, it is not surprising that DNN-BPE is also outperformed; the BPE representation is similar to the 2-mer one, in the sense that it looks at sub-sequences of length 2 and condenses them into one symbol.

Based on the results shown in Table 1, the rest of the models we investigate retain the embedding layer as in the baseline DNN in [18]. Table 2 relates the impact of the choices with regard to the recurrent layer. Recall that in the DNN-RNN^0, DNN-SimpleRNN, DNN-GRU, and DNN-BiLSTM models we investigate, the embedding layer, the convolutionational layer, and the max pooling layer are kept as in the baseline DNN. The only modification is with regards to the recurrent layer, which is removed entirely or replaced with variants.

Analysis of the results in Table 2, where the top three values under each metric are highlighted in bold font, reveals that the recurrent layer is not essential to achieve a very high performance, comparable or higher than the baseline DNN model. In almost all metrics, the

Table 2: Comparison of models with changes on the recurrent layer.

Models	Max Accur	Accur	Prec	Rec	F1	AUROC
DNN	**0.924**	**0.917**	0.927	**0.926**	0.926	**0.972**
DNN-RNN^0	**0.933**	**0.922**	**0.945**	**0.919**	**0.931**	0.970
DNN-SimpleRNN	0.920	0.907	**0.928**	0.909	0.918	0.964
DNN-GRU	**0.927**	**0.920**	**0.948**	0.911	**0.929**	**0.974**
DNN-BiLSTM	0.919	0.899	**0.928**	**0.927**	**0.927**	**0.972**

Table 3: Comparison of models with changes on the convolutional and max pooling layer(s).

Models	Max Accur	Accur	Prec	Rec	F1	AUROC
DNN	**0.924**	**0.917**	**0.927**	**0.926**	**0.926**	**0.972**
DNN-Conv^0-MaxPool^0	0.882	0.873	**0.912**	0.859	0.884	0.943
DNN-Conv^0-MaxPool^1	0.895	0.868	0.898	0.870	0.880	0.941
DNN-Conv^1-MaxPool^0	**0.918**	0.883	0.909	0.899	0.900	**0.958**
DNN-Conv^2	**0.938**	**0.926**	**0.934**	**0.938**	**0.936**	**0.972**
DNN-Conv^2-Att	0.921	0.911	0.962	0.914	0.937	0.81

DNN with no RNN layer outperforms the other variants. The DNN-GRU model achieves a good performance, as well, but underperforms on many metrics compared to the DNN-RNN^0. These results highlight two observations. First, assumptions that complexity affords higher performance need to be carefully evaluated. In this case, for instance, a recurrent layer is not necessary. Second, the reason for the recurrent layer not contributing to a superior performance may be related to the fact that sequence diversity among AMPs is quite high; that is, knowing something about one AMP may not help much to make a prediction on another. This further illustrates the challenge of the learning task in this bioinformatics problem.

Based on the results shown in Table 2, the rest of the models we investigate retain the embedding layer as in the baseline DNN but remove the LSTM layer. Table 3 lists the performance of several variants with no, one, or two convolution layers (and no, one, or two max pooling layer). Analysis of the results in Table 3, where the top three values (above 0.9) under each metric are highlighted in bold font, allows reaching several conclusions. The convolution layer plays an important role in the model, as removing this layer brings down performance. This suggests that sub-sequences and neighboring amino acids indeed play a role in determining the antimicrobial activities of the whole sequence. The max-pooling layer helps in condensing the data exploded by the convolution layer. As observed, it does not have much effect on the model performance by itself, but it does plays a role in combination with the convolution layer. On experimenting by replacing the LSTM layer with SimpleRNN, GRU, or BiLSTM, we see that there is no significant change in the results. However, when using SimpleRNN, the model trains much faster and in fewer epochs. Most importantly, the analysis reveals a winning model that outperforms the baseline DNN model. Adding a second convolutional layer to the network yields the best performance; adding more convolutional layers, however, does not improve performance (data not shown). The overall architecture of the best-performing model, DNN-Conv^2, is shown in Figure 1. Adding an attention layer to the best-performing model DNN-Conv^2, to which we refer as DNN-Conv^2-Att in Table 3, lowers the performance.

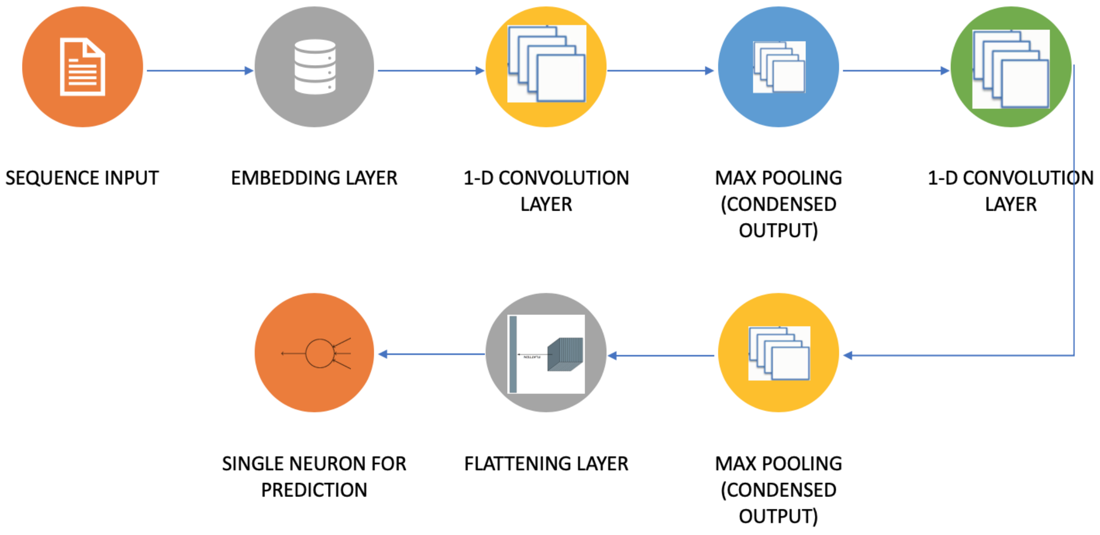
SEQUENCE INPUT
EMBEDDING LAYER
1-D CONVOLUTION LAYER
MAX POOLING (CONDENSED OUTPUT)
1-D CONVOLUTION LAYER
MAX POOLING (CONDENSED OUTPUT)
FLATTENING LAYER
SINGLE NEURON FOR PREDICTION

Figure 1: Best Performing Deep Neural Network Architecture For AMP Recognition.

4 Conclusion

This paper has presented a case study on model architectures for classification of AMP Sequences. Prompted by a state-of-the-art deep, discriminative model recently developed in our laboratory [18] that outperforms other machine learning approaches on this problem, we have presented here an improvement by altering the model (removing the LSTM layer and adding additional convolution and max-pooling layers), and have suggested other alternate simpler and faster models. The evaluation of several models suggests that recurrent layers play no role in the identification of AMP sequences, and that the removal of this layer from the model actually improves the model performance. The improved performance with adding a convolution layer suggests that there are patterns that help AMP recognition that emerge even from the already convolved features. It is worth noting that these models are fast and can be used to virtually screen peptide sequences. As already understood, however, these deep models do not provide any insight as to what exactly governs antimicrobial activity. They do not readily provide rules that wet-laboratory researchers may employ to intelligently design novel AMPs. Attention layers may provide some guidance in this direction. Our preliminary investigation, however, shows that adding an attention layer lowers the performance. While it is possible that this may be due to the size of the training dataset, we intend to investigate this direction further and experiment with variant implementations of attention layers.

5 Acknowledgements

We thank members of the Shehu laboratory for useful feedback on this work.

References

[1] Barney M. Bishop, Melanie L. Juba, Megan C. Devine, Stephanie M. Barksdale, Carlos Alberto Rodriguez, Myung C. Chung, Paul S. Russo, Kent A. Vliet, Joel M. Schnur, and Monique L. van Hoek. Bioprospecting the american alligator (*Alligator mississippiensis*) host defense peptidome. *PLoS ONE*, 10(2):e0117394, 02 2015.

[2] S. Boseley. Overuse of antibiotics risks return to dark ages of life-threatening surgery, 2018.

[3] M. C. Chung, S. N. Dean, C. N. Propst, B. M. Bishop, and M. L. van Hoek. Komodo dragon-inspired synthetic peptide drgn-1 promotes wound-healing of a mixed-biofilm infected wound. *NPJ Biofilms Microbiomes*, 3(9), 2017.

[4] C. Fjell, R. Hancock, and A. Cherkasov. AMPer: a database and an automated discovery tool for antimicrobial peptides. *Bioinformatics*, 23(9):1148–1155, 2007.

[5] C. Fjell, H. Jenssen, K. Hilpert, W. A. Cheung, N. Pante, R. E. Hancock, and A Cherkasov. Identification of novel antibacterial peptides by chemoinformatics and machine learning. *J. Med. Chem.*, 52(7):2006–2015, 2009.

[6] Christopher D Fjell, Jan A Hiss, Robert EW Hancock, and Gisbert Schneider. Designing antimicrobial peptides: form follows function. *Nature reviews Drug discovery*, 11(1):37–51, 2012.

[7] Hailin Hu, An Xiao, Sai Zhang, Yangyang Li, Xuanling Shi, Tao Jiang, Linqi Zhang, Lei Zhang, and Jianyang Zeng. DeepHINT: understanding HIV-1 integration via deep learning with attention. *Bioinformatics*, 35(10):1660–1667, 10 2018.

[8] Melanie L Juba, Paul S Russo, Megan Devine, Stephanie Barksdale, Carlos Rodriguez, Joel M Schnur, Monique L van Hoek, and Barney M Bishop. Discovery of novel antimicrobial peptides from Varanus komodoensis (komodo dragon) by large-scale analyses and de-novo-assisted sequencing using electron-transfer dissociation mass spectrometry. *J Proteome Res*, 16(4):1470–1482, 2015.

[9] S. Lata, N. K. Mishra, and G. P. Raghava. AntiBP2: improved version of antibacterial peptide prediction. *BMC Bioinformatics*, 11(Suppl 1):S1–S19, 2010.

[10] Ernest Y Lee, Benjamin M Fulan, Gerard CL Wong, and Andrew L Ferguson. Mapping membrane activity in undiscovered peptide sequence space using machine learning. *Proceedings of the National Academy of Sciences*, 113(48):13588–13593, 2016.

[11] Prabina Kumar Meher, Tanmaya Kumar Sahu, Varsha Saini, and Atmakuri Ramakrishna Rao. Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC. *Scientific Reports*, 7(42362), 2017.

[12] R. Nuti, N. S. Goud, A. P. Saraswati, R. Alvala, and M. Alvala. Antimicrobial peptides: A promising therapeutic strategy in tackling antimicrobial resistance. *Curr Med Chem*, 24(38):4303–4314, 2017.

[13] Elena G. Randou, Daniel Veltri, and Amarda Shehu. Binary response models for recognition of antimicrobial peptides. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, page 76. ACM, 2013.

[14] Jiřina Slaninová, Veronika Mlsová, Hilda Kroupová, Lukáš Alán, Tereza Tumová, Lenka Monincová, Lenka Borovičková, Vladimír Fučík, and Václav Čeřovskỳ. Toxicity study of antimicrobial peptides from wild bee venom and their analogs toward mammalian normal and cancer cells. *Peptides*, 33(1):18–26, 2012.

[15] S. Thomas, S. Karnik, R. S. Barai, V. K. Jayaraman, and S. I. Thomas. CAMP: a useful resource for research on antimicrobial peptides. *Nucl. Acids Res.*, 38(Suppl 1):D774–D780, 2009.

[16] Marc Torrent, David Andreu, Victòria M Nogués, and Ester Boix. Connecting peptide physicochemical and antimicrobial properties by a rational prediction model. *PLoS One*, 6(2):e16968, 2011.

[17] U.S. Department of Health and Human Services. Antibiotic resistance threats in the united states, 2013.

[18] D. Veltri, U. Kamath, and A. Shehu. Deep learning improves antimicrobial peptide recognition.

Bioinformatics, 34(16):2740–2747, 2018.

[19] Daniel Veltri. *A Computatioanl and Statistical Framework for Screening Novel Antimicrobial Peptides*. PhD dissertation, George Mason University, 2015.

[20] Daniel Veltri, Uday Kamath, and Amarda Shehu. Improving recognition of antimicrobial peptides and target selectivity through machine learning and genetic programming. *Transactions on Computational Biology and Bioinformatics*, 14(2):300–313, 2017.

[21] Guangshun Wang, Xia Li, and Zhe Wang. APD3: the antimicrobial peptide database as a tool for research and education. *Nucl Acids Res*, 44:D1087–D1093, 2016.

[22] William C Wimley and Kalina Hristova. Antimicrobial peptides: successes, challenges and unanswered questions. *The Journal of membrane biology*, 239(1-2):27–34, 2011.

[23] World Health Organization. Antimicrobial resistance: global report on surveillance, 2014.

[24] World Health Organization. Antibacterial agents in clinical development – an analysis of the antibacterial clinical development pipeline, including tuberculosis, 2017.

[25] Xuan Xiao, Pu Wang, Wei-Zhong Lin, Jian-Hua Jia, and Kuo-Chen Chou. iAMP-2L: A two-level multi-label classifier for identifying antimicrobial peptides and their functional types. *Analytical biochemistry*, 2013.