RECENT ADVANCES IN ANTIMICROBIALS IDENTIFICATION USING NEURAL NETWORKS

[1]Dr.S.SAVITHIRI
[1]PROFESSOR Department of CHEMISTRY
Sri Manakula Vinayagar Engineering College
Madagadipet, Pincode: 605107, Puducherry, India
E-mail ID : harishoct52007@gmail.com
Orcid id: 0009-0006-4535-9705

[2]Dr.K.KARTHIKEYAN
[2]PROFESSOR Department of CHEMISTRY
Sri Manakula Vinayagar Engineering College
Madagadipet, Pincode: 605107, Puducherry, India
E-mail ID: karthikeyank2005@gmail.com
Orcid id: 0009-0007-6104-0494

[3]Dr.SUDHEER MANAWADI
[3]ASSISTANT PROFESSOR DEPARTMENT OF *BIOTECHNOLOGY
GOVERNMENT SCIENCE COLLEGE (AUTONOMOUS), HASSAN*
SALAGAME ROAD, HASSAN, Pincode- 573201, KARNATAKA, INDIA
E-mail ID – sudheermanawadi@gmail.com
Orchid id-https://orcid.org/0000-0001-5927-694X

[4]Barinderjit Singh
[4]Assistant Professor Department of Food Science and Technology
I.K. Gujral Punjab Technical University
Kapurthala, Punjab, India -144601
Email: barinderjitsaini@gmail.com
ORCID ID:- https://orcid.org/0000-0002-7564-0550

Abstract: Peptides that are efficient against bacteria, fungi, and viruses are known as antimicrobial peptides. Feature vectors generated by the vast majority of machine learning approaches are always the same length, even though the number of amino acids in various peptides could vary widely. It is common practise to select features that are not optimal for the task at hand since there is a lack of direction regarding whether or not the features reflect periodic patterns in the peptide sequence that are significant to the classification issue that is now being faced. As a consequence of this, the product is permitted to contain a sizeable amount of filler we build a feature vector by constructing feature representations of individual amino acids. This allows us to tackle the challenges we were facing. This indicates that there will be no surprises regarding the size of the final feature vector. When it comes to the classification of antimicrobial peptides, this study evaluates the performance of k-nearest neighbour classifiers, Random Forest classifiers, and LSTM Recurrent neural networks to see which yields the most accurate results.

Keywords: Antimicrobial peptides, Recurrent neural networks, Antibacterial , LSTM

I. INTRODUCTION

It has been proven since the discovery and widespread usage of antibiotics that the bacteria and viruses doctors use to treat their patients eventually develop resistance to the drugs. Antibiotic resistance is spreading rapidly, and new antibiotics are urgently needed to stop it. A serious public health issue is the worldwide proliferation of bacteria resistant to antibiotics. The need for effective antibiotics during medical procedures, the prevalence of fatal resistant infections, and the rising cost of treating such infections all contribute to the problem. Antibiotic resistance is responsible for an estimated 2 million annual instances of sickness and 23,000 annual deaths [1], the vast majority of which occur in underdeveloped countries. Furthermore, healthcare facilities have seen an increase in the prevalence of bacteria resistant to antibiotics [2] over the previous few decades. Pharmaceutical companies have less incentive to invest in the discovery of novel antibiotics since the turn of the century [3, 4], and neither the government nor academic institutions have allocated sufficient resources to the cause. Antimicrobial peptides remain a type of essential resistant scheme component that acts as a front line of defence against bacteria and other infections [3, 5]. Gene-expressed small amphipathic peptides (AMPs) are effective against a wide variety of bacteria. They might form the foundation for a brand-new category of antibiotics [3].

The problem of computationally discovering and synthesising AMPs has been attacked from several directions. Machine learning classifiers such as chance forests, provision vector machines, also neural networks have been employed by researchers at a variety of institutions to detect these peptides [6]-[10]. Researchers are seeking to replicate the features of a training set of AMPs by generating new instances of AMPs using generative models of these peptides [11]. Subsequence identification and labelling is an exciting field of research. There is evidence that several of the amino acids in use today can inhibit bacterial growth [12]. While high-quality datasets are required for training machine learning procedures, there is presently no standard dataset against which maximum models are being evaluated. Information and examples of antimicrobial peptides can be found in a number of databases [13, 14]. It is challenging to build a consistent dataset that draws from all of these sources because different databases have chosen to emphasise different properties of the peptides. For the purposes of either microbial peptide discovery or development, no uniform dataset has been created to train machine learning algorithms.

II. METHODS AND IMPLEMENTATION

To identify whether or not a peptide is antimicrobial, we employ a recurrent neural network equipped through a LSTMemory , a variant of Random Forests , and a k-Nearest Neighbours algorithm. The same dataset containing peptides with varied degrees of antibacterial activity and those that are considered innocuous to bacteria is used to test the efficacy of each method.

A. Dataset Construction

Results-oriented antibacterial examples were selected from the Database of Antimicrobial Movement also Construction of Peptides [14]. More than a hundred different bacterial species are covered by the peptides stored in this database. Peptides were chosen from this database because of their antibacterial activity, their lack of D- or non-proteinogenic amino acids, and their length (more than three but less than fifty-five). UniProtKB and Swiss-Prot were used as negative examples [15] from the Universal Protein Resource (UniProt). The database was searched for functional peptides ranging in size from 3 to 55 amino acids with no references to antibacterial activity or secretion. In order to reduce the size of the dataset, we first created a training set with both positive and negative examples, and then we removed any sequences that did not contain one of the 20 proteinogenic amino acids found in the normal genetic code. Protein-protein BLAST+ is one method for doing comparisons; it rules out sequences containing only three amino acids. There just weren't enough peptides in the DBAASP database that were 55 amino acids or longer to warrant not having a limit. This ensured that there would be a similar number of good and bad records in the set. All of the required peptide sequences were compiled into a single file using Biopython .

Protein-protein BLAST+ was used to create a second dataset using the positive and negative instances to reduce peptide sequence similarity. Peptides are deleted from the dataset until all remaining peptide pairs have sequence similarity below a threshold. Because all classification methods predict similarity, increasing sample dissimilarity would make categorization harder. All classification methods predict using similarity. A smaller dataset was created to test the methods.

B. Peptide Representations

In this study, we employed four distinct representations for peptide features. The latter, a pairwise similarity measure, is essential to the k-nearest neighbour (kNN) method. The initial representation was built using the ProtDCal software, and it consisted of the same amount of features for each peptide. The analysis relies on the 45,494 attributes provided by ProtDCal. Individual residue traits such as hydrophobicity and electrical charge are used to drive the evolution of these properties. They are reinforced by the local and global context of the amino acids around them. A peptide-wide feature can be derived after the groups have been generated using an operation such as averaging or summing the features of the groups. A null result is returned by the programme when a peptide lacks all of the values for a given category. After excluding topographies that consistently repaid null for altogether peptides popular the sample, only 45,378 were left. After applying scaling and normalisation, any residual values that were not zero were set to 0. The second representation takes as input a peptide's amino acid sequence and outputs a list of vectors with finite equivalent lengths, each of which represents a single amino acid. The overall number of finite length vectors fluctuates since the peptides in this collection span the gamut from extremely short to extremely lengthy. For this purpose, we employ the 20 one-hot vectors, 40 substitution matrices.

As a last step, we enrich the vector illustration of respectively amino acid with features known to be present in the secondary structure. The likelihood that a given amino acid in a peptide receipts on a given minor construction is represented here by one of eight attributes. We used a programme written by Wang et al. to estimate the secondary structure of all peptides and then ran it through a series of computations to find out what properties their predicted secondary structures have. Eight dissimilar secondary structures, such as alpha-helices, beta-strands, and coils, are used. The NCBI protein-protein BLAST+ database is one of the four feature representations utilised by kNN classification. Since NCBI protein-protein

BLAST+ distances are used to express these features, the resulting representation is inherently a remoteness matrix amongst all of the peptides popular the dataset. All of the above-mentioned feature representations, with the exception of predicted secondary structure characteristics, are created on protein amino acid arrangements also convey roughly the same information. Since the techniques under consideration require quite different feature representations, we primarily use a large number of feature representations. In contrast to Random Forest and kNN, which only require features at the peptide level, the LSTM framework requires characteristics at the level of individual amino acid residues.

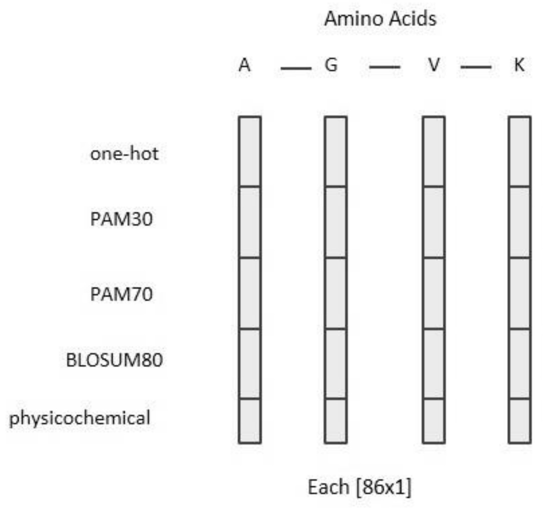

Figure 1. Feature Representation.

C. A Method for Generating a Dataset of Minimal Sequence Similarities

To compile the new data set, an algorithm was utilised to find peptides that were significantly dissimilar to one another. Let's begin by classifying all of the peptides in the initial dataset into group A. Let's start by incorporating a piece of set A into the peptides of our choice, set K. Also, this peptide meets the criteria for non-A inclusion. Examine the similarity between the recently added set K peptide and the remaining set A peptides. Set A peptides that are highly similar to set E peptides will be removed from Set A and added to Set E, with Set E not being included in Dataset 2. Compare each peptide in set A to the maximum recent addition in set K until you reach a conclusion about whether or not it belongs in set E. We are aware that the peptides we add to set K shouldn't share too much sequence similarity with the peptides already present in set K.

D. Implementations of Classification Algorithms

In order to create outputs for each input node in a sequence, recurrent neural networks (RNNs) rely on a hidden layer representation. Like a feedback loop, the representations stored in a recurrent neural network's hidden layers help the network predict what will happen next. A softmax layer is located at the very last stage of this feedforward network. This layer is responsible for estimating the probability that the peptide in question is a member of a specific category. One is able to make an estimate of the overall quantity of data that is lost inside a network by using these probabilities as a baseline.

Making use of current observations Those RNN that have a recurrent hidden layer unit are a special case. In contrast to traditional recurrent neural networks, which only make use of a single recurrent weight matrix and a single input heaviness matrix, LSTM wedges make use of numerous weight matrices . One of these matrices controls the data-receiving and data-transmitting gates in each memory block. Information transmission along a sequence can be modified with the help of these supplemental weights and the on-board memory . The vector formulas (1)–(6) in reflect forward pass events for a unidirectional LSTM. These calculations make use of the following inputs: the peephole weight vector p, the bias vector b, the rectangular input matrices W and R, and the input path at location s in the sequence, indicated by xs. The hyperbolic tangent function (h) and the logistic sigmoidal purpose (g) are two examples of non-linear activation functions at a single point. The multiplication of two vectors is shown here in a point-by-point manner.

$$z^{s} = g(W_{z}x^{s} + Rzy^{s-1} + b_{z}) \qquad (1)$$

$$i^{s} = \sigma(Wix^{s} + Riy^{s-1} + pi\ \ c^{s-1} + b_{i}) \qquad (2)$$

As can be seen in Fig. 2, In order to create a larger vector, a feedforward network combines its starting and final outputs. Data can flow in both ways along the peptide's backbone in bidirectional recurrent neural networks, which prevents the network from favouring either the C- or N-terminus. Importance because it demonstrates that the direction of antibacterial properties is irrelevant. For instance, the location of the antibacterial effect (such as interacting with and breaking the bacterial membrane) inside the peptide, whether at the C- or N-terminus, is irrelevant. In the unidirectional configuration, the equations employed by each LSTM block are different from those used in the bidirectional mode TensorFlow, an open-source numerical tool for making and training a wide variety of common neural network topologies, is used to construct the bidirectional LSTM.

The Random Forests technique was used to model the facets of ProtDCa that were of interest to us. In particular, we exploited the RF functionality available in scikit-learn [29 The kNN method is included in scikit-learn, a machine learning library. NCBI protein-protein BLAST+ bit-scores were used to generate a distance matrix that was given into the kNN algorithm. Aligned residue matches and mismatches are assigned a cost, which is then used to derive the bit-score [30]. Adjustments must be made to this method before it can withstand changes in database size and query order length.

III. RESULTS AND DISCUSSION

We refer to the first dataset, which includes 5779 peptides, as the "original dataset." There were probably not antimicrobial peptides (3,170) and antimicrobial peptides (2,609) present. By removing all pairs of peptides in the first dataset that shared a bit-score of 17 or higher in sequence similarity or higher, we were able to construct the second dataset. In total, there are 2475 peptides in the second collection, with 565 being judged positive and 1910 negative. Using stratified splits, we divided the first dataset into a test set and a training set, with the test set including 20% of the data and the training set comprising 80% of the data, in order to calculate hyperparameters using 5-fold cross-validation. Sequence similarity was lower for peptides in the second dataset, requiring the use of stratified splits for analysis.

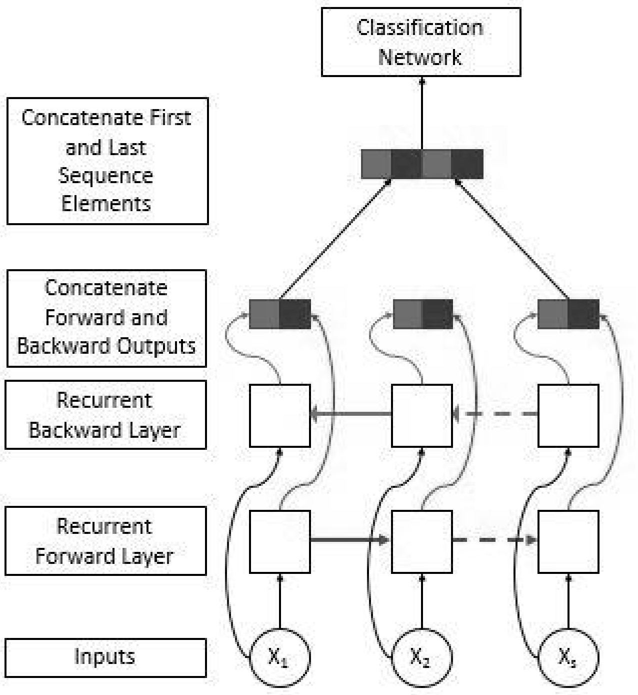
Classification Network
Concatenate First and Last Sequence Elements
Concatenate Forward and Backward Outputs
Recurrent Backward Layer
Recurrent Forward Layer
Inputs
X₁
X₂
X₅

Figure 2: Recurrent neural network with bidirectional LSTM topology.

The amino acid sequence and other peptide characteristics were encoded in a bidirectional LSTM recurrent neural network. That's because every possible amino acid combination was considered by the neural network, in both the forward and reverse directions. Previous work has employed protein sequences and bidirectional LSTMs . Each of the 86 amino acids in the sequence contributed to the maturation of a certain trait. A 1024-length vector was used to represent the 512 LSTM hidden units found in the bidirectional hidden layer. Figure 2 is a sketch of this structure's layout. The fundamental topology of LSTMs did not change whether they were trained on the complete or reduced similarity datasets with or without secondary structural variables. For our implementation of a bidirectional LSTM.

Manual The hyperparameters of the LSTM network were adjusted by means of 5-fold cross-validation. Adjustable parameters included the total number of hidden units, the pace of learning, the size of training batches, and the total number of training iterations. Cross-validation is used to determine which factors are most predictive before final selection is made. Adam, an adaptive momentum optimizer with a 0.001 epoch learning rate, was utilised. There were a total of 55 iterations used to train the final network, and the batch size was 128. In addition, the network did not make use of the peepholes that are a part of the default installation of Tensorflow. Because of the presence of p-containing words, Equations (2), (3), and (5) are all irrelevant.

The LSTM was taught using a dataset including amino acid sequences. With the exception of the NNAA Index factors, the input used for respectively feature crosswise all sequence components in the exercise set was mean-centered also scaled such that the average input for each feature has mean zero and unit variance. We used 8 non-scalar features in our LSTM architectures to account for the discontinuous probability distribution of the expected secondary structure types. The secondary structure of the LSTMs was built using features from 94 different amino acids.

Training the LSTM on the raw information without any projected subordinate structure resulted in a fairly good performance, as shown in Table I, with an accuracy of 95.74% and a Matthews correlation coefficient of.892. When minor structure prediction be located included to the feature representation of respectively amino acid in the peptide, accuracy rose from 94.64% to 94.98% and MCC rose from.892 to.899. These results demonstrate that the arrangement of amino acid feature vectors used in the classification task was efficiently summarised by the concatenation layer LSTM peptide representation.
The MCC of the LSTM decreases from.899 with secondary structure to.827 without it when applied to the smaller sequence similarity dataset. This trend persists regardless of the approach taken. Incorporating secondary structure into the LSTM does not improve accuracy, and both variants of the LSTM suffer when applied to the dataset with decreasing sequence similarity. Two tables, Tables I and II, detail the results.

TABLE I. EVALUATION OF ALGORITHMS USING RAW DATA

Algorithm	Accuracy	MCC
kNN	92.88%	.944
RF	95.39%	.896
LSTM	95.74%	.899
LSTM with Secondary Structure	95.99%	.901

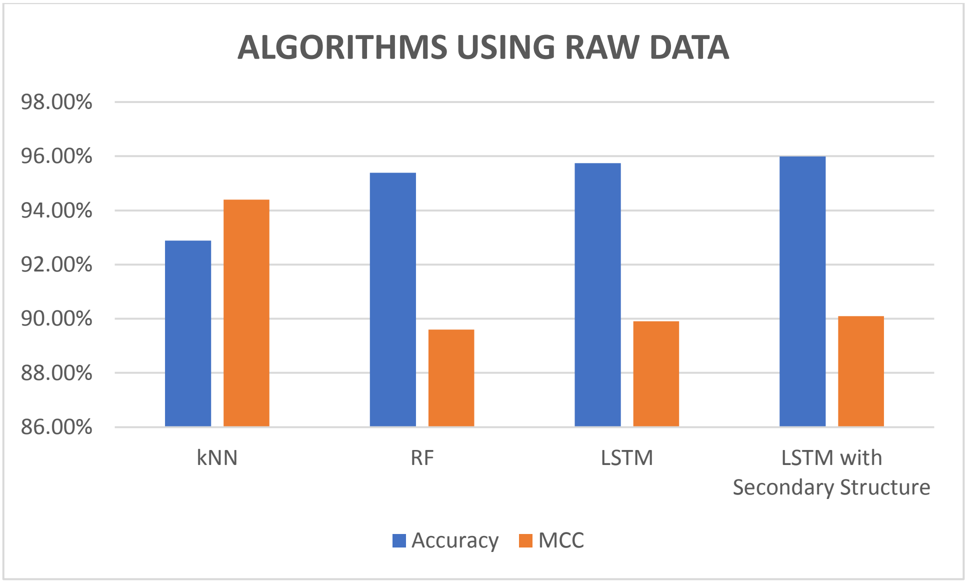

Figure 3: Comparison of Algorithms Using Raw Data

TABLE II. EFFICACY OF ALGORITHM USING SHARED SIMILARITY INFORMATION FOR REDUCED SEQUENCES

Algorithm	*Accuracy*	*MCC*
kNN	87.88%	.587
RF	93.34%	.790

LSTM	94.74%	.834
LSTM with Secondary Structure	94.95%	.837

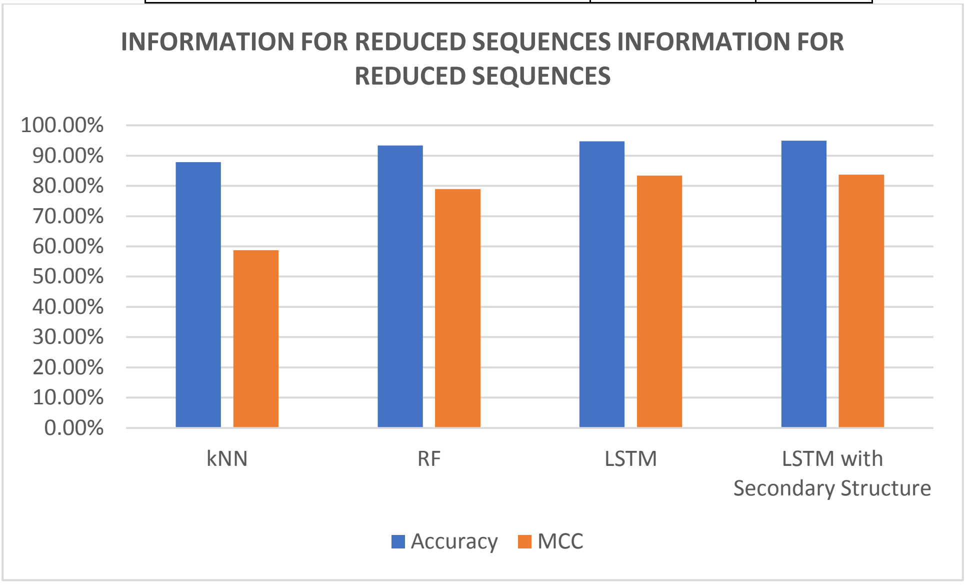

Figure 3: Comparison Information for Reduced Sequences

A. *Random Forest Outcomes*

The RF algorithm nearly matched the effectiveness of the LSTM on the benchmark dataset. Results showing a high MCC and accuracy for the RF technique (94.29%) are shown in Table I. When compared to LSTM algorithms, performance on the shorter sequence similarity dataset was worse, notably in terms of MCC. The machine calculated an MCC score of.780 based on the smaller dataset.

The best possible values for the RF parameters were uncovered through the use of five-fold cross validation. When running cross-validation on the initial dataset, the RF that performed the best did so by making use of Gini impurity and the square root of the total number of features at each split. Additionally, there should have been three samples taken from each leaf node in the tree. A grand number of 512 trees was tallied up in the woods during the survey. The information gain criterion and the square root of the number of features per split were used in the RF that achieved the best results in the cross-validation test conducted on the dataset containing reduced sequence similarities. At least three leaves from each of the 128 decision trees need to be taken into consideration.

B. *KNN Outcomes*

We were able to quantify the impact that sequence similarity has on the efficacy of machine learning techniques. This allowed us to label the test set with the same information that was used to label the training set. By doing so, we may compare the efficacy of algorithms that use physicochemical descriptors to those that rely solely on sequence similarity.

Using five-fold cross-validation, the kNN algorithm was fine-tuned. k (the number of nearest neighbours) could be anything from 1 to 20, depending on the circumstances. Metrics for peptide similarity or distance cannot be generated by NCBI protein-protein BLAST+ without the E-value threshold. Alignment information is not given for peptide pairs with an E-value above this cutoff. The E-value [30] of an alignment is a statistical assessment of how likely it is that the alignment was discovered accidentally during a database search. With the goal of learning as much as possible about peptide separation, the E-value cutoff was made quite high (1030). The bit-score cares only about the raw score, or the total cost of aligned and misaligned residues, regardless of the size of the database or the query sequence. When constructing protein-protein BLAST+ calls [16, 30], we selected bit-score due to its greater accuracy compared to other metrics of similarity. kNN had a 91.78% classification accuracy on test data with an MCC of 0.834 percent, however random forests and the LSTM were superior. With an MCC of.597 and accuracy of.8687% on the test set, the validation findings imply that a configuration of two neighbours is optimal for the decreasing similarity dataset.

C. *General Outcome*

The final test sets for all three methods on the raw data are shown in Table I. They demonstrate that LSTM is more accurate than RF and kNN approaches, even though the difference between the two is supposed to be minimal, at roughly 1% on the same test set. It's possible that the kNN algorithm is ineffective because it overemphasises sequence similarities. The LSTM and RF's success might probably be attributed to their incorporation of other criteria, such as physicochemical properties, that are essential for recognising antibacterial peptides. Table I further shows that adding features predicted by the secondary structure of the algorithm to the LSTM did not significantly improve its performance.

IV. CONCLUSION

Building LSTM features, which are constrained to the amino acid level, is simpler than building RF classifier features. The RF classifier's characteristics are transferable to the LSTM. The RF feature could be used more than once for each amino acid feature vector in a peptide to ensure a continuous input to the LSTM. This quality would be present in every peptide, but it wouldn't be the same in any two. The decreased similarity dataset demonstrates that LSTM algorithms are less reactive to the updated data. Since the new dataset employs sequence similarity to establish classifications, the kNN should have a harder time with it. Seeing the gulf grow between LSTM algorithms and RF as one moves from the whole dataset to the reduced similarity data is amazing. It's possible that the LSTM relies less on sequence similarity than the RF does. The importance of the quality of the dataset's negative cases was briefly touched on in the introduction. Negative cases that have been verified experimentally using measurements like minimum inhibitory concentration (MIC) are rare in antibacterial peptide discovery databases. Weak MIC readings against a sufficient number of bacterial pathogens are necessary to confidently include this peptide in a set of negative cases for antibacterial versus nonantibacterial. Swiss-Prot peptides that have been studied extensively but have no evidence of antibacterial or secretory activity are included as counterexamples. A huge dataset of MIC values for both positive and negative samples could one day be used to train organism-specific classifiers. This investigation used the DBAASP database to identify peptides that showed promise in killing off pathogens like E. coli. As a result of the provided examples, the number of negative cases will increase while the number of positive cases will drop.It is still quite unlikely that a negative MIC measurement would be discovered. Next-generation research will concentrate on generative models that can synthesise antimicrobial peptides from a predetermined distribution of amino acids. By offering a large number of candidate peptides for testing against a predetermined collection of bacterial strains, such approaches have the potential to speed up the identification of novel antibacterial peptides. A generative model might be used to generate candidate peptides, and a discriminative model could be used to determine whether or not they are effective against a given bacterial species.

REFERENCES

1. R. Laxminarayan, A. Duse, C. Wattal, A. K. M. Zaidi, H. F. L. Wertheim, N. Sumpradit, et al., "Antibiotic resistancethe need for global solutions", *The Lancet Infectious Diseases*, vol. 13, no. 12, pp. 1057-1098, 2022.
2. W. C. Wimley and K. Hristova, "Antimicrobial peptides: successes challenges and unanswered questions", *The Journal of membrane biology*, vol. 239, no. 1–2, pp. 27-34, 2021.
3. M. Pasupuleti, A. Schmidtchen and M. Malmsten, "Antimicrobial peptides: key components of the innate immune system", *Critical reviews in biotechnology*, vol. 32, no. 2, pp. 143-171, 2020.
4. B. Deslouches, J. D. Steckbeck, J. K. Craigo, Y. Doi, T. A. Mietzner and R. C. Montelaro, "Rational design of engineered cationic antimicrobial peptides consisting exclusively of arginine and tryptophan and their activity against multidrug-resistant pathogens", *Antimicrobial Agents and Chemotherapy*, vol. 57, no. 6, pp. 2511-2521, 2013.
5. S. Lata, N. K. Mishra and G. P. Raghava, "Antibp2: improved version of antibacterial peptide prediction", *BMC Bioinformatics*, vol. 11, no. 1, pp. 1-7, 2010.
6. C. Fernandes, D. J. Rigden and O. L. Franco, "Prediction of antimicrobial peptides based on the adaptive neuro-fuzzy inference system application", *Peptide Science*, vol. 98, no. 4, pp. 280-287, 2012.
7. S. Joseph, S. Karnik, P. Nilawe, V. K. Jayaraman and S. Idicula-Thomas, "Classamp: A prediction tool for classification of antimicrobial peptides", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 5, pp. 1535-1538, Sept 2012.
8. Y. Lee, B. M. Fulan, G. C. L. Wong and A. L. Ferguson, "Mapping membrane activity in undiscovered peptide sequence space using machine learning", *Proceedings of the National Academy of Sciences*, vol. 113, no. 48, pp. 13588-13593, 2016.
9. S. Gigure, F. Laviolette, M. Marchand, D. Tremblay, S. Moineau, X. Liang, et al., "Machine learning assisted design of highly active peptides for drug discovery", *PLOS Computational Biology*, vol. 11, no. 4, pp. 1-21, 04 2015.
10. K. Y. Chang, T.-p. Lin, L.-Y. Shih and C.-K. Wang, "Analysis and prediction of the critical regions of antimicrobial peptides based on conditional random fields", *PLOS ONE*, vol. 10, no. 3, pp. 1-16, 03 2015.
11. H. Waghu, R. S. Barai, P. Gurung and S. Idicula-Thomas, "Campr3: a database on sequences structures and signatures of antimicrobial peptides", *Nucleic Acids Research*, vol. 44, no. D1, pp. D1094-D1097, 2016.
12. M. Pirtskhalava, A. Gabrielian, P. Cruz, H. L. Griggs, R. B. Squires, D. E. Hurt, et al., "Dbaasp v.2: an enhanced database of structure and antimicrobial/cytotoxic activity of natural and synthetic peptides", *Nucleic Acids Research*, vol. 44, no. D1, pp. D1104-D1112, 2016.

13. Boutet, D. Lieberherr, M. Tognolli, M. Schneider, P. Bansal, A. J. Bridge, et al., "UniProtKB/Swiss-Prot the Manually Annotated Section of the UniProt KnowledgeBase: How to Use the Entry View", *Methods Mol. Biol.*, vol. 1374, pp. 23-54, 2016.
14. Y. B. Ruiz-Blanco, W. Paz, J. Green and Y. Marrero-Ponce, "Protdcal: A program to compute general-purpose-numerical descriptors for sequences and 3d-structures of proteins", *BMC Bioinformatics*, vol. 16, no. 1, pp. 162, May 2015.
15. C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, et al., "BLAST+: architecture and applications", *BMC Bioinformatics*, vol. 10, pp. 421, Dec 2009.