ANTIMICROBIAL PEPTIDE HOST TOXICITY PREDICTION WITH TRANSFER LEARNING FOR PROTEINS

by

Figali Taho

B.Sc., The University of British Columbia, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Bioinformatics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, a thesis entitled:

Antimicrobial peptide host toxicity prediction with transfer learning for proteins

submitted by Figali Taho in partial fulfillment of the requirements for

the degree of Master of Science

in Bioinformatics

Examining Committee:

Dr. Inanç Birol, Professor, Department of Medical Genetics, UBC
Supervisor

Dr. Steven Jones, Professor, Department of Medical Genetics, UBC
Supervisory Committee Member

Dr. Andrew Roth, Assistant Professor, Department of Pathology & Laboratory Medicine, Department of Computer Science, UBC
Supervisory Committee Member

Dr. Faraz Hach, Assistant Professor, Department of Urologic Sciences, UBC
Additional Examiner

Abstract

Antimicrobial peptides (AMPs) are host defense peptides produced by all multicellular organisms, and can be used as alternative therapeutics in peptide-based drug discovery. In large peptide discovery and validation pipelines, it is important to avoid time and resource sinks that arise due to the necessity of experimentally validating a large number of peptides for toxicity. Therefore, *in silico* methods the prediction of antimicrobial peptide toxicity can be applied in advance to filter out any sequences that may be of toxic nature. While many machine learning-based approaches exist for predicting toxicity of proteins, it is often defined as a problem of classifying venoms and toxins from proteins that are nonvenomous.

In my thesis I propose a new method called tAMPer that focuses on the classification of AMPs that may or may not induce host toxicity based on their sequences. I have used deep learning model ELMo as adapted by SeqVec to obtain vector embeddings for a dataset of synthetic and natural AMPs that have been experimentally tested *in vitro* for their toxicity through hemolytic and cytotoxicity assays. This is a balanced dataset that contains ~2600 sequences, split 80/20 into train and test set. By utilizing the latent representation of the data by SeqVec, and by further applying ensemble learning methods on these embeddings I have built a model that is capable of predicting toxicity of antimicrobial peptides with a F1 score of 0.758 and accuracy of 0.811 on the test set, and performing comparably better than state-of-the-art approaches both when trained and tested on our dataset as well as on other methods' respective train and test datasets.

Lay Summary

Antibiotic resistance poses a global health concern that prompts scientific exploration of alternative therapeutics. Antimicrobial peptides (AMPs) are produced by all multicellular organisms and provide natural defense against pathogens. In order to avoid potential resource and time sinks due to the necessity of screening many peptides for toxicity against host organisms, it is important to use computational methods to prioritize and filter through large amounts of AMPs for synthesis and testing. Although many approaches were developed in the last decade to predict toxicity of proteins, the current state-of-the-art methods are outdated and cannot predict AMP host toxicity. I describe in my thesis a robust and competitive approach that uses machine learning algorithms to predict potential host toxicity of AMPs based on the peptide sequences.

Preface

The concept and design of my research was conceived by my supervisor, Dr. Inanç Birol, and myself. All software programming, model development and analysis were carried out by myself with very helpful discussions with Mr. Chenkai Li and Mr. Darcy Sutherland.

Table of Contents

Abstract ..... **iii**

Lay Summary ..... **iv**

Preface ..... **v**

Table of Contents ..... **vi**

List of Tables ..... **ix**

List of Figures ..... **x**

List of Abbreviations ..... **xii**

Acknowledgements ..... **xiii**

Dedication ..... **xiv**

Chapter 1: Introduction **1**

1.1 AMP activity 2

1.2 AMP host toxicity 3

1.3 Protein toxicity prediction 4

1.4 Phylogeny-based prediction of host toxicity 6

1.5 Inspiration for embeddings in sequence classification tasks 7

1.5.1 Activation functions 8

1.5.2 Recurrent neural network (RNN) 9

1.5.3 LSTM 10

1.6 Thesis objectives 11

Chapter 2: Methods **12**

2.1 Dataset collection and curation 12

2.2 Protein embedding methods 15
2.2.1 SeqVec embedding representations 15
2.3 Classification models 17
2.3.1 Base classifiers 18
2.3.1.1 Random Forest 18
2.3.1.2 Gradient Boosting 19
2.3.1.3 Logistic Regression 19
2.3.2 Stacking ensemble 20
2.3.3 Sequential LSTM model 21
2.3.3.1 Bi-LSTM model architecture 23
2.3.3.2 Optimization and loss function 23
2.3.4 Final ensemble 24
2.3.5 Implementation details 24
2.4 tAMPer ensemble model evaluation 25
2.4.1 Input representation evaluation 25
2.4.1.1 One hot encoding representation 25
2.4.1.2 Other baseline compositions 26
2.4.1.3 UniRep embedding representation 27
2.4.1.4 fastText embedding representation 28
2.4.2 Comparison with state-of-the-art 28
2.4.3 Biological relevance of using tAMPer 29
Chapter 3: Results 31
3.1 Input representation results 31

3.2 Comparison with the state-of-the-art 34
3.2.1 Performance comparison setup 35
3.2.1.1 HemoPI 35
3.2.1.2 ToxinPred 36
3.2.1.3 TOXIFY 36
3.2.1.4 Additional considerations 36
3.2.2 tAMPer and TOXIFY trained on tAMPer training set 38
3.2.3 tAMPer and TOXIFY trained on other methods' training sets 42
3.3 Biological relevance of using tAMPer 45
3.4 tAMPer ensemble score interpretation 52
Chapter 4: Discussion 53
Bibliography 55

List of Tables

Table 1: Comparison of current methods for protein toxicity prediction. 6

Table 2: Overview of different database sources for antimicrobial peptides. 12

Table 3: Previously reported and tested AMPs and corresponding mutations, where the residues highlighted in red are the ones that are mutated. 29

Table 4: P-values for paired Student t-test of each representation method compared with SeqVec input representation. 32

Table 5: Overview of training and testing setup for comparisons with existing methods. 35

Table 6: Performance of different models on the tAMPer test set. 39

Table 7: Performance of all models on the HemoPI test set. 40

Table 8: Performance of all models on the ToxinPred test set. 40

Table 9: Performance of all models on the TOXIFY test set. 40

Table 10: Performance of models on the test sets of the methods whose training sets were used for training. 42

Table 11: Performance of all re-trained tAMPer and TOXIFY models on the tAMPer test set. . 44

List of Figures

Figure 1: Schematic high-level overview of direct interactions of AMPs with cells. 2

Figure 2: Phylogenetic tree for a subset of the AMP dataset used for analysis. 7

Figure 3: Various non-linear activation functions used in the sequential model. 9

Figure 4: Detailed internals of an LSTM cell. 10

Figure 5: ELMo-based SeqVec model schematic. 16

Figure 6: Per protein and per residue embeddings for an example peptide. 16

Figure 7: Overview of tAMPer model. 17

Figure 8: Stacking ensemble model training overview 21

Figure 9: Overview of the Bi-LSTM sequential model used for prediction. 22

Figure 10: One hot encoding representation for an example AMP. 26

Figure 11: Physicochemical features as representation of sequences. 27

Figure 12: Comparison of all input representations considered using the three base models' ensemble. 33

Figure 13: Comparison of SeqVec embedding models with the one hot encoding feature baseline. 34

Figure 14: Size of the train and test sets for the different methods. 37

Figure 15: Box plot of the length distribution of the different methods' positive and negative training sets. 38

Figure 16: ROC plots performance of different models on the available test sets. 41

Figure 17: ROC plots performance of re-trained and static models on the test sets of the different methods. 43

Figure 18: ROC plot of performance of all re-trained TOXIFY and tAMPer models on the tAMPer test set. 44

Figure 19: Example heatmap of P2_WT for showing biological significance of tAMPer. 47

Figure 20: All WT AMP heatmaps (in order left to right, top to bottom, as P2-P6, P11). 48

Figure 21: Heatmap for P2_WT and P2_M1. 49

Figure 22: Heatmap for P3_WT and P3_M1. 49

Figure 23: Heatmap for P4_WT and P4_M1. 50

Figure 24: Heatmap for P5_WT and P5_M1. 50

Figure 25: Heatmap for P6_WT and P6_M1. 51

Figure 26: Heatmap for P11_WT and P11_M1. 51

Figure 27: Toxicity concentration (uM, log scale) vs. tAMPer ensemble score. 52

List of Abbreviations

AAcomp	Amino acid composition
AMP	Antimicrobial peptide
MDR	Multi-drug resistant
MIC	Minimum inhibitory concentration
NLP	Natural Language Processing
Phys-chem	Physicochemical feature representation
ROC	Receiver Operating Characteristic

Acknowledgements

First, I would like to thank Dr. Inanç Birol for providing me the opportunity to develop my research skills and communication skills as a graduate student under his guidance. I would also like to thank him for his patience, his constructive criticisms, for regularly encouraging me and for providing me many opportunities to present my work at various international conferences.

I would like to thank my thesis committee members, Dr. Steven Jones and Dr. Andrew Roth, for their valuable feedbacks and time throughout my degree.

I want to express my sincerest gratitude to the Natural Sciences and Engineering Research Council of Canada and the Institute of Health Research, for providing financial support for my thesis.

I would like to thank all of the wonderful members of the Bioinformatics Technology Lab at the Genome Sciences Centre, especially Mr. Chenkai Li and Mr. Darcy Sutherland for their valuable discussions on various aspects of my research.

I am also very thankful to have made great friends in the Bioinformatics program, that helped me better my communication skills and research.

Lastly, I would like to thank my parents, my brother, and my partner for their kind words, encouragement and unconditional support for me throughout my graduate studies.

Dedication

I dedicate this thesis to my family and my partner, for their unconditional love and support.

Chapter 1: Introduction

Clinical isolates of many pathogenic bacterial species have been able to develop resistance to antibiotics discovered during the 1930s to 1960s, and this poses an imminent global threat to human life [1, 2]. Deaths as a result of infections caused by antibiotic-resistant organisms take the lives of more than 35,000 people in the U.S. each year, and it is estimated that more than 2.8 million antibiotic-resistant infections occur in the U.S. each year [3]. For this reason, scientists are turning their attention to alternative antimicrobial agents that provide defense and induce lower or no resistance to pathogens.

In the constant struggle for survival of life on Earth, even the tiniest of organisms such as bacteria or viruses undergo physical adaptations aimed at maintaining species survival. Antimicrobial peptides (AMPs), also called host defense peptides, are a class of oligopeptides that are part of the innate immune systems across all life. In animals they are believed to serve as the first line of defense, stopping infections before they cause any symptoms [4]. AMPs have a range of properties that makes them a strong candidate for AMP based drug design. Due to their short sequence and amino acid composition, it is easy to modify their structure, as well as make novel synthetic peptides that have desirable properties by means of chemical synthesis. Although the understanding of resistance mechanisms and degree of microbial resistance against AMPs is still quite limited mainly because of a lack of extensive analysis and data, certain AMPs are found to limit the evolution of resistance [5]. Additionally, the co-evolution of AMPs with their host microbiomes in animals suggests that their induced resistance is to a lower degree than conventional antibiotics [6]. As such, they present a strong case for being developed commercially and joining a novel class of therapeutics as arsenal against pathogens, provided thorough studying and testing upon them.

1.1 AMP activity

Antimicrobial peptides are short, typically in the range of 10-50 amino acid residues long. They usually display a positive charge between +2 and +11 and are amphipathic, meaning they are comprised of both hydrophilic and hydrophobic parts. Many AMPs display direct antimicrobial activity due to disruption of the physical integrity of the membrane or by translocating across the membrane and inhibiting intracellular targets. It is thought that differences in the composition of the microbial and mammalian membranes, mainly the contrast between negatively charged bacterial membranes versus neutral mammalian cell membranes, as well as the changes in lipid composition, are the cause for toxic selectivity of AMPs toward microbial cells and what protects the host mammalian cells from similar interactions [7, 8].

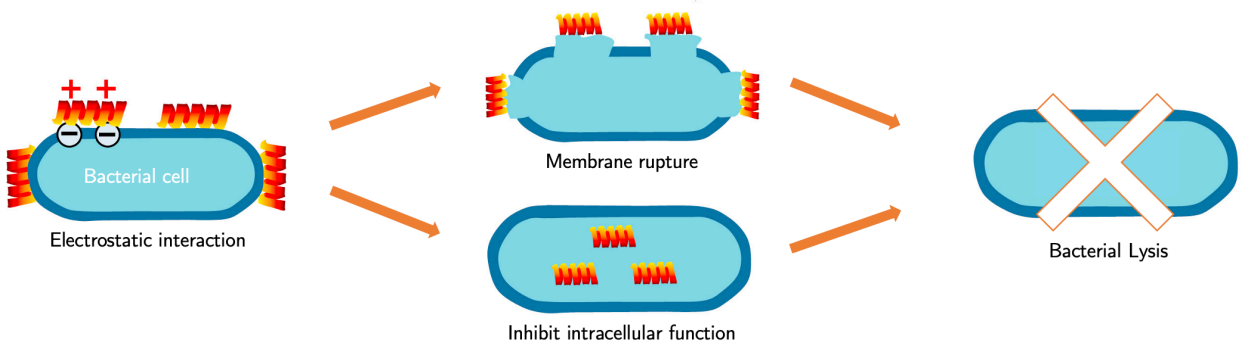

Figure 1: Schematic high-level overview of direct interactions of AMPs with cells.

Electrostatic interactions between the negatively charged bacterial cell membranes and the cationic AMPs causes the bacterial cell death, either through direct membrane rupture, or by membrane translocation and inhibition of intracellular functions like protein and nucleic acid synthesis. Figure adapted from [9].

AMPs have also been found to have other mechanisms of killing, e.g. they can modulate immune response by attracting leucocytes and modulating host-cell response to Toll-like receptor (TLR) ligands, or act as a proteolytic agent and stop protein and DNA synthesis [9].

In vitro trials are widely applied for analyzing the therapeutic potential of antimicrobial peptides and their direct activity against pathogens. Typically, in these experiments the inhibitory effect of culture growth for various pathogens is evaluated by measuring the Minimum Inhibitory Concentration (MIC) and the Minimum Bactericidal Concentrations (MBC). MIC is the lowest concentration of an antimicrobial agent that prevents the visible growth of bacteria, and MBC is the lowest concentration of an antibacterial agent required to kill a bacterium over a fixed period of time under certain conditions [10]. Ideally, the lower these values are, the more efficient an antimicrobial agent is in inhibiting or killing the tested pathogens. The increasing interest in studying AMPs has created resources of knowledge on naturally occurring or synthetic AMPs that have been utilized in the development of various *in silico* methods for distinguishing AMPs from non-AMPs [11-13]. Recognizing peptides that are potentially active against pathogens is a key step in any pipeline aimed at discovering and designing AMPs for immunity enhancing purposes.

1.2 AMP host toxicity

Optimal AMPs would be characterized both by high activity against pathogens and low toxicity towards host cells. Many AMPs have been shown to be highly effective against pathogens while simultaneously expressing hemolytic properties against red blood cells [14]. The hemolytic activity of AMPs is assessed *in vitro* by determining either the maximum concentration of peptide that results in no hemolysis (also known as MHC), or the concentration

of peptide (μg/mL) required for 50% hemolysis (HC_{50}) [14]. Similarly, CC_{50} for cytotoxicity assays refers to the concentration of antimicrobial (μg/mL) that is required for the reduction of cell viability by 50%.

The therapeutic index is used to characterize the efficacy of an antimicrobial peptide and it is defined as the ratio MHC/MIC. It is desirable that the MIC is as low as possible, while the MHC is as high as possible, which implies that the peptide is not hemolytic unless the concentration is high, and is very active against bacteria in small concentrations. Therefore, to achieve a high therapeutic index either the antimicrobial activity of the peptide should be increased, or the hemolytic activity should be decreased while maintaining activity high, or a combination of both [15].

The objective of my thesis is developing a robust *in silico* method for characterizing antimicrobial peptides' host toxicity, to enable cheaper and faster design optimization of AMPs for therapeutic use.

1.3 Protein toxicity prediction

Protein toxicity prediction is typically treated as a binary classification problem, where the positive set is composed of protein sequences that show toxic characteristics, and the negative set is composed of protein sequences of non-toxic proteins. Supervised learning approaches are employed for solving this task, with methods HemoPI [16] and ToxinPred [17] using Support Vector Machine [18] and the method TOXIFY [19] using a GRU-based recurrent network [20]. The choice of input representation also varies between these methods.

The most common representation of protein sequences is one-hot encoding, where each residue in the sequence is represented by a binary vector of size equal to the vocabulary (e.g. 20

amino acids in the protein space) filled with zeros except for at the index of the current amino acid residue. This creates a sparse matrix representation for each peptide sequence, with padded zero-vectors added to sequences in order to fix an input shape for training.

The amino acid composition, $Comp(i)$, of amino acid i is defined as the percent composition of amino acid i in a peptide: $Comp(i) = 100\frac{r_i}{n}$, where r_i is the number of residues of type i and n is the total number of residues in the peptide. These representations, as well as the dipeptide composition, defined as the frequency of pairs of neighboring amino acid residues in a peptide among all possible amino acid pairs ($20 \times 20 = 400$), have been used in both HemoPI and ToxinPred.

In contrast, TOXIFY uses the Atchley factors [21] for a per-amino-acid input representation [19]. Atchley factors summarize amino acids by five highly interpretable numerical values of amino acid attributes in the literature that reflect the physicochemical properties of polarity, secondary structure, molecular volume, codon diversity and electrostatic charge. Because the method admits sequences of length up to 500 residues, any sequence of shorter length in the original method has been padded with zeros, so the per-sequence input is a matrix $X \in \mathbb{R}^{5\times500}$.

Among these methods, HemoPI is the only one providing a model for classification of peptides trained on a (small) dataset of sequences labelled into two classes based on experimentally validated hemolytic potency values [22]. TOXIFY and ToxinPred focus on binary classification of venomous proteins from nonvenomous ones, and they are both trained on imbalanced data, with TOXIFY having a negative set of over six times bigger than the positive set, and ToxinPred similarly having a negative set almost twice as big as the positive set (Table 1). Neither of these available methods are specialized for AMP host toxicity prediction, which

creates a gap in the literature for the task. Availability of such a tool would be useful in designing better AMPs *in silico*.

Table 1: Comparison of current methods for protein toxicity prediction.

Model	Train	Server	AMP or VENOM	Method	Pos. set	Neg. set	Pub year	Last update	Ref.
TOXIFY	+	+	VENOM	GRU-NN	5884	38616	2019	2019	[19]
HemoPI	-	+	AMP	SVM	442	370	2016	2016	[16]
ToxinPred	-	+	VENOM	SVM	1784	3572	2013	2013	[17]

1.4 Phylogeny-based prediction of host toxicity

While the literature hints at the application of machine learning based methods for the AMP host toxicity prediction problem, it is important to evaluate whether the data space of this problem contains inherent information in the sequence that may be easily detectable with phylogeny-based approaches to AMP sequences. With a subset of the AMP data collected, labelled, and used throughout this thesis, a phylogenetic tree was built using multiple sequence alignment based on the MUSCLE algorithm [23], and the RAxML tool [24], a standard for maximum-likelihood based phylogenetic inference. The resulting phylogeny suggests that this data lacks an underlying similarity between the two categories of toxic and nontoxic labelled AMP sequences, as toxic and nontoxic categories do not present a clear separation (Figure 2). This encourages the exploration of more elaborate machine learning based classification methods.

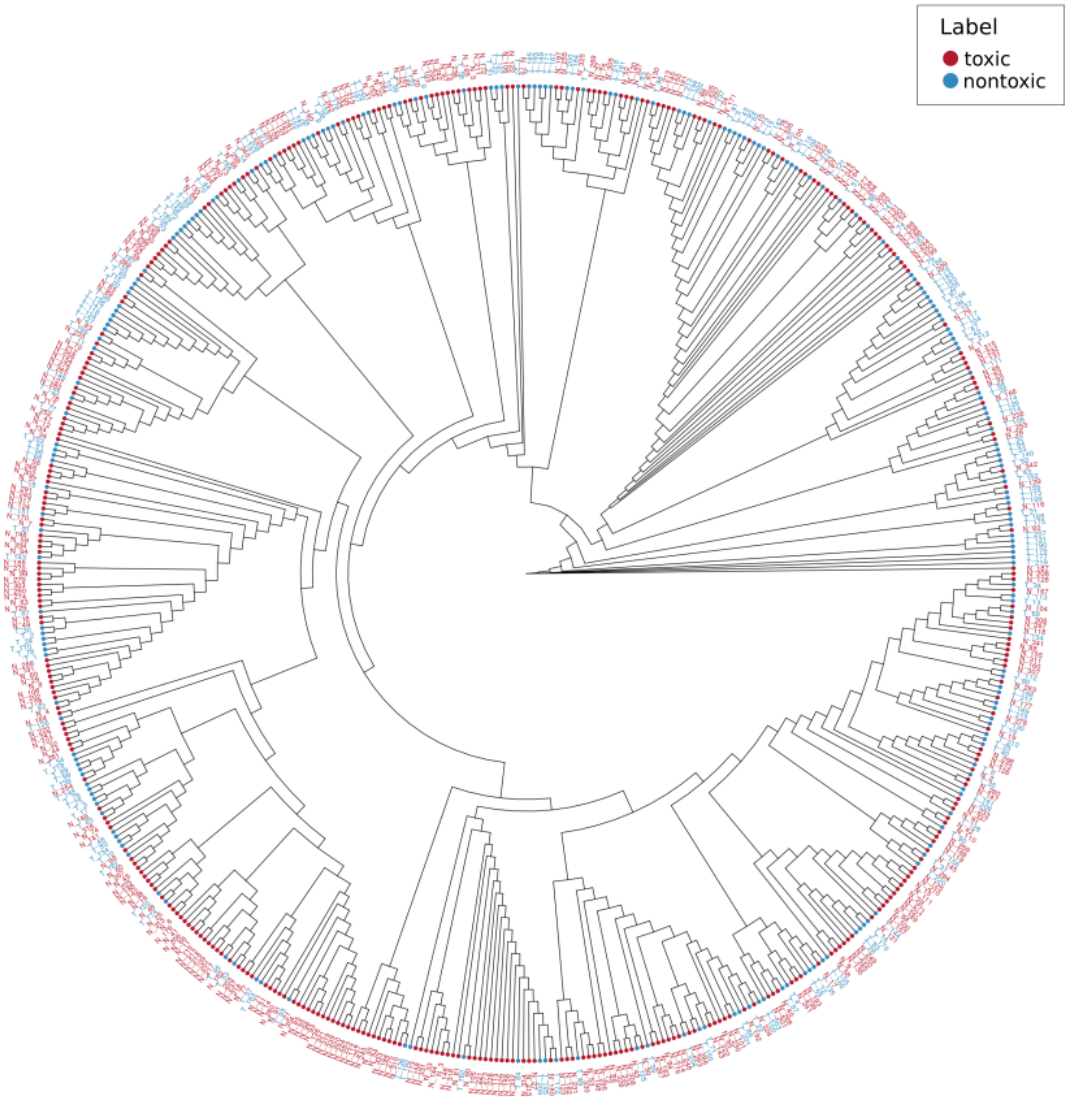

Figure 2: Phylogenetic tree for a subset of the AMP dataset used for analysis.

The data is labelled based on toxicity concentrations reported in the literature (described in 2.1). The phylogeny contains 220 peptides labelled as toxic and 342 peptides labelled as nontoxic.

1.5 Inspiration for embeddings in sequence classification tasks

Input sequence representation based on one-hot encoding suffers from the additional padding, and data sparsity. Also, both amino acid composition encodings described in Section

1.3 are very limited in their consideration of context of residue appearance in the peptide. Considering physicochemical properties for input representation, while more holistic, does require domain expertise, and makes the implicit assumption that the chosen features encode all necessary information, which may not always be the case.

Embedding methods have been extensively used in Natural Language Processing (NLP), making the training of large corpora of text a necessity for developing models that perform better in prediction tasks related to language, like sentiment analysis, translation, Part-Of-Speech tagging etc. [25, 26]. Many embedding methods have been adapted for protein prediction tasks, and the latent representations have been shown to encode information that is relevant to sequences and their biological functions [27]; such methods have been used under this premise for solving a wide range of protein prediction tasks.

Using embeddings of the input sequences eliminates the need for domain expertise in choosing sequence descriptors or features, enables transfer learning from large protein sequence datasets, as well as creating an effective per sequence vector input representation that simplifies downstream employment of classical machine learning algorithms. Embedding methods are based on sequential or recurrent neural network models, and the next sub-sections detail some necessary background for the topic.

1.5.1 Activation functions

Activation functions are nonlinear transformations of the weighted sum of inputs in the network units, and the ones used in this model are ReLU, $\mathrm{ReLU}(z) = \max(0, z)$ and "sigmoid", $\sigma(z) = \frac{1}{1+e^{-z}}$. Another activation function used in the LSTM unit is the tanh transformation

function $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. The graphical representations of these functions are given in Figure 3.

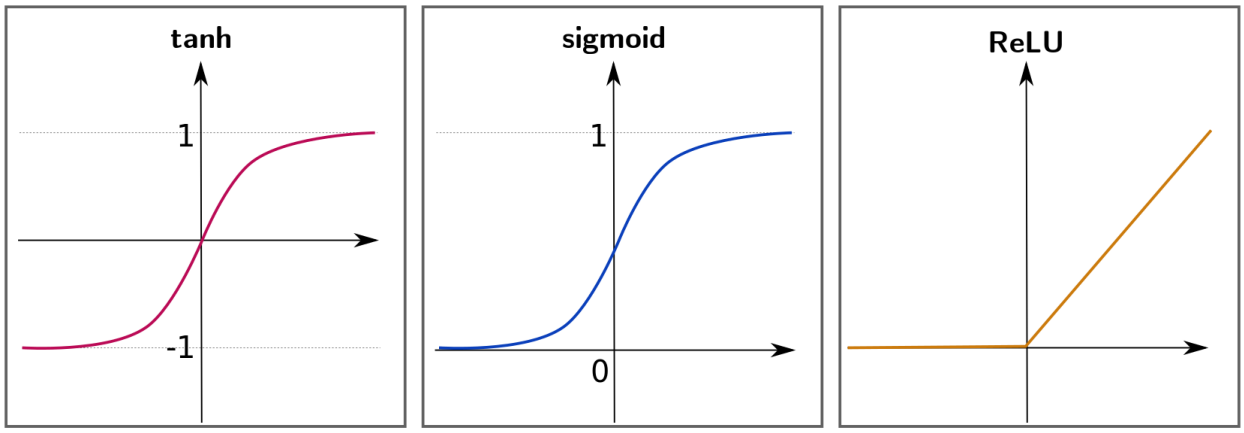

Figure 3: Various non-linear activation functions used in the sequential model.

1.5.2 Recurrent neural network (RNN)

The ability of RNNs [28] to process and model input of any length is the reason why their usage is ubiquitous in sequence and time-series related tasks. Additionally, the model does not increase in size with the size of its input – a property that is ideal for processing large datasets. RNNs are capable of learning temporal dynamics, by mapping the input to a sequence of hidden states, and mapping these states to outputs. Also, activations at a given hidden layer arrive both from the external input, and from the hidden layer activations of the previous timestep. The recurrence can be expressed as $h_t^l = \tanh W^l \binom{h_t^{l-1}}{h_{t-1}^l}$. The parameter W^l is the weight matrix shared through time for a layer l [29], and $t = 1 \ldots T$, where T is the timestep dimension.

On the other hand, RNNs suffer in practice from incapacity to hold long range information, especially when the gap between the prediction and the necessary context become

too large. RNNs also suffer from exploding or vanishing gradients during training and backpropagation through time in the computation graph [30]. GRU and LSTM were designed to address these issues.

1.5.3 LSTM

The LSTM computes not only hidden values, but also memory cells. The retention of long-range information is enabled via input, output, forget and update/memory gates. The formulas of the update are below.

$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ (forget gate)

$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ (input gate)

$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$ (new memory cell)

$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$ (final memory cell)

$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$ (output gate)

$h_t = o_t * \tanh(c_t)$ (hidden state computation)

Figure 4 visually represents the role of these gates.

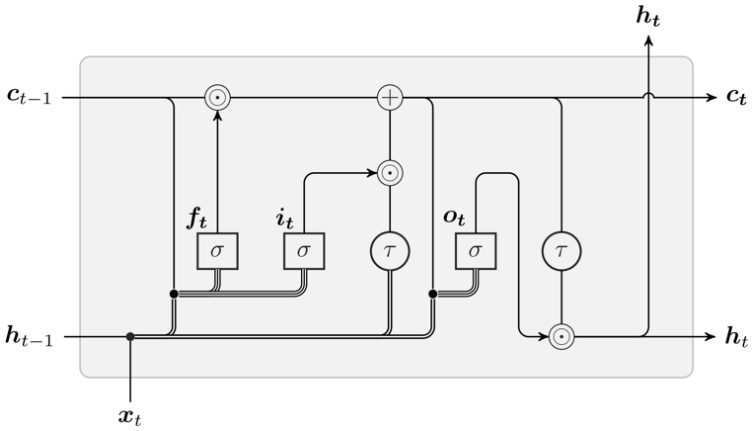

Figure 4: Detailed internals of an LSTM cell.

σ is the sigmoid activation, τ is the tanh activation function.
Image credits: Majid al-Dosari's comment in [31].

The forget gate decides what information to discard from the cell state based on previous cell state h_{t-1} and current input x_t, by outputting a number between 0 and 1 for each number in cell state c_{t-1}, such that, 0 gets rid of the input (forgets), while 1 retains it. Next, the cell state is computed. The sigmoid transformation for i_t decides the values to update. A new vector of candidate values is computed in $\tilde{c}_t$. These are combined and summed with the product of the old cell state to the output of the forget gate to compute the new cell state (final memory cell). Finally, the output o_t is calculated as the product of a sigmoid transformation of the previous hidden layer and this is used to compute the hidden state, as the product of the output (o_t) and a tanh transformation of the current cell state (c_t) (hidden state computation) [29, 32].

1.6 Thesis objectives

The increasing interest in AMP based therapeutics and the lack of an effective *in silico* approach for AMP host toxicity prediction have motivated the development of tAMPer, a method that characterizes AMPs' host toxicity. The method is robust, and performs better than existing methods of protein toxicity classification. The following chapters describe how these were achieved and benchmarked.

Chapter 2: Methods

tAMPer consists of an ensemble model trained on deep learning embedding representations of an antimicrobial peptide dataset binarized based on thresholding upon the hemolysis and cytotoxicity concentrations reported in the literature. Dataset collection and curation, the models as well as the model evaluation setup are provided in the following sections.

2.1 Dataset collection and curation

The data was primarily collected from the Database of Antimicrobial Activity and Structure of Peptides (DBAASP) v2.702 [33]. The database contains curated information about antimicrobial peptides that are both synthesized and naturally occurring, and contain annotations for activity against different bacterial species as well as for hemolytic and cytotoxic activities as reported in the literature. Table 2 justifies the choice of dataset. While CAMP [34] also has cytotoxicity information reported for some AMPs, the information therein is incomplete. DBAASP is the largest resource for the hemolytic and cytotoxic activity of AMPs.

Table 2: Overview of different database sources for antimicrobial peptides.

Database	No of entries	Target object	Detailed chemical structure	Activity against microbial/cancer cell	Cytotoxicity against healthy cell
DBAASP [33]	>15000	+	+	+	+
APD3 [35]	>2000	+	NC	NC	-
CAMP [34]	>8000	-	-	NC	NC
YADAMP [36]	>2000	-	-	NC	-

NC: not complete
'+': data exists in the database
'-': data does not exist in the database

I used the HC_{50} and CC_{50} values reported in DBAASP in order to binarize the data by labelling entries as toxic and non-toxic. For this purpose, a custom heuristic was applied, where a peptide is considered toxic if it has an annotation of HC_{50} and CC_{50} of up to 250 μg/mL and nontoxic if the HC_{50} or CC_{50} concentration exceeds that. These thresholding values indicate the ultimate therapeutic index windows that were considered appropriate for creating a labelled dataset to be used for downstream classification.

In order to retrieve the data from DBAASP, the regular search option in the database website was used for the purpose of querying the sequences, with selected options:

- Regular search
- Full sequence
- Complexity: Monomer
- UniProt ID: All
- Without N or C terminus modification
- No unusual amino acids
- Hemolytic and cytotoxic activities: checked

After the sequences were collected from the query result (accessed on 20 March 2020), the Python API of DBAASP was used to perform peptide card queries, which collected all relevant information from the database for the peptides into a python pandas data frame [37]. The hemolysis values were checked throughout for binarizing the data based on the above custom-defined thresholds.

In order to automate the process of labelling the collected peptides, the following edge cases were considered:

- Multiple annotations of a given peptide may be the result of different publications studying the same peptide for toxicity, or due to multiple reported experiments on that peptide. In such cases, all the annotations were considered and the peptide was added to the dataset if all the inferred labels agreed.
- Under the assumption that toxicity increases proportionally to the concentration of peptide, whenever there were annotations of toxicity deviating from the typical "50% (hemo)lysis", which is common due to the lack of standard for reporting, a linear extrapolation was performed on the provided concentration. For peptide i, if the toxicity/lysis level reported were x_i (%) for a peptide concentration of c_i, the concentration considered for labelling would be $50c_i/x_i$.
- Any concentrations expressed in μM were converted to μg/mL by using the peptide molecular weights, which were queried through the tool ExPASy [38].

To augment the positive set, potentially hemolytic or cytotoxic antimicrobial peptides were also queried from the SwissProt database, using the following keywords:

- Hemolysis [KW-0354], Antimicrobial [KW-0929], NOT Neurotoxin [KW-0528], length: [5 TO 100]
- Toxin [KW-0800], Antimicrobial [KW-0929], NOT Neurotoxin [KW-0528], NOT Ion channel impairing toxin [KW-0872], length: [5 TO 100]

After removing any duplicates and restricting the length of peptides to 100 residues, the remaining dataset comprised 2635 peptides. Peptides labeled as positive (1093) and negative (1542) are further split into a training set of 873 positive and 1200 negative peptides, and a test set of 220 positive and 342 negative peptides for downstream training and analysis.

2.2 Protein embedding methods

Protein sequence embedding methods are a new approach to approximating fundamental protein features in a domain-independent manner. The successful application of deep learning methods into problems like protein structure prediction, functional annotation, and protein engineering, coupled with the rapidly increasing availability of raw protein sequence data has inspired us to adapt deep learning language modelling approaches [26, 39] into the protein representation space. There are few methods for the task, with the most common ones being SeqVec [27], ProtVec [40], and UniRep [41].

2.2.1 SeqVec embedding representations

SeqVec adapts the ELMo natural language model [26] for protein sequences, and is available at https://github.com/Rostlab/SeqVec [27]. The model itself is composed of three layers: a non-contextual CharCNN layer [42], followed by two Bidirectional LSTM (Bi-LSTM) layers (Figure 5). Each layer outputs a single latent vector of length 1024 by default, and the authors refer to SeqVec (or SeqVec embedding) as the vector sum of all of these hidden layer outputs. The authors explain that they have adapted the model for protein sequences by training it on UniRef50 [43] for ~3 weeks. In this training process, the model learns internal representations and dynamic relations of the residues in the context of the proteins, with the goal of correctly predicting the next amino acid; thus, each protein is treated as a sentence and each residue as a word.

For every sequence in the tAMPer training and test sets, a per protein vector embedding of length 1024, as well as a per residue matrix embedding $X \in \mathbb{R}^{n \times 1024}$, where n is the length of the peptide, was obtained as shown in Figure 6.

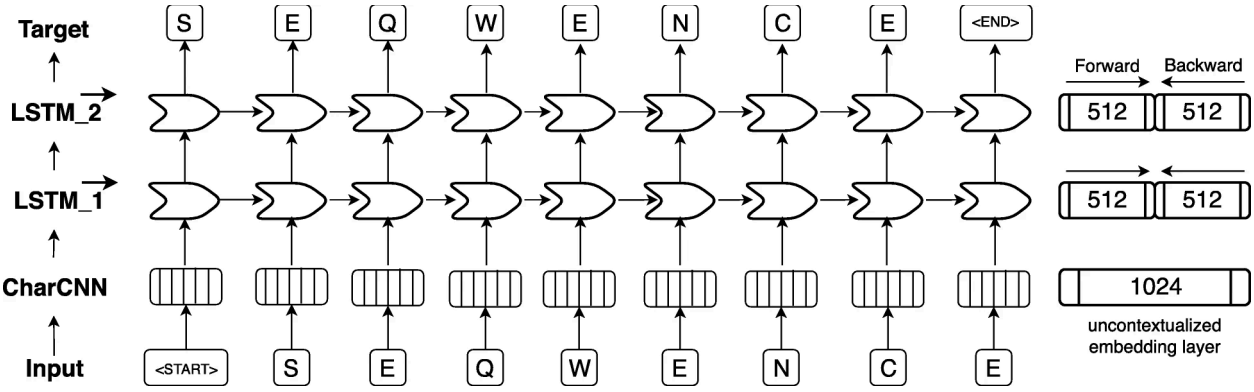

Figure 5: ELMo-based SeqVec model schematic.

The bottom row represents the input layer to the model. In this case it is a peptide "S E Q W E N C E", padded with special <start> and <end> tokens. The CharCNN layer maps each word (or residue) into a 1024-dimensional latent space. This output is input to the first Bi-LSTM layer (512 LSTM units for each direction, shown to the right), and an identical second Bi-LSTM layer is stacked on top of the first, taking as input the hidden states of the previous layer. The forward and backward pass are optimized independently in order to prevent "information leakage" between the passes, and both directions are trying to correctly predict the next word (residue) in the peptide. Only the forward direction of the Bi-LSTM layers is depicted for simplicity or representation.

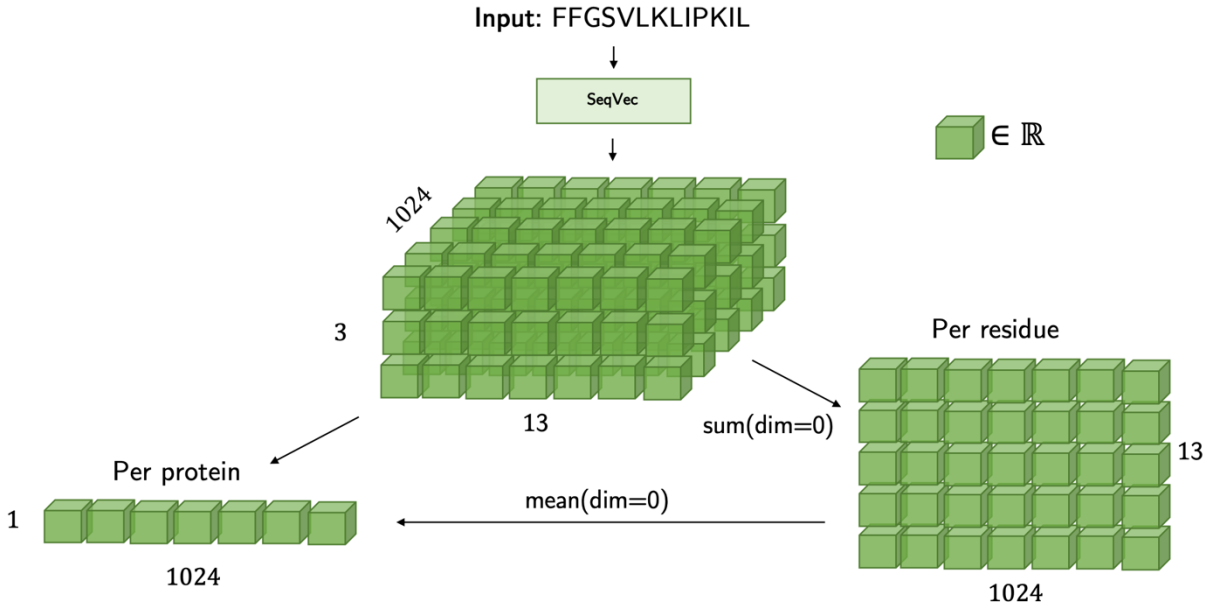

Figure 6: Per protein and per residue embeddings for an example peptide.

The SeqVec model returns the hidden state vectors for all residues across all three layers of the Elmo model, providing a data structure $\mathbf{K} \in \mathbb{R}^{3\times n\times 1024}$. Summing over the three layers' outputs, returns a matrix $L \in \mathbb{R}^{n\times 1024}$, which is interpreted as the SeqVec per residue matrix embedding. Going further and averaging over all residue vectors the SeqVec per protein embedding is obtained $\mathbf{x} \in \mathbb{R}^{1\times 1024}$.

2.3 Classification models

The tAMPer model is an ensemble of five base classifiers. The per protein embedding set is used to train a Random Forest [44], a Gradient Boosting classifier [45] and a Logistic Regression model [46], as well as a stacking ensemble model [47] that uses these models in its first level, and a Logistic Regression model in the second and final level. Additionally, a sequential Bi-LSTM [48] model was trained on the SeqVec per residue embedding set (Figure 7).

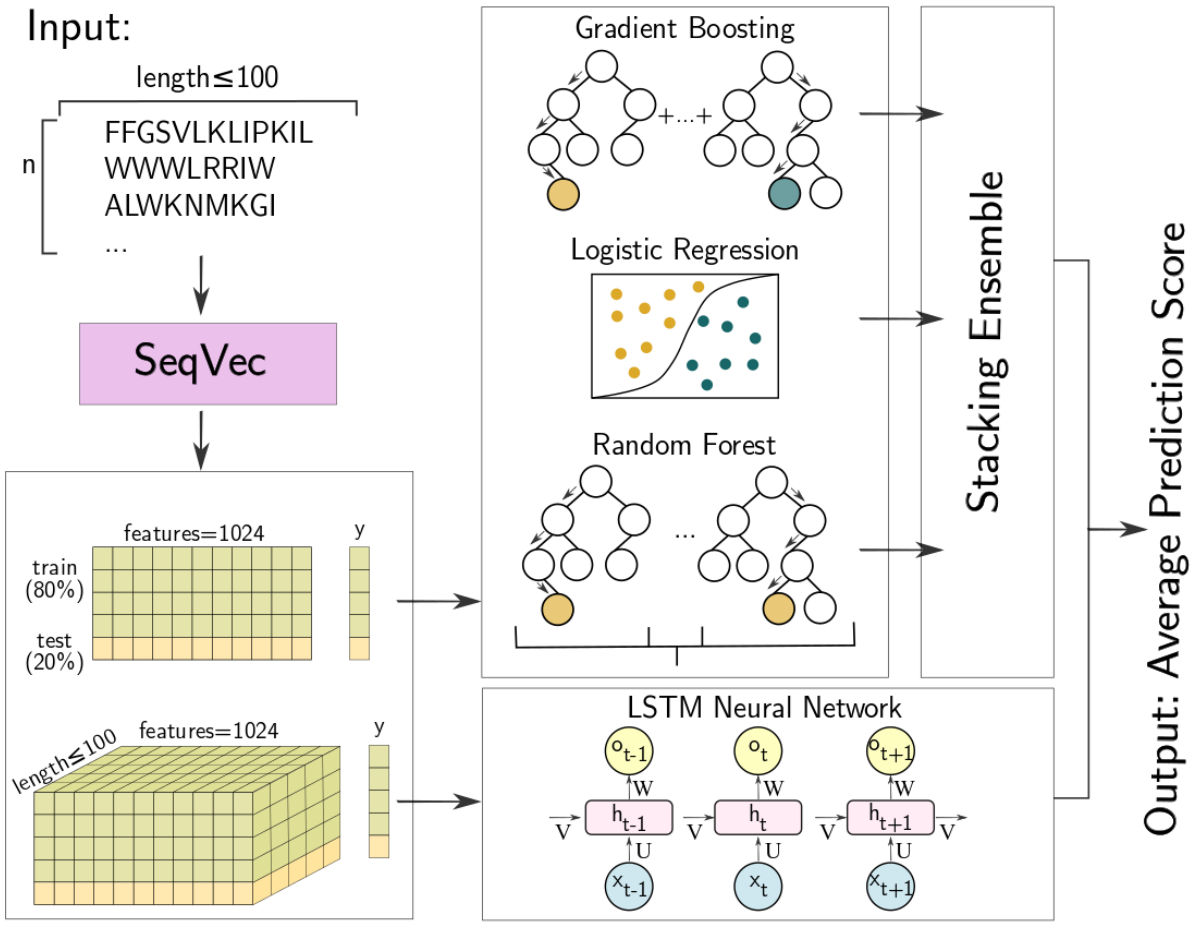

Figure 7: Overview of tAMPer model.

For a dataset of peptides of length at most 100 residues, the SeqVec model was used to obtain per protein embeddings, where each peptide was expressed as a length 1024 vector, as well as per residue embeddings, where each amino acid was embedded into a length 1024 vector. After splitting the data 80/20% into train and test sets, respectively, three base models and a stacking ensemble were trained on the per protein embeddings, and an LSTM network was trained on the per residue embeddings of the data. The average prediction score of the five models was taken as the final output of the ensemble.

2.3.1 Base classifiers

The Random Forest model is composed of 300 estimators with a maximum depth of 15, while the Gradient Boosting classifier is a model of 100 estimators, with a learning rate of 0.1 and maximum depth of 3. The hyperparameters for these models as well as the third base classifier (Logistic Regression) were chosen through 6-fold cross validation and exhaustive search over extensive parameter sets for each model. The scikit 0.20.1 implementations of these models were utilized for training [49].

2.3.1.1 Random Forest

The model was instantiated with parameters max_depth=15, n_estimators=300. The rest of the parameters were kept the same as the default package version. This implementation of random forests fits 300 decision trees on various sub-samples of the dataset, and averages the trees' predictions to improve the predictive accuracy. With the default values, it uses bootstrapping to select a dataset of samples, at most as big as the original dataset used for training. This enables the different learners to train on similar but not the same sequences, increasing overall performance of the classifier. Additionally, random forests make use of random choices when making splitting decisions on the $\sqrt{d}$ of features, where d is the total number of features in the input dataset. In this case $d = 1024$, meaning in every split decision of

the trees in the random forest around 512 features will be selected randomly and used for the decision. This enables the errors to be more independent for different trees.

2.3.1.2 Gradient Boosting

The model was instantiated with parameters n_estimators=100, learning_rate=0.1, max_depth=3. The rest of the parameters were kept the same as the default package version. The main idea of boosting is to add new models to the ensemble sequentially. The gradient boosting implementation used throughout this work fits 100 regression trees, or weak learners.

More specifically, for a sample peptide embedding vector x_i the prediction can be expressed as $\hat{y}_i = F_M(x_i) = \sum_{m=1}^{M} h_m(x_i)$, where h_m are estimators or weak learners, and $M = 100$. The purpose of the approach is to consecutively fit a new estimator h_m given the previous ensemble F_{m-1} in order to minimize a sum of losses L_m: $h_m = \arg\min_h L_m =$ $\arg\min_h \sum_{i=1}^{n} loss(y_i, F_{m-1}(x_i) + h(x_i))$. In this case, n is the number of samples in the training. An approximation of the loss value in the equation above enables us to write: $h_m \approx \arg\min_h \sum_{i=1}^{n} h(x_i) g_i$, where g_i is the derivative of the loss evaluated at F_{m-1}. This is minimized when $h(x_i)$ is fitted to predict a value that is proportional to $-g_i$, enabling training via gradient descent in a functional space. For classification, in order to get probability prediction of the class, the sigmoid function $\sigma(\cdot)$ is used. For example, the probability that x_i is positive (toxic) is modelled as $p(y_i = 1|x_i) = \sigma(F_m(x_i))$.

2.3.1.3 Logistic Regression

The model was initialized with default parameters. The optimization model tries to minimize the cost function $\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(e^{-y_i(X_i^T w + c)} + 1)$. The first term in the equation is the l_2 regularization term, which is used to control overfitting, C is the coefficient

that controls regularization, w is the weight parameter vector that is being learned and 1 is a bias term.

2.3.2 Stacking ensemble

A popular ensemble method used to improve predictions in machine learning problems is stacking ensemble. This method combines a range of well-performing classifiers via a meta classifier or regression method. For the stacking model, the base classifiers described above were used as level 1 classifiers. First, the training set was split into five folds, and four of the folds are used to train the three base models and predictions are obtained for the 5^{th} validation holdout set. After repeating this for the whole train set, all three sets of predictions gradually obtained from training on 4/5 splits and validating on the remaining splits comprise a new training set for a second level training of a Logistic Regression model, and the output score of that is considered as the final prediction of the stacking model. The first level classifiers were trained again on the whole train set to obtain the test set predictions for evaluation of the performance of this ensemble. Stacking model training is depicted in Figure 8.

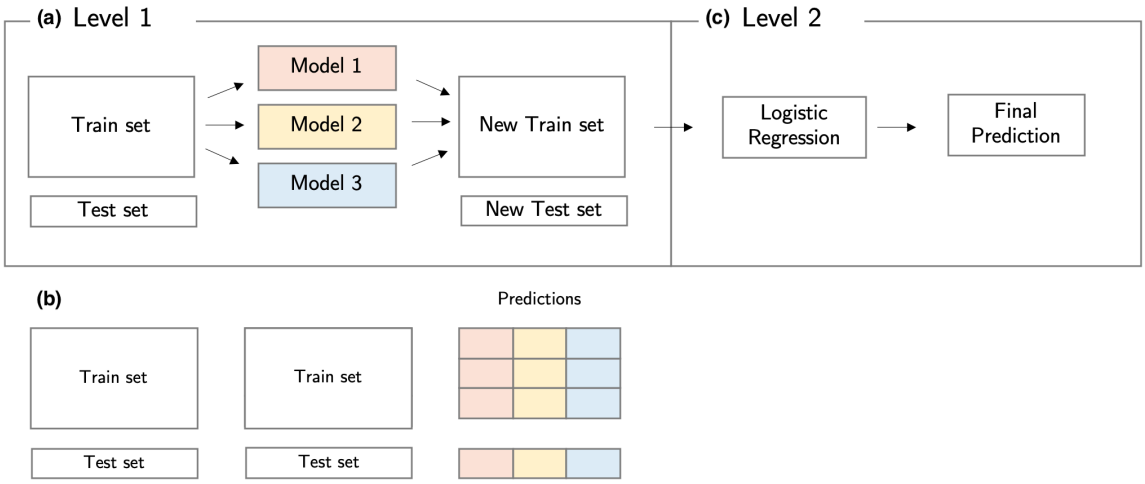

Figure 8: Stacking ensemble model training overview.

(**a**) The three base models (section 2.3.1) are trained on the SeqVec vector embeddings in the first level of training. As depicted in (**b**), this is done by splitting the training set into k-folds and repeatedly training each model on k-1 of the folds and predicting on the remaining k^{th} fold. The prediction scores from each fold were used to create a new dataset for the meta classification. Additionally, to obtain the prediction scores for the test set, which are used for evaluation, the whole training set was used for training on each classifier and the prediction scores were saved for the downstream model performance evaluation on the test set. (**c**) The level 2 classifier is a Logistic Regression, and its predictions are the final stacking model output.

2.3.3 Sequential LSTM model

The LSTM model is built using Keras [50], and it is a composed of a Bi-LSTM layer of 50 units and two Dense layers of size 50 and 10 with "relu" activation and an output layer with "sigmoid" activation. Keras 2.2.4 with the Tensorflow 1.13.1 backend library was used for developing the models, making use of the *Sequential, Bidirectional, LSTM, Masking, Dense, Dropout* layers internally. A sketch of the model is provided in Figure 9. The Adam optimizer [51] was used for training, with the mean squared error at the objective loss function. Six-fold cross validation over a set of hyperparameters was used for tuning these hyperparameters and deciding the best performing model.

The input for this LSTM network is the per residue embeddings of the training set as obtained from SeqVec, with each sequence being represented by a matrix $X \in \mathbb{R}^{100\times1024}$. A Masking layer ensures only the relevant residues of these zero-padded matrices are used during training of the model. The Bi-LSTM layer encodes positional information for both forward and backward directions. A dropout rate of 0.2 is used for regularization. The hidden state of the last residue for both directions is concatenated and serves as input to the next layer. This is a fully-connected layer of 50 units, followed by a fully-connected layer of 10 units, both with 'relu' activation, and finally followed by an output layer with 'sigmoid' activation.

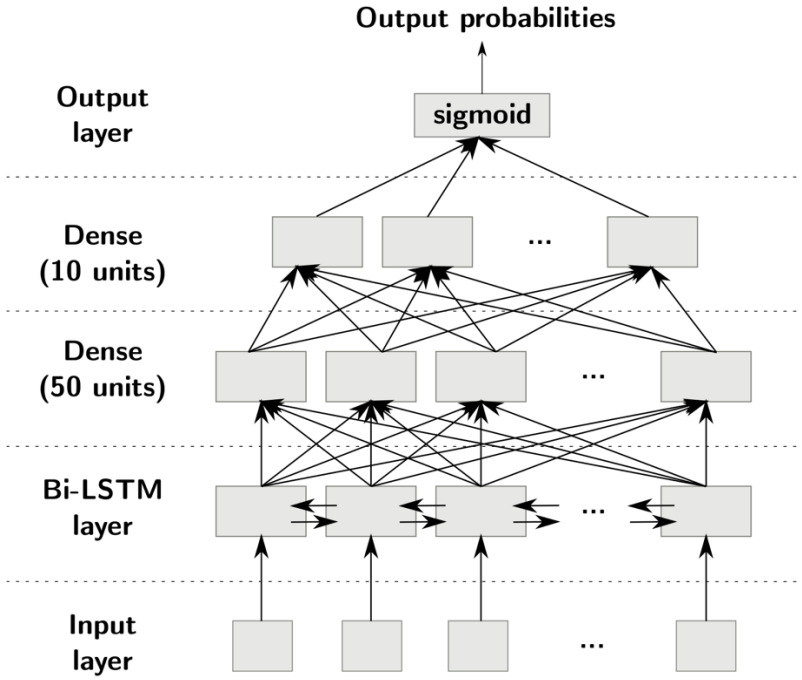

Figure 9: Overview of the Bi-LSTM sequential model used for prediction.

2.3.3.1 Bi-LSTM model architecture

A single sequence input for the Bi-LSTM model is a series $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_L}$ where L is the length of the sequence and each $\mathbf{x_t}$ is a 1024-dimensional vector of real numbers ($t = 1,2 \ldots L$). The Bi-LSTM layer takes these residue embedding vectors as input and encodes positional information for both the forward and the backward LSTM layers, such that the output vector for each residue is a concatenation of the forward and backward directions, with a dimension of $d_h = 2 \times 50 = 100$.

An experimentally evaluated dropout rate of 0.2 is applied to the Bi-LSTM layer to allow regularization during training. The input to the subsequent fully connected layer is the hidden state of the last residue in the sequence, which we can denote as vector $\mathbf{h} \in \mathbb{R}^{d_h}$. The first fully connected layer is composed of 50 hidden units, and the vector **h** is used to optimize parameters $W^{(1)} \in \mathbb{R}^{50 \times d_h}, v \in \mathbb{R}^{50 \times 1}$. The output of this first fully connected layer is calculated through the transformation $y_i^{(1)} = v^{\mathrm{T}}\mathrm{ReLU}(W^{(1)}h_i)$; the second fully connected layer performs the input transformation $y_i^{(2)} = u^{\mathrm{T}}\mathrm{ReLU}(W^{(2)}y_i^{(1)})$ with parameters $W^{(2)} \in \mathbb{R}^{10 \times 50}, u \in \mathbb{R}^{10 \times 1}$ (note subscript i refers to an arbitrary node i for any layer); the third layer is a unit sigmoid transformation of this output as $\hat{y} = w^{\mathrm{T}}\sigma(W^{(3)}y_i^{(2)})$ with parameters $W^{(3)} \in \mathbb{R}^{1 \times 10}, w \in \mathbb{R}^{1 \times 1}$. The bias inputs are implicit for all layers and were not included for brevity.

2.3.3.2 Optimization and loss function

The sequential Bi-LSTM model minimizes the mean squared error loss: $\ell(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$. The Adam optimizer [51] with parameters initialized by default as $\lambda = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ was used. A batch size of 10 was used for training, and the training was stopped after 10 epochs to avoid overfitting. The default Keras parameter

initializer, the *Xavier initializer* [52], was used for all layers. In this initializer, a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ is initialized as: $\boldsymbol{W} \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}, +\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}\right]$, where $U[a, b]$ is the uniform distribution in the range $[a, b]$.

2.3.4 Final ensemble

The final prediction score used is the mean of the predictions of the three base models, the stacking model and the Bi-LSTM model. It can be interpreted as a probability score between [0,1], and sequences with a prediction score > 0.5 are categorized as toxic, and those with score ≤ 0.5 nontoxic. By the toxic label, we indicate that the peptide has a hemolytic or cytotoxic potential beyond the set threshold.

2.3.5 Implementation details

All the code for implementing the methods described was written in Python. A custom Anaconda [53] environment was used, and the packages described below are necessary dependencies:

- Python 3.6.9
- Tensorflow 1.13.1
- Keras 2.2.4
- Pytorch 0.4.1
- Allennlp 0.7.2
- Biopython 1.76

2.4 tAMPer ensemble model evaluation

2.4.1 Input representation evaluation

The choice of SeqVec embeddings is evaluated against the baselines of one hot encoding representation of input, amino acid composition, physicochemical properties, as well as against the choice of fastText embedding [41] and UniRep embedding [54] representations – two alternate embedding models. To better understand the benefit of using SeqVec embeddings over other baseline representations and alternate embedding models, the six-fold cross validation performance of a VotingClassifier [49] ensemble of the three base models described in Section 2.3.1 is compared against the same ensemble performance when using the same six-folds on the SeqVec representation input. The class label returned as the output of the VotingClassifier is the argmax of the sum of predicted probabilities, as the approach used here is "soft" for the ensemble. A paired Student t-test is used to test the null hypothesis that the metrics obtained by the six-fold cross validation are not drawn from the same distribution, with a p-value threshold of 0.05.

2.4.1.1 One hot encoding representation

The one hot encoding representation is illustrated in Figure 10.

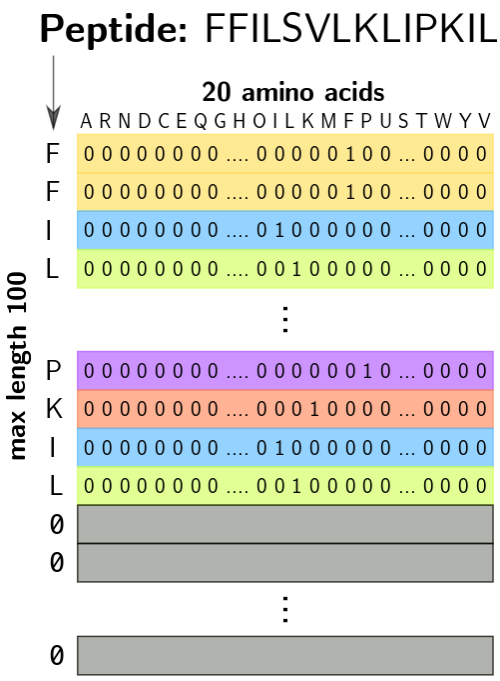

Figure 10: One hot encoding representation for an example AMP.

As shown in Figure 10, the one hot encoding is a sparse matrix of size 100×20, where 100 is the max length of peptide, and 20 is the amino acid alphabet size. If the peptide has a smaller number of residues, it is padded by zero-vectors. Whenever the algorithm admits a vector input, the one hot encoding matrix was flattened to a 2000-dimensional vector. The three base models and the LSTM model were used to train with the one hot encoding representation, and the average of the four prediction scores was considered for evaluations.

2.4.1.2 Other baseline compositions

The base models were used to train on the amino acid composition and the physicochemical features as baseline representations, and they were compared against respective models trained on the SeqVec embeddings. The stacking model was omitted from these baseline comparisons for simplicity, while the sequential Bi-LSTM model cannot be used for these input

baselines as they are not sequential. Similar to above, the “soft” VotingClassifier is used to obtain the ensemble predictions.

The physicochemical feature representation used as a baseline is depicted in Figure 11. The features used to describe the sequences merely serve to understand whether there is potential to distinguish based merely on this different representation.

	hmoment	hydrophob.	charge	mol. weight	pI.
FFILSVLKLIPKIL	0.54	-0.34	4.74	2169	9.53
WWWLRRIW	0.64	-0.53	1.99	1301	12.5
RLKRWWKFL	0.74	-0.91	3.99	1332	12.7
...			...		

Figure 11: Physicochemical features as representation of sequences.

Each peptide is represented by the features of hydrophobic moment, hydrophobicity, charge, molecular weight and isoelectric point.

2.4.1.3 UniRep embedding representation

The UniRep architecture is a sequential model that is based on the mLSTM units [55]. mLSTM units are designed to allow for input-dependent transitions by adding a multiplicative gate so that the hidden-to-hidden weight matrix is a function of the current input. Thus a “factor state” is also computed, adjusting the input x_t to the LSTM cell by these multiplicative weights [55, 56]. The three variations provided by the authors differ in the number of mLSTM units the model comprises, namely 64, 256 and 1900 units. All three models are trained on the UniRef50 dataset [43], which contains about 24 million sequences, no two of which have a similarity higher than 50% with each other. Additionally, unlike previous NLP models that use the final

hidden state (corresponding to the C terminus in our case) as embedding, the UniRep embedding combines all hidden states by averaging the hidden states of all model units, and thus integrating long-range dependencies across amino acids. The corresponding embeddings for the three types of models are vectors of length 64, 256, 1900 for all model architectures.

2.4.1.4 fastText embedding representation

fastText is a fast and scalable NLP framework implementing the skip-gram model, which is a supervised learning task that learns word representations based on the likelihood that a target word appears within a context [41]. In order to utilize fastText for embeddings, the fastText skip-gram model was trained on the UniRef50 dataset, with every protein treated as a sentence, and every amino acid as a word, similar to SeqVec mode of training. The result of the training is ability to represent each of the 20 amino acids as a numerical vector of length 100, $e_w \in \mathbb{R}^{100}$ for a given amino acid w. The fastText embedding for a sequence is the average of the sequence residue fastText vectors.

2.4.2 Comparison with state-of-the-art

The performance of tAMPer against other methods in the literature was evaluated using the metrics of accuracy, sensitivity, specificity, precision, F1 score, and area under the receiver operating characteristic curve (AUROC). These methods include the comparators HemoPI, ToxinPred and TOXIFY. Both the benefits of the tAMPer ensemble model over other models, as well as the choice and curation of dataset for the AMP host toxicity prediction problem were evaluated thoroughly. For this reason, the tAMPer model trained on tAMPer data (noted as tAMPer-t) was compared against TOXIFY, HemoPI and ToxinPred methods on the tAMPer test set. The comparators were evaluated with their original models online. Additionally, as the only

comparator with availability of re-training, TOXIFY was re-trained on the tAMPer training set. The predictions of all models on all available test sets were also obtained to better understand the difference in performances among the models. Further, the tAMPer and TOXIFY models were also re-trained on the training sets of HemoPI and ToxinPred, and the predictions on their respective test sets, as well as the tAMPer test set were obtained for a complete and fair comparison of models and datasets.

2.4.3 Biological relevance of using tAMPer

The significance and usability of the tAMPer method is evaluated by showing the change in prediction score when highly active AMPs are mutated one amino acid at a time.

In order to illustrate this, heatmaps indicating the tAMPer model score change when single point mutations were performed on highly active AMPs were plotted. The chosen peptides for this analysis have been previously reported [57], and a few of these sequences were shown to have high activity against some WHO "Priority 1" and "Priority 2" pathogens [58], including a multi-drug resistant (MDR) strain of *Escherichia Coli*. The chosen mutants have been shown to be at least as active as the original peptides in soon to be published work by our lab. Table 3 details the original and corresponding mutated sequences.

Table 3: Previously reported and tested AMPs and corresponding mutations, where the residues highlighted in red are the ones that are mutated.

Name	Sequence	Mutated sequence
P2	FFPRVLPLANKFLPTIYCALPKSVGN	FFPRVLRLANKFLP**K**IYCALPKSVGN
P3	GLLDIIKDTGKTTGILMDTLKCQMTGRCPPSS	GLLDIIK**K**TGKTTGILMDTLKC**C**MTGRCPPSS

P4	GLLDIIKTTGKDFAVKILDNLKCKLAGGCPP	GLLDIIKTTGKDFAVKILDNLKCKLAGGCP**K**
P5	FFPIIARLAAKVIPSLVCAVTKKC	FFPIIARLAAKVIPSLVC**K**VTKKC
P6	GLWETIKTTGKSIALNLLDKIKCKIAGGCPP	GLWETIKTTGK**K**IALNLLDKIKCKIAGGCPP
P11	ALVAKIQKFPVFNTLKLCKLELEII	ALVAKIQKFPVFNTLKLCKL**K**L**R**II

Chapter 3: Results

The following sub-sections describe the results obtained on different evaluations on the method, including evaluation of the dataset and the ensemble models, as well as the biological importance of the method.

3.1 Input representation results

The six-fold cross validation results on the same splits for the three base classifiers ensemble are evaluated against all described input representations described in Section 2.4.1. The results in Table 4 consistently show that there is a significant increase in the sensitivity, F1-score, AUROC and accuracy performance metrics when using the SeqVec embeddings for input representation, compared to others, as evaluated by a paired Student t-test on each metric based on a six-fold cross validation of the methods. The metrics of specificity and precision in the case of the physicochemical feature representation, amino acid composition and one hot encoding representation, both for the base models' ensemble, as well as the ensemble with the sequential model were shown to be better than using SeqVec representations. Further, the only case in which SeqVec vectors did not improve the sensitivity is when the UniRep-1900 features were used, and is also the only case that does not exhibit significant increase in F1-score when using SeqVec embeddings. Notably, the fastText representation was shown to not improve specificity over SeqVec (Table 4, Figure 12). In Figure 13, it can be observed that the SeqVec representation has a better performance than the corresponding one hot encoding feature inputs for the ensemble of the base models, the Bi-LSTM sequential model and the ensemble of these two, which was taken to be the average score of the prediction of each model. The only exception is for the specificity and precision metrics (Table 4).

Table 4: P-values for paired Student t-test of each representation method compared with SeqVec input representation.

Model (BASE)	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
Phys-chem	0.0041	*0.2232*	*0.0541*	0.0080	0.0094	0.0128
AAcomp	0.0084	*0.5423*	*0.0770*	0.0105	0.0121	0.0151
fastText	0.0183	*0.1887*	0.0428	0.0170	0.0170	0.0181
UniRep-1900	*0.1574*	0.0262	0.0130	*0.0648*	0.0492	0.0355
UniRep-256	0.0017	0.0347	0.0059	0.0015	0.0018	0.0021
UniRep-64	0.0009	0.0480	0.0032	0.0003	0.0003	0.0004
OneHot	0.0019	*0.3114*	*0.0553*	0.0024	0.0031	0.0049
OneHot (+Bi-LSTM)	0.0062	*0.3152*	*0.0713*	0.0034	0.0035	0.0046

All insignificant p-values ≥ 0.05 are indicated in italic.
BASE: The ensemble using VotingClassifier of the three base classifiers described in section 2.3.1.
(+Bi-LSTM): Average of the predictions of the **BASE** models' ensemble with the (**Bi-)LSTM** sequential model prediction.

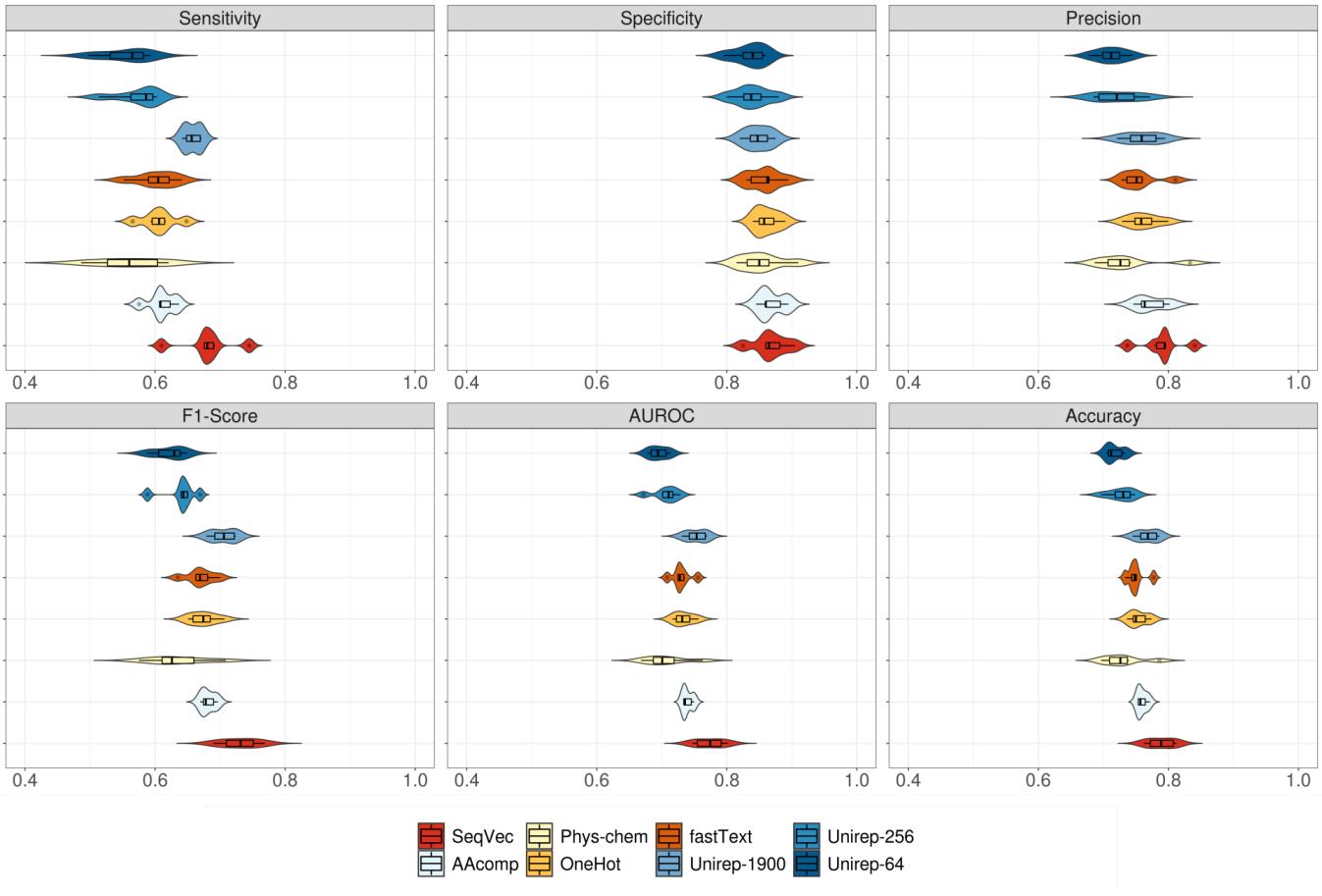

Figure 12: Comparison of all input representations considered using the three base models' ensemble.

The training set for each representation is used for a 6-fold cross validation training of the base models' ensemble (described in section 2.3.1). The SeqVec representation (in red) appears to be performing better than other representations for the metrics of sensitivity, F1-score, AUROC and accuracy. For these metrics, the closest high performing representation is UniRep-1900.

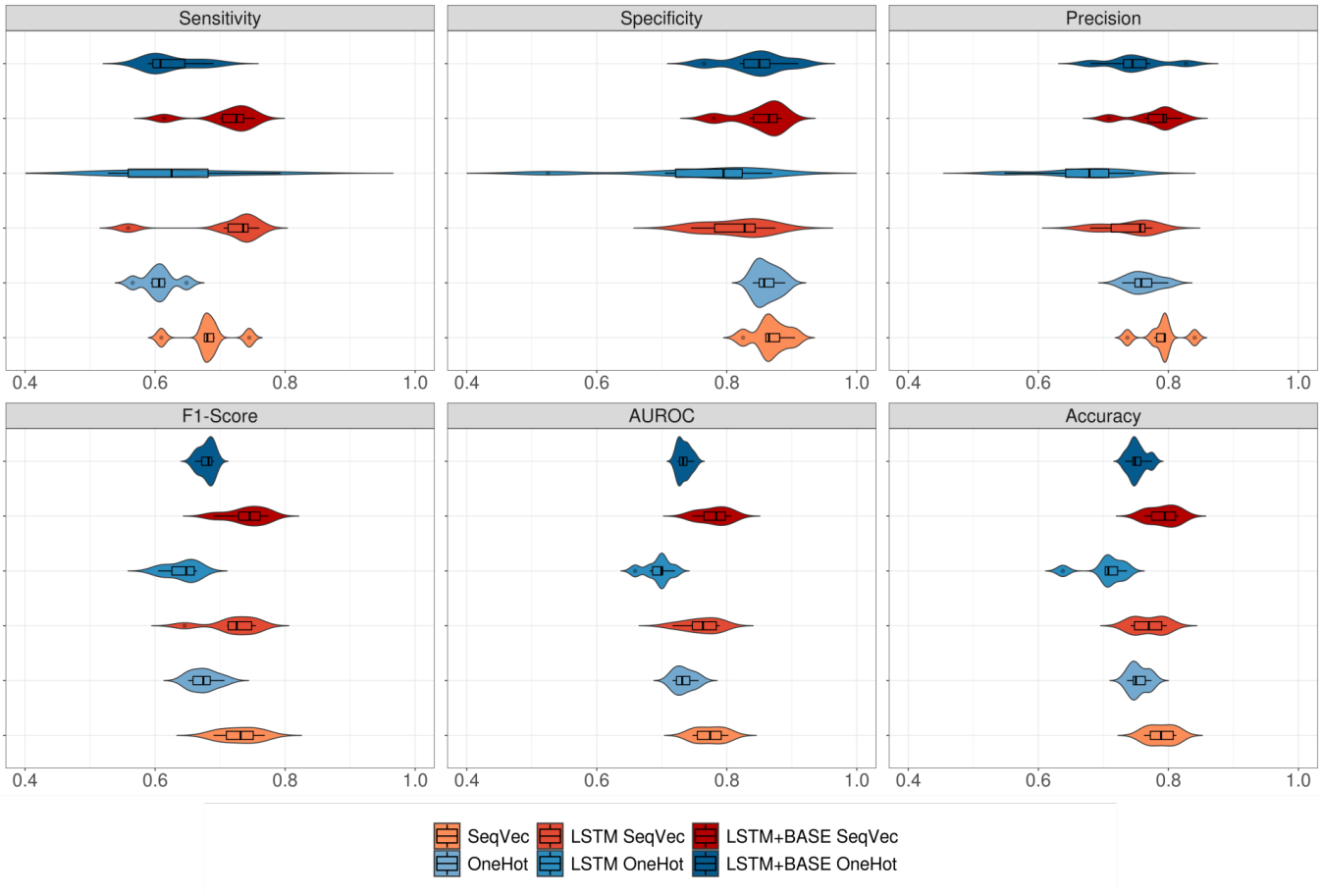

Figure 13: Comparison of SeqVec embedding models with the one hot encoding feature baseline.

The training set for each of the SeqVec and one-hot encoding representation is used for a 6-fold cross validation training of the base models' ensemble (base models described in section 2.3.1, VotingClassifier ensemble used and shown as BASE here), the sequential model (shown as LSTM), and the average of the predictions of the BASE and LSTM sequential model predictions (shown as LSTM+BASE). The SeqVec representation appears to be performing better than other representations for the metrics of sensitivity, F1-score, AUROC and accuracy.

3.2 Comparison with the state-of-the-art

The following sections detail the performance of the tAMPer method against other methods in the literature, both when trained on the tAMPer training set as well as when trained on the other methods' training sets.

Table 5: Overview of training and testing setup for comparisons with existing methods.

		Predicted on test set of			
	Train set of	tAMPer	HemoPI	ToxinPred	TOXIFY
Trained on	tAMPer (-t)	✓	✓	✓	✓
	HemoPI (-H)	✓	✓		
	ToxinPred (-TP)	✓		✓	
	TOXIFY (-TF)	✓			✓

Figure 16

Figure 17

Figure 18

The table highlight color mapping legend to the right points to the ROC figures related to these combined performance evaluations.
✓: *Trained on train set of* and *Predicted on test set of* evaluation combinations performed.
tAMPer and TOXIFY are trained on tAMPer train set. These models are used for predictions on all available methods' test sets (shown in red highlight). tAMPer and TOXIFY are trained on HemoPI, ToxinPred, TOXIFY train sets. These models are used for predictions on all respective methods' test sets (shown in yellow highlight). These models are also used for predictions on the tAMPer test set (shown in blue highlight).

3.2.1 Performance comparison setup

3.2.1.1 HemoPI

The server predictions of HemoPI were used for comparisons (https://webs.iiitd.edu.in/raghava/hemopi/). Under the "Virtual Screening" tab, the SVM (HemoPI-2) based model was used for predictions. Additionally, the HemoPI-2 datasets downloaded from the "Get Datasets" tab were used throughout this work, as they seem more relevant to our task in terms of the similarity in dataset collection and labelling. The main dataset was used as a training set and the validation dataset as test set.

3.2.1.2 ToxinPred

The server predictions of ToxinPred were used for comparisons (https://webs.iiitd.edu.in/raghava/toxinpred). Under the "Batch Submission" tab, the SVM (Swiss-Prot) based model, with the default values for the e-value cut-off (10) and the SVM threshold (0.0) was chosen for performing all predictions. Similarly, we have chosen the Swiss-Prot datasets of this method; Main dataset–1 as the training set and Independent dataset-1 as the test set. This dataset is more balanced than the alternative, and the best reported model performance discussed was based on this dataset.

3.2.1.3 TOXIFY

The Github version of this method was used for trainings and predictions (https://github.com/tijeco/toxify). Under the sequence_data directory, the files under training_data were used as a training set, and the files under benchmark_data as a test set. Any duplicate sequences in the training and test sets were removed. We have used the reported best version parameters for training TOXIFY on the other datasets. These parameters are: learning rate (-lr) 0.01, -epochs 50, -units 270, -window 0. The max length (-max_len) parameter was changed from the default 500 used for training on the TOXIFY train set, to 100 for the tAMPer training set, to 35 for the ToxinPred training set and 100 for HemoPI training set, according to the maximum length of each set.

3.2.1.4 Additional considerations

To avoid any bias that may arise due to sequences in a test set also belonging to some training set of a model being used, any such sequences found in the intersections between train and test were removed from the test sets. For example, when using the HemoPI model to predict

on the tAMPer test set, any sequences in the tAMPer test set that were used for training the HemoPI model were omitted from this prediction.

The size of the train and test sets for different methods, including tAMPer, are provided in Figure 14. Figure 15 displays a box plot view of the length distribution of each of the methods' positive and negative training sets. As can be seen in Figure 14, the TOXIFY method has a very imbalanced training set, and a comparably, counter-intuitively-small test set. Additionally, the length of the sequences in the positive set is in general smaller than the length of the sequences in the negative set, which carries in itself a bias that no other method seems to suffer from (Figure 15).

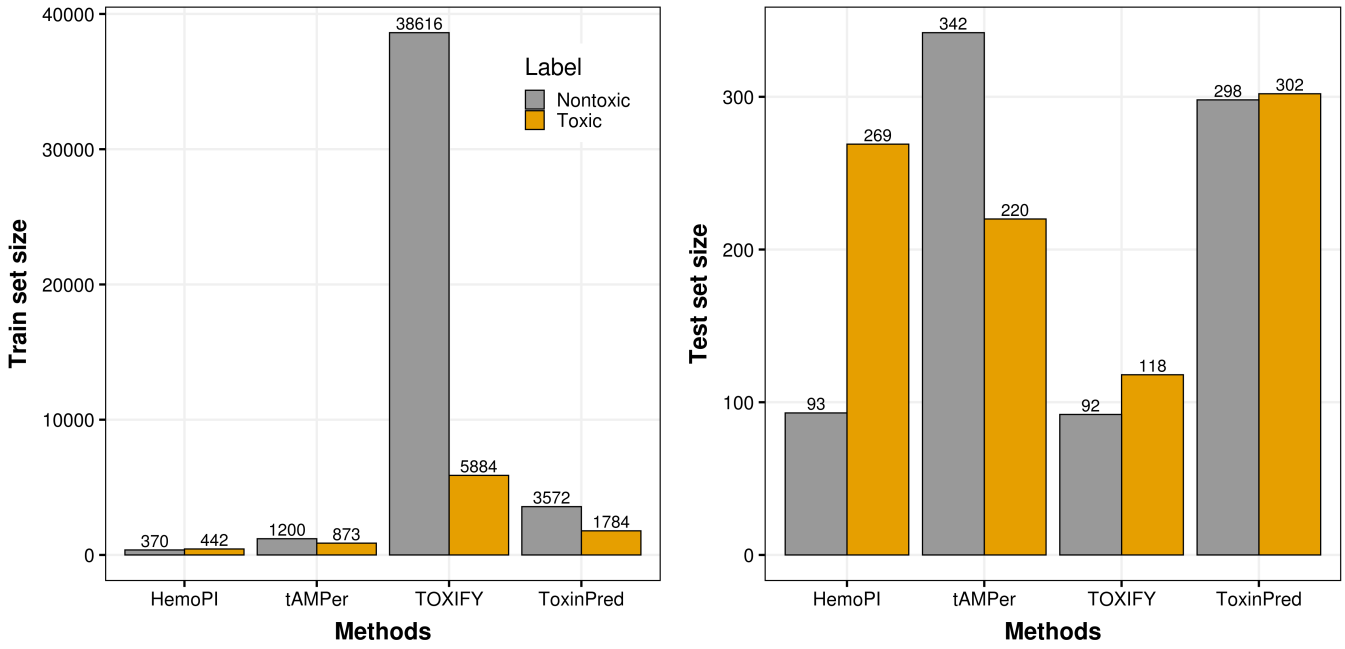

Figure 14: Size of the train and test sets for the different methods.

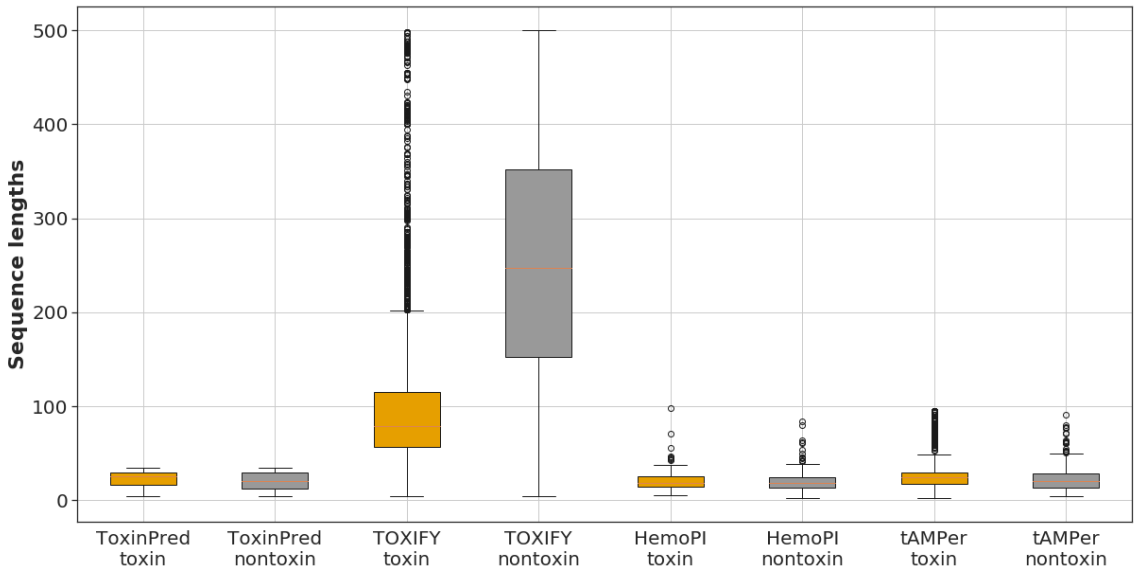

Figure 15: Box plot of the length distribution of the different methods' positive and negative training sets.

3.2.2 tAMPer and TOXIFY trained on tAMPer training set

The performance of tAMPer on the tAMPer test set was compared against all other available methods, including the re-trained TOXIFY method. To avoid confusion the re-trained version of TOXIFY on tAMPer data is noted as “TOXIFY-t”, and it is trained with the best hyperparameters as stated in their paper, while the original model is denoted as “TOXIFY-TF”.

Table 6: Performance of different models on the tAMPer test set.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI	0.445	0.640	0.440	0.443	0.545	0.564
ToxinPred	0.038	**0.938**	0.231	0.065	0.492	0.640
TOXIFY-TF	0.511	0.454	0.342	0.409	0.495	0.474
TOXIFY-t	0.668	0.740	0.623	0.645	0.773	0.712
tAMPer-t	**0.755**	0.848	**0.761**	**0.758**	**0.873**	**0.811**

Note: The HemoPI and ToxinPred static server models were used for predictions. The methods tAMPer-t and TOXIFY-t were trained on the tAMPer training set, indicated by the additional '-t' in their name. The model TOXIFY-TF is the original (static) TOXIFY model.

Among the original models of the three comparators, HemoPI has the best AUROC and F1 metrics, and ToxinPred shows a very high specificity (93.8%) but very low sensitivity (3.8%), which boosts the accuracy to be the best among these models. All metrics were improved for TOXIFY when re-trained (TOXIFY-TF vs TOXIFY-t). tAMPer (tAMPer-t) outperforms all original competitor models in all metrics except specificity, with ToxinPred having a 9% higher specificity. tAMPer achieves the highest accuracy (81.1%), F1 score (75.8%), sensitivity (75.5%) and AUROC (87.3%), the only closest competitor in each of these metrics being the re-trained TOXIFY model, with accuracy (71.2%), F1 score (64.5%), sensitivity (66.8%) and AUROC (77.3%). In order to provide a better comparison of the classification performance of these methods, Figure 16 presents a series of receiver operating characteristic (ROC) curves for the models compared. In these plots the tAMPer and TOXIFY versions trained on to the tAMPer training set are shown. As illustrated by Figure 16, and Tables Table 7, Table 8, and Table 9 detail the performance of all models shown across all metrics, generally the methods perform better on their own test sets, and the performance of tAMPer (or tAMPer-t) is only relatively

good on the HemoPI test set, close across performance metrics with HemoPI itself, and better in specificity and precision by 7.7% and 3.4% respectively.

Table 7: Performance of all models on the HemoPI test set.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI	**0.736**	0.793	0.81	**0.771**	**0.821**	**0.762**
ToxinPred	0.286	0.583	0.426	0.342	0.478	0.429
TOXIFY-t	0.545	0.815	0.779	0.642	0.717	0.668
tAMPer-t	0.591	**0.87**	**0.844**	0.695	0.789	0.718

Table 8: Performance of all models on the ToxinPred test set.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI	0.116	0.822	0.398	0.18	0.357	0.467
ToxinPred	0.328	**0.993**	**0.98**	**0.491**	**0.833**	**0.658**
TOXIFY-t	**0.374**	0.728	0.582	0.456	0.577	0.55
tAMPer-t	0.272	0.711	0.488	0.349	0.524	0.49

Table 9: Performance of all models on the TOXIFY test set.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI	0.060	**0.844**	0.516	0.108	0.414	0.268
ToxinPred	0.596	0.75	0.967	0.737	0.705	0.607
TOXIFY-TF	**0.838**	0.667	**0.984**	**0.905**	**0.791**	**0.831**
TOXIFY-t	0.386	0.625	0.965	0.551	0.508	0.395
tAMPer-t	0.558	0.5	0.968	0.708	0.583	0.556

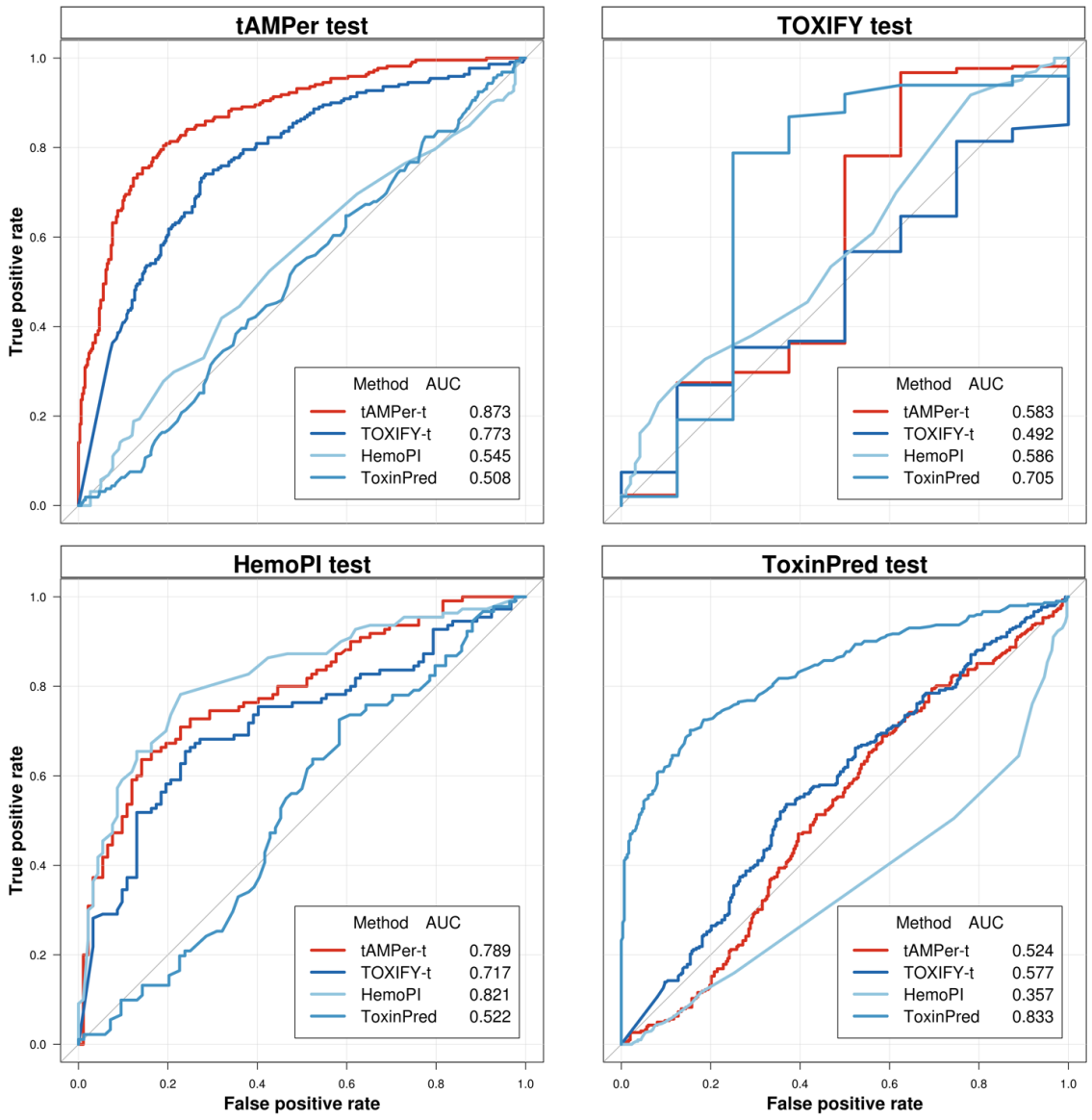

Figure 16: ROC plots performance of different models on the available test sets.

The tAMPer and TOXIFY models are trained on the tAMPer train set. The performance of tAMPer (in red) is best for the tAMPer test set, and similar in performance to HemoPI for the HemoPI test set, whilc it is no better than chance in the TOXIFY and ToxinPred test sets.

3.2.3 tAMPer and TOXIFY trained on other methods' training sets

tAMPer and TOXIFY were also trained on the training sets of all the other comparator methods (TOXIFY, HemoPI and ToxinPred).

Table 10: Performance of models on the test sets of the methods whose training sets were used for training.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI[a]	0.736	**0.793**	**0.81**	0.771	0.821	0.762
TOXIFY-H[a]	0.718	0.739	0.767	0.742	0.768	0.728
tAMPer-H[a]	**0.836**	0.707	0.773	**0.803**	**0.888**	**0.777**
ToxinPred[b]	0.328	**0.993**	**0.98**	0.491	**0.833**	0.658
TOXIFY-TP[b]	0.364	0.946	0.873	0.514	0.764	0.653
tAMPer-TP[b]	**0.368**	0.963	0.91	**0.524**	0.826	**0.663**
TOXIFY-TF[c]	0.838	**0.667**	**0.984**	0.905	0.791	0.831
tAMPer-TF[c]	**0.937**	0.333	0.972	**0.954**	**0.807**	**0.913**

Note: Model names ending with '-H' are trained on HemoPI training set, with '-TP' trained on ToxinPred train set and with '-TF' trained on TOXIFY train set.
a: Predictions on the HemoPI test set
b: Predictions on the ToxinPred test set
c: Predictions on the TOXIFY test set

Table 10 displays the result metrics of predictions of the re-trained models alongside the originals on the original models' test sets. In all cases the re-trained tAMPer model offers higher sensitivity, F1 score and accuracy than the original model or the re-trained TOXIFY model. The AUROC is also similarly improved for the HemoPI and TOXIFY predictions and it is slightly lower (tAMPer-TP AUROC 82.6%) than the ToxinPred AUROC (83.3%) on the ToxinPred test set. Figure 17 displays the ROC curves of the retrained tAMPer and TOXIFY models as well as the HemoPI and ToxinPred model performances on the competitors' test sets. Re-trained tAMPer performs similar to or better than the original methods, and better than re-trained

TOXIFY in all predictions. All these re-trained models were also used for prediction on the tAMPer test set. The ROC is provided in Figure 18. As it is clear from Figure 18 and Table 11, none of the re-trained models can achieve a high performance on the tAMPer test set, and all the re-trained models perform as good as chance or slightly better on this test data.

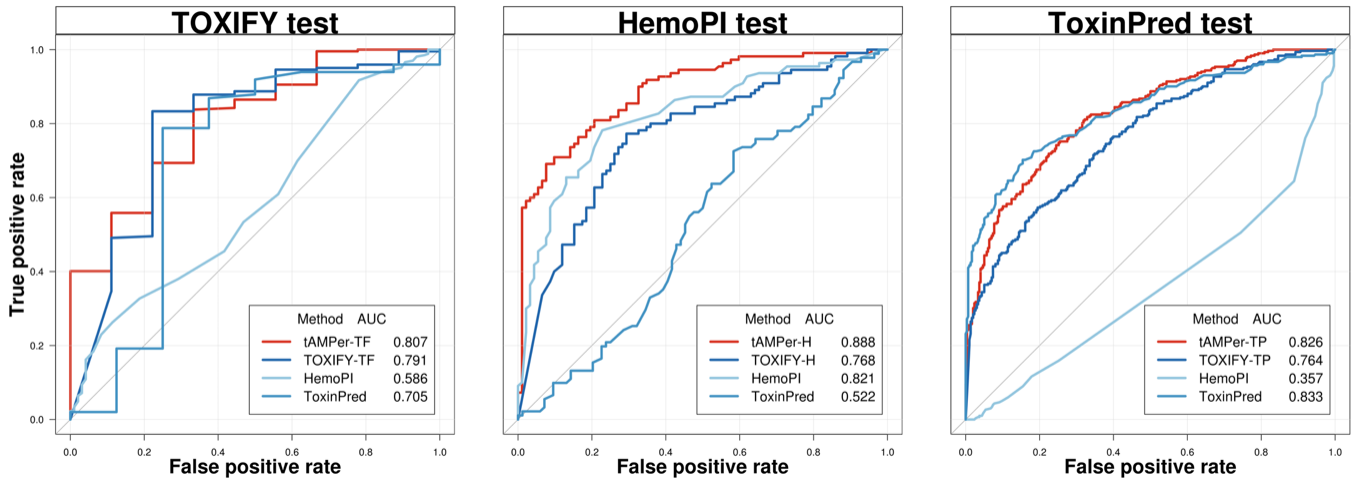

Figure 17: ROC plots performance of re-trained and static models on the test sets of the different methods.

The performance of the re-trained tAMPer models (in red) is better than or similar to the performances of the respective methods whose training sets are used for re-training. The re-trained TOXIFY models perform worse than the re-trained tAMPer or the original methods of HemoPI and ToxinPred in their respective test sets.

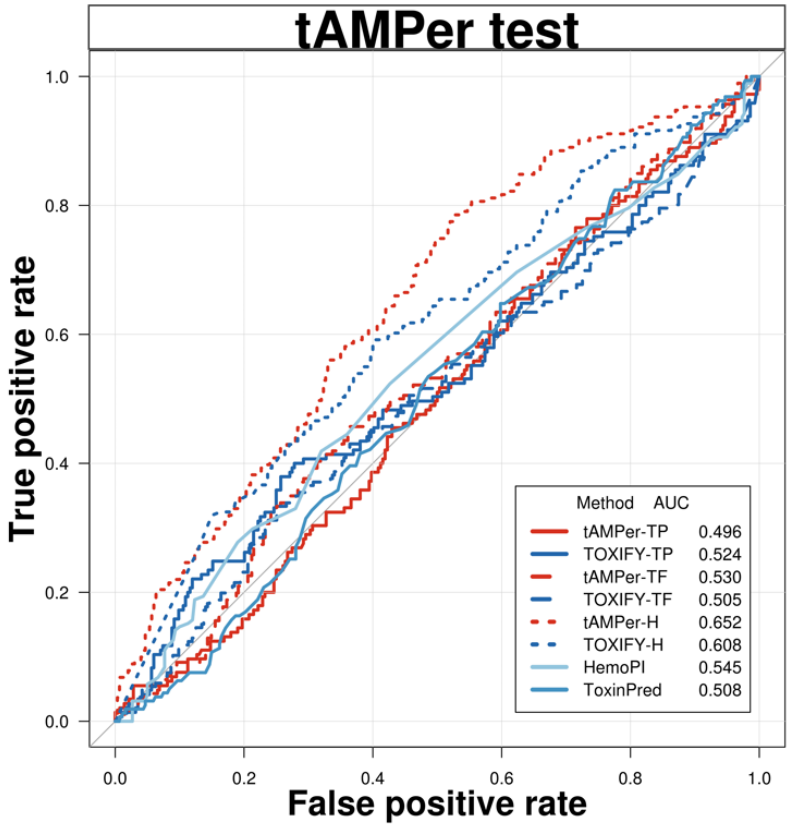

Figure 18: ROC plot of performance of all re-trained TOXIFY and tAMPer models on the tAMPer test set.

The re-trained tAMPer and TOXIFY models do not perform well on the tAMPer test set, which alludes to the fact that the datasets used for the training of the other methods cannot learn enough to make predictions of high accuracy about the tAMPer dataset.

Table 11: Performance of all re-trained tAMPer and TOXIFY models on the tAMPer test set.

Model	Sensitivity	Specificity	Precision	F1 Score	AUROC	Accuracy
HemoPI*	0.445	0.640	0.440	0.443	0.545	0.564
ToxinPred*	0.038	0.938	0.231	0.065	0.492	0.640
TOXIFY-TF	0.511	0.454	0.342	0.409	0.495	0.474
TOXIFY-TP	0.138	0.898	0.408	0.206	0.476	0.641
TOXIFY-H	0.607	0.557	0.466	0.527	0.608	0.576
TOXIFY-t*	0.668	0.740	0.623	0.645	0.773	0.712
tAMPer-H	0.654	0.567	0.49	0.561	0.652	0.601
tAMPer-TF	0.667	0.251	0.331	0.442	0.470	0.399

tAMPer-TP	0.062	**0.958**	0.429	0.108	0.504	0.655
tAMPer-t*	**0.755**	0.848	**0.761**	**0.758**	**0.873**	**0.811**

3.3 Biological relevance of using tAMPer

Figure 19 illustrates the P2_WT sequence heatmap indicating the deviation of the score of the mutated peptides, with the increasing red colour intensity meaning the new mutated peptide has a higher toxicity prediction score, and increasing blue colour intensity meaning the new mutated peptide is less toxic than the original AMP. This display allows us to draw ideas about what are the best ways to mutate sequences by visualizing score change patterns. Horizontal patterns indicate the benefits or disadvantages of using an amino acid, and vertical patterns offer insight into the key positions in a peptide, with consistent changes in mutations' scores indicating importance of the position or residue for the overall toxicity of the peptide. Since the y-axis in these heatmap plots indicates the 20 amino acids in decreasing order of hydrophilicity based on the Hopp and Woods scale [59], from Figure 20 we can observe that using the less hydrophilic residues in single point mutations creates more toxic sequences than single point mutations using the polar and more hydrophilic residues. Additionally, a row-wise observation into Figure 20 suggests that using proline (P) for single point substitutions generally decreases the toxicity score (Figure 23, Figure 24, Figure 25), while tryptophan (W) substitutions generally increase the toxicity score (Figure 21, Figure 22, Figure 23, Figure 26). At the same time, mutating this amino acid in a sequence (Figure 25) decreases the toxicity score. A similar observation holds for proline (P) substitutions in a sequence (Figure 21, Figure 23, Figure 25, Figure 26), where the prediction score generally increases if proline is substituted. Sequence-wise, it can be noted than single point mutations in N terminus of the sequence makes them less toxic (Figure 22, Figure 23, Figure 25). Notably, the sequences that exhibit this pattern have

hydrophobic residues in the N terminus, and hydrophilic in the C terminus. Additionally, some insight into the structural configurations and toxicity can be obtained from the observation that there are clearly repeating score increases and decreases along some sequences, alluding the presence of α helix structures in the sequence, which can be confirmed with the SABLE secondary structure predictions [60] in the figures (Figure 24, Figure 26). These figures suggest that substitutions on the hydrophobic side of α helices decrease the toxicity of the peptide.

It has been suggested that increasing levels of hydrophobicity are associated with mammalian cell toxicity, in addition to increased activity against microbial targets due to increased membrane lysis [7, 61-63]. These increased toxicity observations due to hydrophobic amino acid substitutions are also reflected in the presented heatmaps, highlighting the utility of the tAMPer method for AMP toxicity prediction. However, these results remain to be validated.

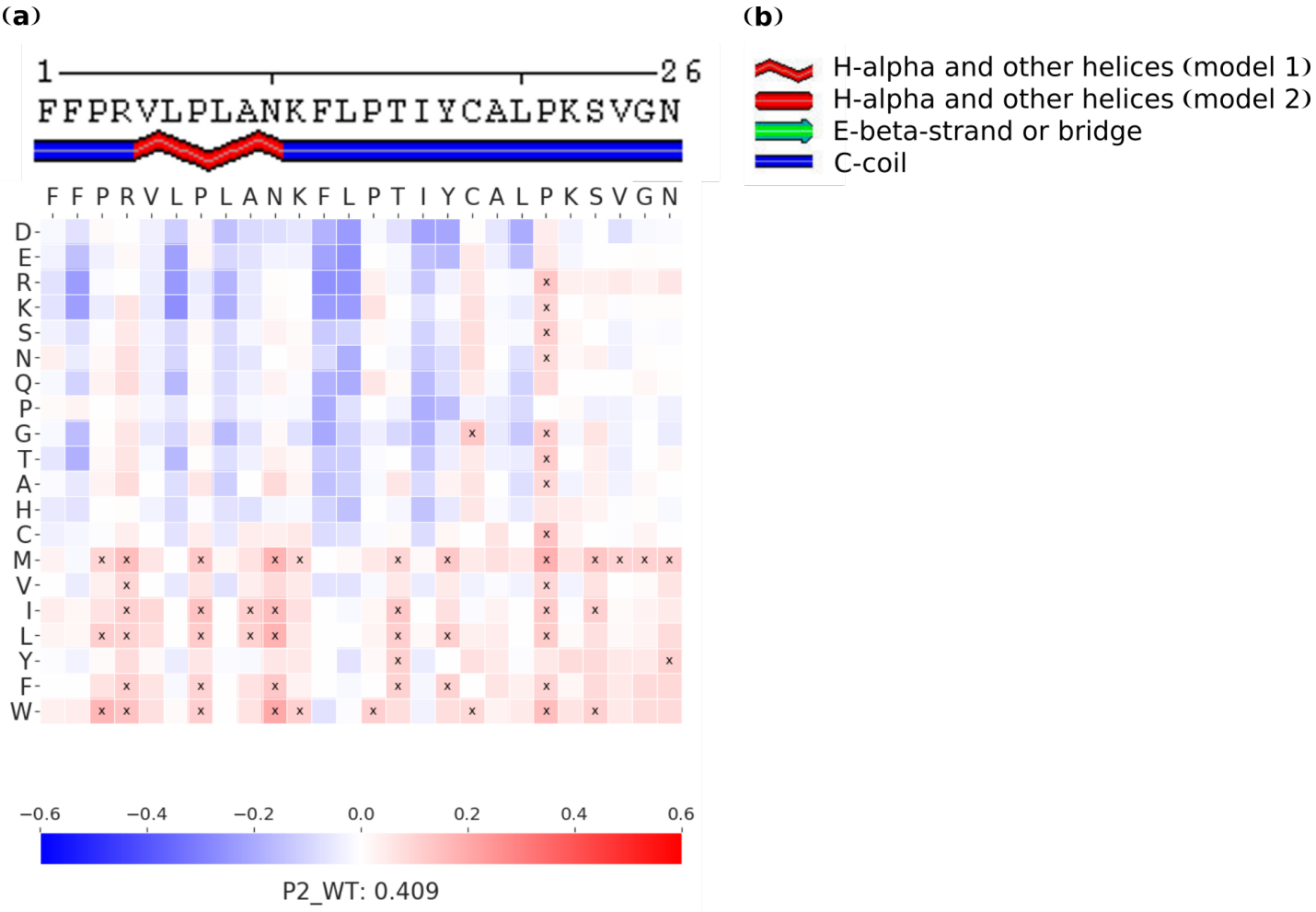

Figure 19: Example heatmap of P2_WT for showing biological significance of tAMPer.

(a) The y-axis lists the 20 amino acids, ordered based on decreasing hydrophilicity according to the Hopp and Woods scale [59]. The "x" symbol annotates tAMPer scores that are > 0.5. The colorbar legend also describes the name of the peptide, as well as the peptide tAMPer score. At the top x-axis, the SABLE [60] secondary structure prediction is depicted, with SABLE reference legend provided in part (b).

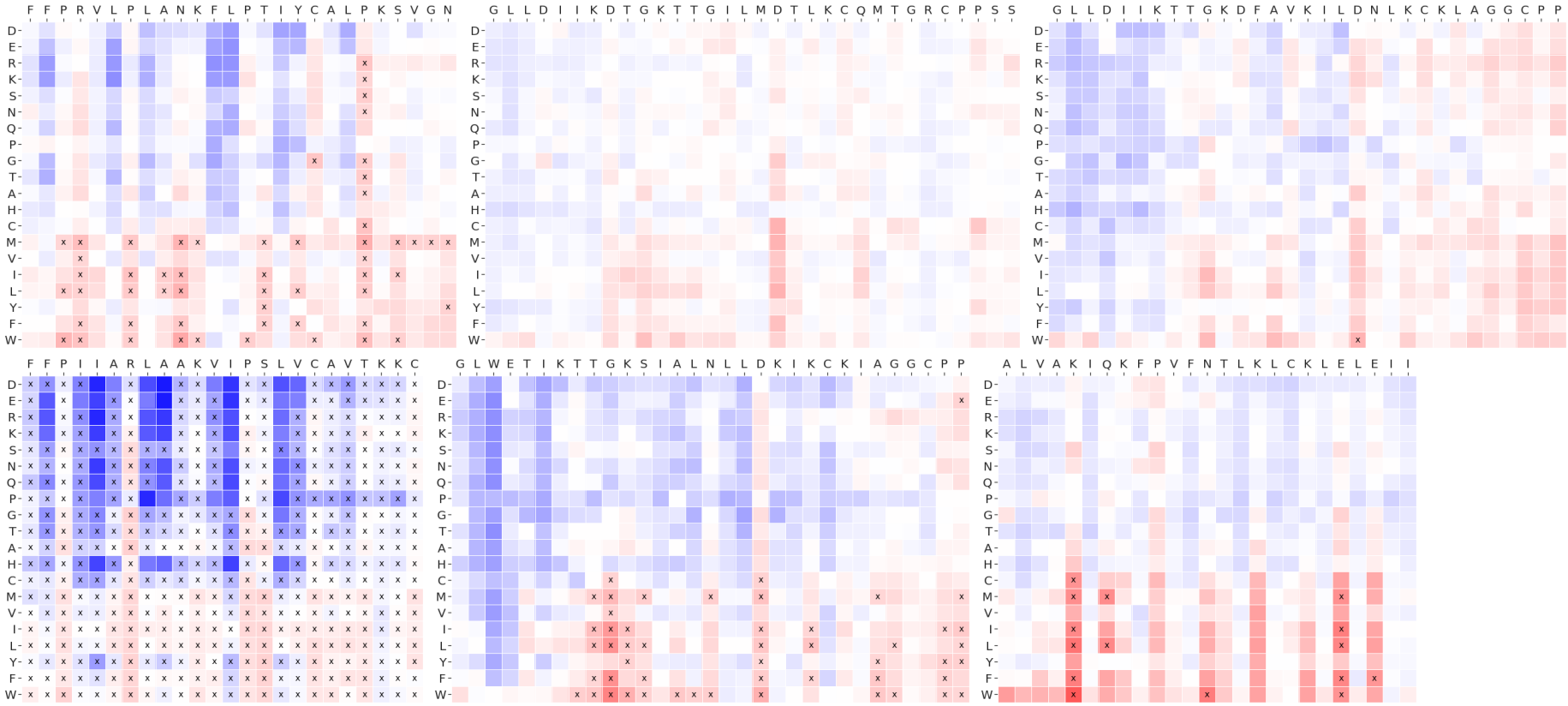

Figure 20: All WT AMP heatmaps (in order left to right, top to bottom, as P2-P6, P11).

The most common pattern discernible from these heatmaps is that performing single amino acid mutations using hydrophobic residues increases the prediction score of tAMPer. Similarly, when hydrophobic residues are substituted with more hydrophilic residues in the peptides, the tAMPer score tends to decrease.

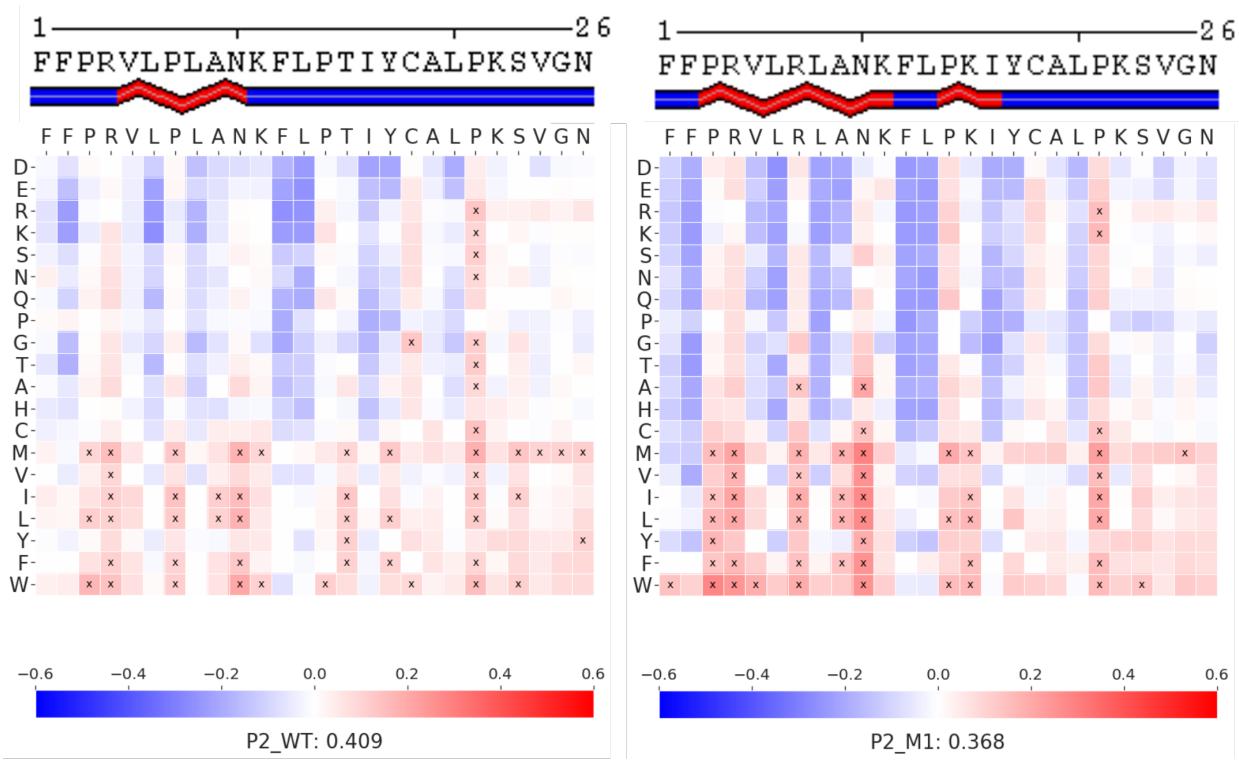

Figure 21: Heatmap for P2_WT and P2_M1.

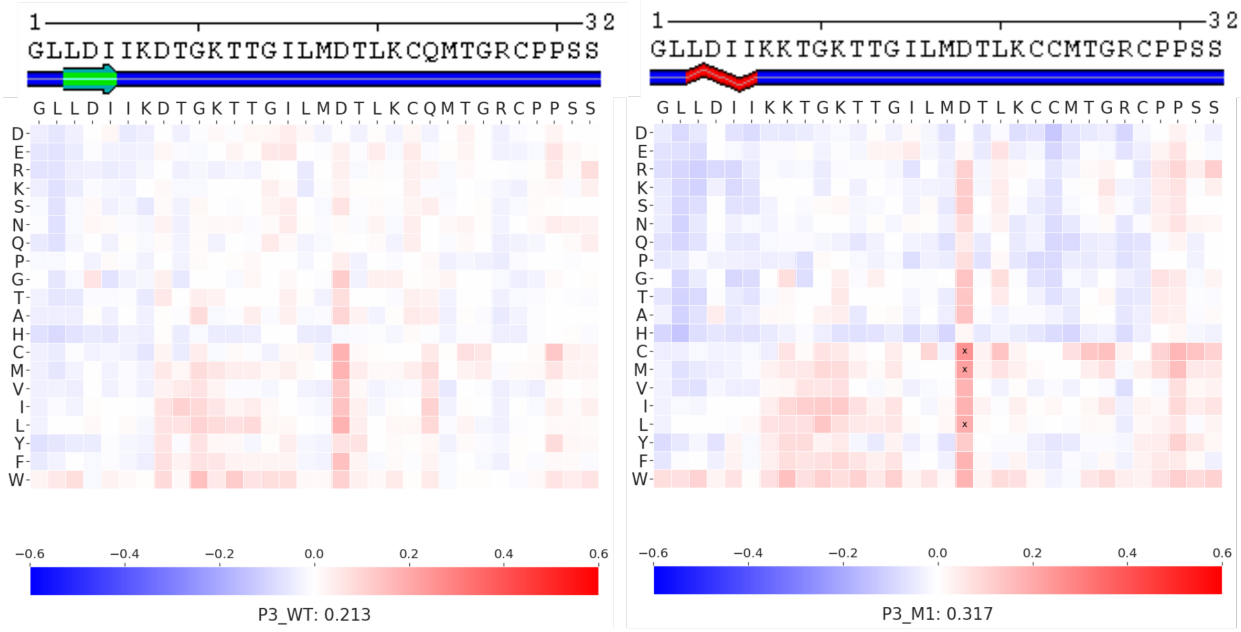

Figure 22: Heatmap for P3_WT and P3_M1.

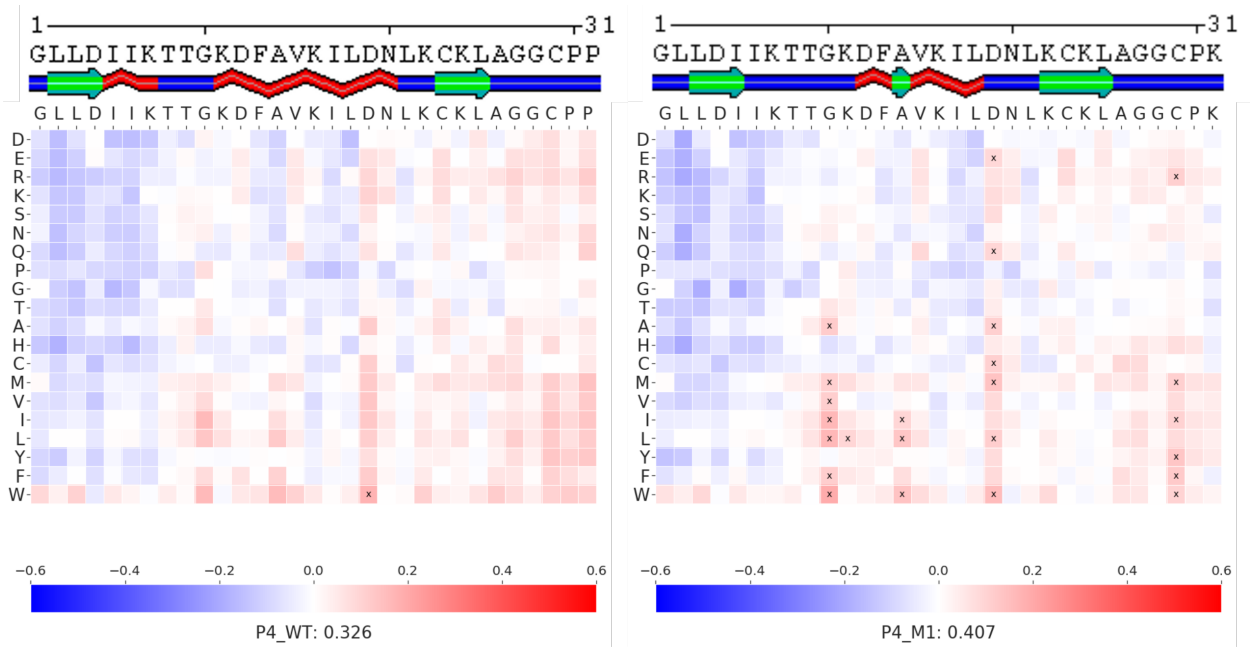

Figure 23: Heatmap for P4_WT and P4_M1.

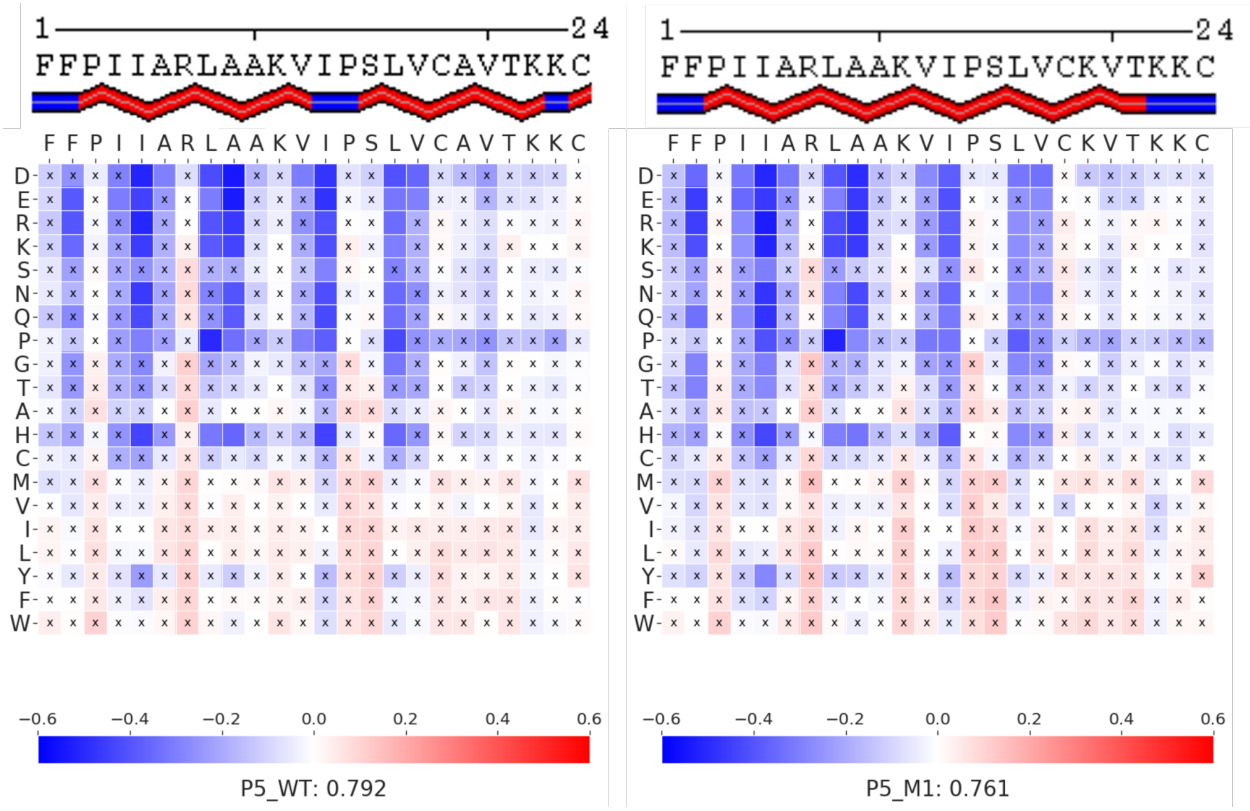

Figure 24: Heatmap for P5_WT and P5_M1.

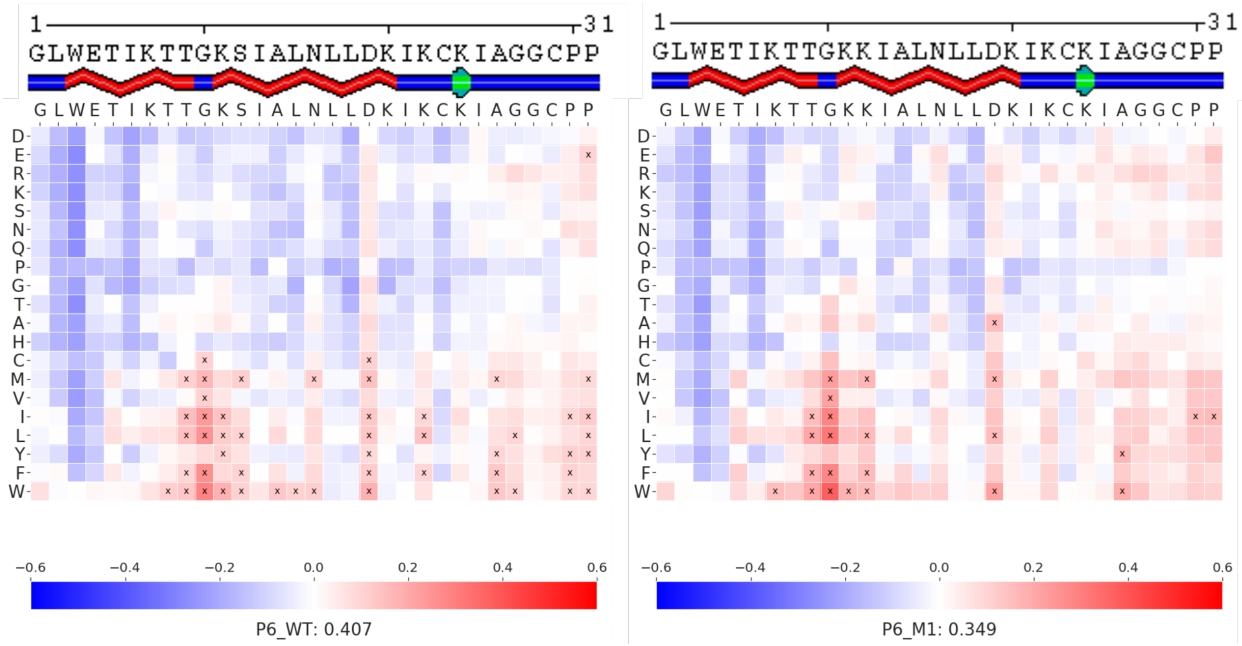

Figure 25: Heatmap for P6_WT and P6_M1.

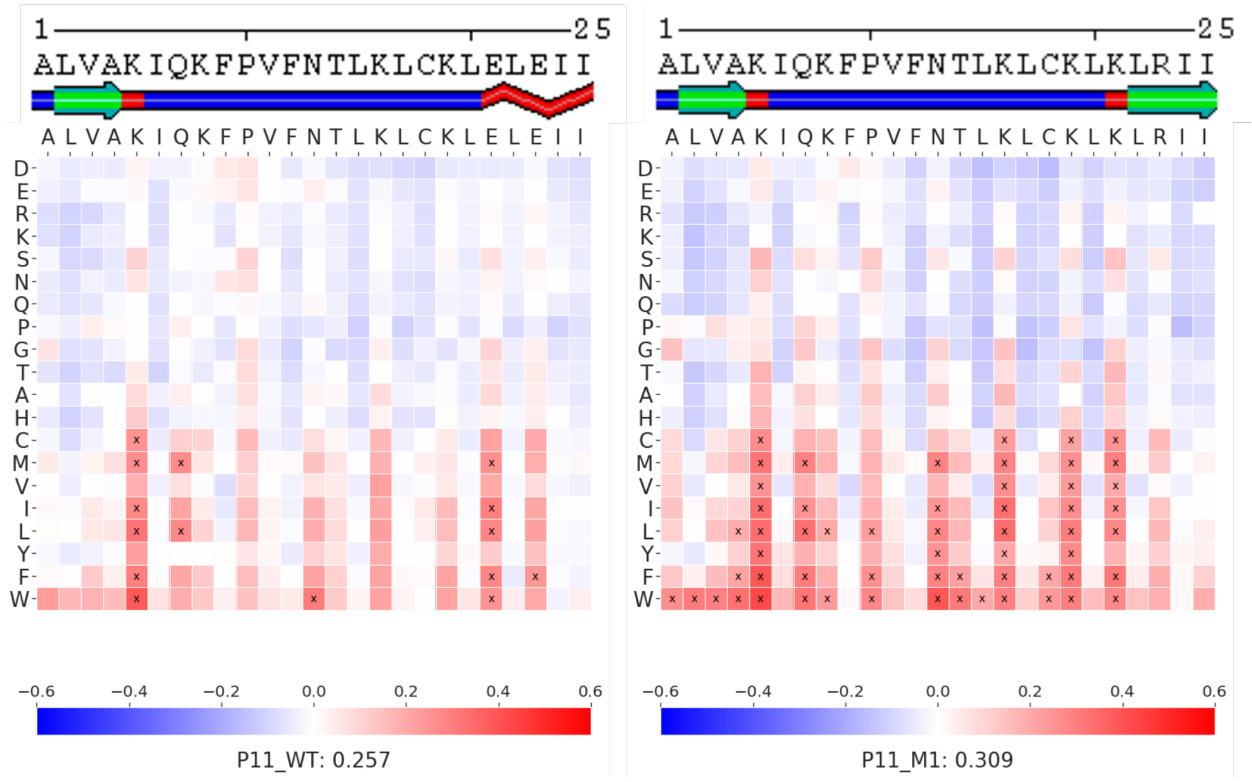

Figure 26: Heatmap for P11_WT and P11_M1.

3.4 tAMPer ensemble score interpretation

The tAMPer ensemble prediction score was plotted against the XX_{50} concentration of the sequences, as available and collected by the database DBAASP v.2 in Figure 27. A Pearson correlation test indicates there is negative correlation with a p-value of 0.01176. This allows the interpretation that a higher tAMPer score is likely indicative of increased host toxicity.

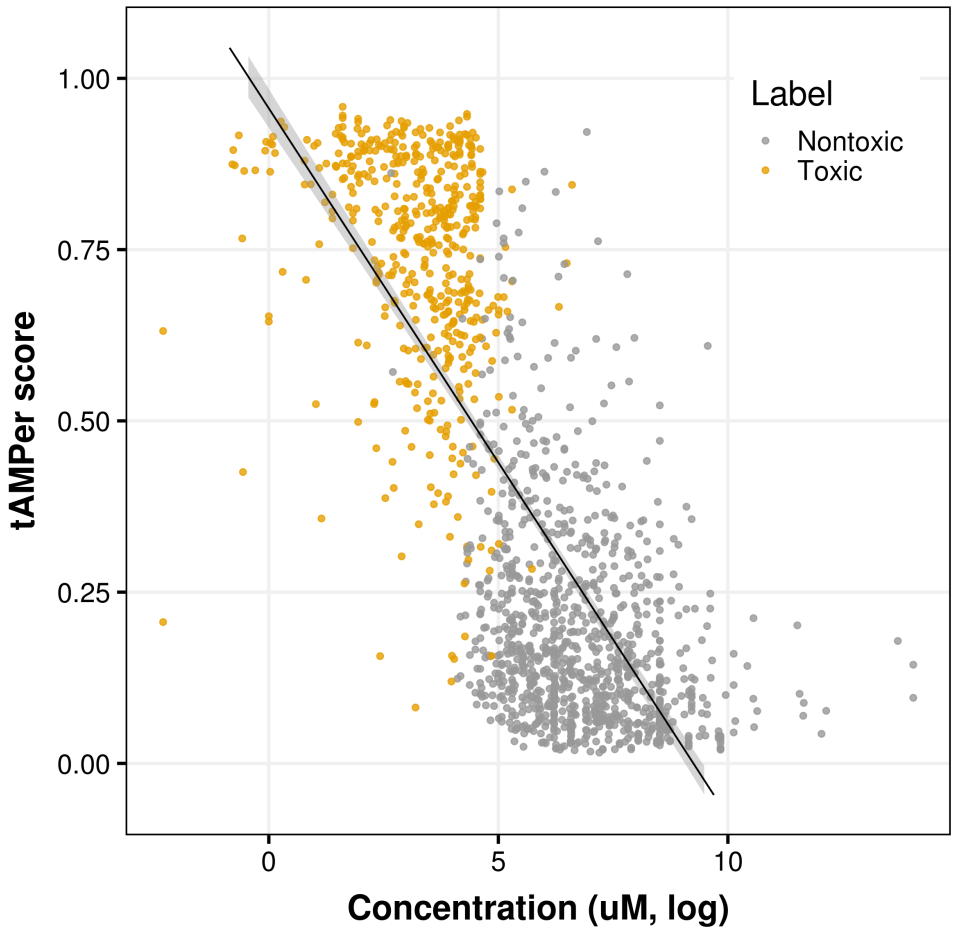

Figure 27: Toxicity concentration (uM, log scale) vs. tAMPer ensemble score.

The regression line implements a linear model fit to the underlying scatterplot data.

Chapter 4: Discussion

tAMPer is a machine learning model for predicting host toxicity of antimicrobial peptides. It achieves a better performance than comparator methods in the literature, and characterizes AMP host toxicity through its prediction score. The data collection and curation steps lead to a dataset that provides a challenge for existing approaches on protein toxicity, both when performing static model predictions, as in the case of HemoPI and ToxinPred, and when re-training existing models (TOXIFY). tAMPer outperforms existing methods of protein toxicity classification, and achieves similar or better performance than these methods when trained and evaluated on their training and test sets.

The biological utility of tAMPer, demonstrated through single and ≤ 3 amino acid mutations, would present its value for potential use in a large AMP mining and validation study, where filtering of sequences of undesired properties is crucial to the downstream operations due to reduced cost, time and effort. Additionally, tAMPer can be used for designing peptides of desired tentative toxicity by mutating existing active AMPs based on the prediction score outputs, e.g. greedily mutating residues for increased activity and decreased host toxicity. The agreement of the score changes due to the single amino substitutions to the existing literature about hydrophobicity in peptides provides some confirmation that this is a valid approach for mutations. Further, the negative correlation of the tAMPer score to concentration can serve as a proxy for mammalian cell toxicity concentration predictions. Existing methods were shown to perform worse in predicting tAMPer test data peptide labels, so its potential for direct application to AMPs makes the method unique in the field.

Limitations of the tAMPer approach arise from the currently lacking standard in both toxicity experimentation and reporting, and this could well mean that the curated data is biased.

Secondly, tAMPer is currently not capable of distinguishing the kind of host toxicity the peptide may induce, i.e. whether the peptide may be hemolytic or cytotoxic against some specific cell line. However, no other method is capable of such host toxicity prediction. And thirdly, the lack of abundant data for training is definitely a setback in the approach, though it is expected as the experimental toxicity data is still not very large. The extensive use of ensemble approaches is aimed at bettering predictions by considering many independent learners, as is the employment of SeqVec embeddings aimed at overcoming the small sample size of the data. As more data becomes available, there ought to be room for improvements in tAMPer host toxicity predictions.

In this thesis, I have presented the work on an ensemble model for AMP host toxicity prediction, which has fulfilled the objective of achieving better performance than the current state-of-the-art methods on a unique curated dataset of AMPs. I foresee that tAMPer would be used by many researchers that study antimicrobial peptides and AMP host toxicity.

Bibliography

1. World Health Organization, *Antimicrobial resistance: global report on surveillance* 2014.
2. Nathan, C. and O. Cars, *Antibiotic resistance--problems, progress, and prospects.* N Engl J Med, 2014. **371**(19): p. 1761-3.
3. CDC, *Antibiotic Resistance Threats in the United States*. 2019, U.S. Department of Health and Human Services: Atlanta, GA.
4. Bahar, A.A. and D. Ren, *Antimicrobial peptides.* Pharmaceuticals (Basel), 2013. **6**(12): p. 1543-75.
5. Spohn, R., et al., *Integrated evolutionary analysis reveals antimicrobial peptides with limited resistance.* Nat Commun, 2019. **10**(1): p. 4538.
6. Hancock, R.E. and H.G. Sahl, *Antimicrobial and host-defense peptides as new anti-infective therapeutic strategies.* Nat Biotechnol, 2006. **24**(12): p. 1551-7.
7. Yeaman, M.R. and N.Y. Yount, *Mechanisms of antimicrobial peptide action and resistance.* Pharmacol Rev, 2003. **55**(1): p. 27-55.
8. Andersson, D.I., D. Hughes, and J.Z. Kubicek-Sutherland, *Mechanisms and consequences of bacterial resistance to antimicrobial peptides.* Drug Resist Updat, 2016. **26**: p. 43-57.
9. Zhang, L.J. and R.L. Gallo, *Antimicrobial peptides.* Curr Biol, 2016. **26**(1): p. R14-9.
10. QLaboratories. *Minimum Inhibitory (MIC) and Minimum Bactericidal Concentration (MBC) Evaluations as R&D Tools*. 2017 [cited 2020 16 June]; Available from: https://www.qlaboratories.com/minimum-inhibitory-mic-and-minimum-bactericidal-concentration-mbc-evaluations-as-rd-tools/.
11. Meher, P.K., et al., *Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC.* Sci Rep, 2017. **7**: p. 42362.
12. Veltri, D., U. Kamath, and A. Shehu, *Deep learning improves antimicrobial peptide recognition.* Bioinformatics, 2018. **34**(16): p. 2740-2747.
13. Xiao, X., et al., *iAMP-2L: a two-level multi-label classifier for identifying antimicrobial peptides and their functional types.* Anal Biochem, 2013. **436**(2): p. 168-77.
14. Castanho, M.A.R.B. and N.C. Santos, *Peptide drug discovery and development translational research in academia and industry*. 2011, Weinheim: Wiley-VCH.
15. Chen, Y., et al., *Rational design of alpha-helical antimicrobial peptides with enhanced activities and specificity/therapeutic index.* J Biol Chem, 2005. **280**(13): p. 12316-29.
16. Chaudhary, K., et al., *A Web Server and Mobile App for Computing Hemolytic Potency of Peptides.* Sci Rep, 2016. **6**: p. 22843.
17. Gupta, S., et al., *In silico approach for predicting toxicity of peptides and proteins.* PLoS One, 2013. **8**(9): p. e73957.
18. Wikipedia. *Support Vector Machine*. [cited 2020 16 June]; Available from: https://en.wikipedia.org/wiki/Support_vector_machine.
19. Cole, T.J. and M.S. Brewer, *TOXIFY: a deep learning approach to classify animal venom proteins.* PeerJ, 2019. **7**: p. e7200.

20. Cho, K., et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. arXiv e-prints, 2014. arXiv:1406.1078.
21. Atchley, W.R., et al., *Solving the protein sequence metric problem.* Proc Natl Acad Sci U S A, 2005. **102**(18): p. 6395-400.
22. Gautam, A., et al., *Hemolytik: a database of experimentally determined hemolytic and non-hemolytic peptides.* Nucleic Acids Res, 2014. **42**(Database issue): p. D444-9.
23. Edgar, R.C., *MUSCLE: a multiple sequence alignment method with reduced time and space complexity.* BMC Bioinformatics, 2004. **5**(1): p. 113.
24. Stamatakis, A., *RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies.* Bioinformatics, 2014. **30**(9): p. 1312-1313.
25. Mikolov, T., et al. *Efficient Estimation of Word Representations in Vector Space*. arXiv e-prints, 2013. arXiv:1301.3781.
26. Peters, M.E., et al. *Deep contextualized word representations*. arXiv e-prints, 2018. arXiv:1802.05365.
27. Heinzinger, M., et al., *Modeling aspects of the language of life through transfer-learning protein sequences.* BMC Bioinformatics, 2019. **20**(1): p. 723.
28. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors.* Nature, 1986. **323**: p. 533.
29. Karpathy, A., J. Johnson, and L. Fei-Fei *Visualizing and Understanding Recurrent Networks*. arXiv e-prints, 2015. arXiv:1506.02078.
30. Goldberg, Y., *Vanishing and exploding gradients*, in *Neural Network Methods in Natural Language Processing*. 2017, Morgan & Claypool Publishers. p. 59-60.
31. Blog, C.s., *Understanding LSTM Networks*. 2015.
32. Socher, R., *Language Models*, in *CS224D: Deep Learning for NLP*. 2015.
33. Pirtskhalava, M., et al., *DBAASP v.2: an enhanced database of structure and antimicrobial/cytotoxic activity of natural and synthetic peptides.* Nucleic Acids Res, 2016. **44**(13): p. 6503.
34. Thomas, S., et al., *CAMP: a useful resource for research on antimicrobial peptides.* Nucleic Acids Res, 2010. **38**(Database issue): p. D774-80.
35. Wang, G., X. Li, and Z. Wang, *APD3: the antimicrobial peptide database as a tool for research and education.* Nucleic Acids Res, 2016. **44**(D1): p. D1087-93.
36. Piotto, S.P., et al., *YADAMP: yet another database of antimicrobial peptides.* Int J Antimicrob Agents, 2012. **39**(4): p. 346-51.
37. pydata.org. *Pandas*. 2020 [cited 2020 16 June]; Available from: https://pandas.pydata.org/.
38. Wilkins, M.R., et al., *Protein identification and analysis tools in the ExPASy server.* Methods Mol Biol, 1999. **112**: p. 531-52.
39. Wikipedia. *Word2vec*. 2020 [cited 2020 16 June]; Available from: https://en.wikipedia.org/wiki/Word2vec.
40. Asgari, E. and M.R. Mofrad, *Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics.* PLoS One, 2015. **10**(11): p. e0141287.
41. Alley, E.C., et al., *Unified rational protein engineering with sequence-based deep representation learning.* Nat Methods, 2019. **16**(12): p. 1315-1322.
42. Kim, Y., et al. *Character-Aware Neural Language Models*. arXiv e-prints, 2015. arXiv:1508.06615.

43. Suzek, B.E., et al., *UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches.* Bioinformatics, 2015. **31**(6): p. 926-32.
44. Breiman, L., *Random Forests.* Machine Learning, 2001. **45**(1): p. 5-32.
45. Friedman, J.H., *Greedy Function Approximation: A Gradient Boosting Machine.* The Annals of Statistics, 2001. **29**(5): p. 1189-1232.
46. Schmidt, M., N. Le Roux, and F. Bach *Minimizing Finite Sums with the Stochastic Average Gradient*. arXiv e-prints, 2013. arXiv:1309.2388.
47. Wolpert, D.H., *Stacked generalization.* Neural Networks, 1992. **5**(2): p. 241-259.
48. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
49. Pedregosa, F., et al., *Scikit-learn: Machine Learning in Python.* J. Mach. Learn. Res., 2011. **12**(null): p. 2825–2830.
50. Chollet, F. *Keras*. 2015; Available from: https://keras.io.
51. Kingma, D.P. and J. Ba *Adam: A Method for Stochastic Optimization*. arXiv e-prints, 2014. arXiv:1412.6980.
52. Glorot, X. and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks JMLR W\&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*. Chia Laguna Resort, Sardinia, Italy.
53. Distribution, A.S., *Vers. 2-2.4.0. Anaconda*. 2016.
54. Bojanowski, P., et al. *Enriching Word Vectors with Subword Information*. arXiv e-prints, 2016. arXiv:1607.04606.
55. Krause, B., et al. *Multiplicative LSTM for sequence modelling*. arXiv e-prints, 2016. arXiv:1609.07959.
56. Sutskever, I., J. Martens, and G. Hinton, *Generating text with recurrent neural networks*, in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, Omnipress: Bellevue, Washington, USA. p. 1017–1024.
57. Li, C., et al., *AMPlify: attentive deep learning model for discovery of novel antimicrobial peptides effective against WHO priority pathogens.* bioRxiv, 2020: p. 2020.06.16.155705.
58. World Health Organization. *WHO publishes list of bacteria for which new antibiotics are urgently needed*. 2017; Available from: https://www.who.int/en/news-room/detail/27-02-2017-who-publishes-list-of-bacteria-for-which-new-antibiotics-are-urgently-needed.
59. Hopp, T.P. and K.R. Woods, *Prediction of protein antigenic determinants from amino acid sequences.* Proceedings of the National Academy of Sciences, 1981. **78**(6): p. 3824-3828.
60. Adamczak, R., A. Porollo, and J. Meller, *Combining prediction of secondary structure and solvent accessibility in proteins.* Proteins, 2005. **59**(3): p. 467-75.
61. SD, S., *NET CHARGE, HYDROPHOBICITY AND SPECIFIC AMINO ACIDS CONTRIBUTE TO THE ACTIVITY OF ANTIMICROBIAL PEPTIDES.* Journal of Health and Translational Medicine, 2014(1): p. 1-7%V 17.
62. Wieprecht, T., et al., *Peptide hydrophobicity controls the activity and selectivity of magainin 2 amide in interaction with membranes.* Biochemistry, 1997. **36**(20): p. 6124-32.

63. Schmidtchen, A., M. Pasupuleti, and M. Malmsten, *Effect of hydrophobic modifications in antimicrobial peptides.* Adv Colloid Interface Sci, 2014. **205**: p. 265-74.