Chapter 3

A multi-layered continual-learning based architecture for discovering antibacterial peptides

The prevalence of infections resulting from multi-drug resistant (MDR) microorganisms, along with the limited availability of effective antibiotics, has compelled researchers to explore the application of *in-silico* machine and deep learning methods for expediting the drug discovery process. The utilization of computational models for identifying antibacterial peptides (ABPs) in proteins of diverse organisms, with the aim of creating novel antibiotics, has emerged as a potential avenue of research. In order to achieve this objective, we employed multi-scale temporal convolutional networks (MSTCN) as the basis for constructing a continual learning (CL)-based deep learning model known as MSTCN-ABPpred. This model demonstrates a classification accuracy of 98%, surpassing the performance of several contemporary models. The primary contribution of this study is in integrating a continual learning module with the proposed model, enabling it to adapt dynamically to new data points by getting re-trained on them. This modified version of the baseline model is called MSTCN-ABPpred (CL). As a pilot study,

the proposed model was retrained using the ABPs and non-ABPs predicted by it in some antibacterial proteins. The empirical evidence suggests that the proposed model does not display any statistically significant decline in performance (due to catastrophic forgetting) following extensive re-training. Furthermore, it acquires additional abilities compared to the initial model. Also, a web application has been implemented and deployed that is freely accessible at `https://mstcn-abppred.anvil.app/`. It identifies ABPs in a given protein, and the underlying model also undergoes retraining in the process.

3.1 Introduction

Following the onset of the 20^{th} century, a notable trend emerged wherein various tasks began to undergo automation. It has become imperative for the scientific community to prioritize research on the expeditious discovery of alternate antibiotics to fight the emergence of antimicrobial resistance (AMR) in bacteria. In this light, developing a new line of drugs based on antibacterial peptides (ABPs) appears to be a plausible recourse. The ABPs are biomolecules synthesized by various cells and tissues across a wide range of living species, including plants, animals, bacteria, and fungi. These peptides safeguard us against harmful disease-causing bacteria. However, the identification of ABPs in living organisms using traditional laboratory methods is both expensive and time-intensive [43].

In recent times, several machine/ deep learning-based solutions have been put forth for finding ABPs in proteins of various living beings, leading to the genesis of various classifiers like StaBle-ABPpred [44], ABPDiscover [45], AMAP [46], Deep-ABPpred [3], iAMPpred [4], MLAMP [47], AMPFUN [48], iAMP-CA2L [49], iAMP-2L [50], AntiBP2 [30], etc. Authors of AMAP [46], iAMPpred [4], Antibp2 [30] built a classifier using support vector machine (SVM). The iAMPpred model [4] predicts peptides as ABPs, AVPs, and AFPs, and the AMAP model [46] gives the predicted probability values for

a variety of functional activities, one of which is the peptide's propensity to act like an ABP. The AMPFUN model [48] is a two-stage classification model where antimicrobial/ non-antimicrobial classification happens in the first stage, and the second stage comprises further classification of predicted AMPs into subcategories (antibacterial, antifungal, antiparasitic, etc.). These approaches employed the conventional machine learning algorithms, which have the inherent shortcoming of being unable to perform better than deep learning methods on large datasets.

Moreover, many deep learning models like StaBle-ABPpred [44], Deep-ABPpred [3], and RNN-ABPD [45] have also been proposed based on recurrent neural networks (RNNs), bidirectional long short-term memory (biLSTM), etc. The authors of [45] developed three models, namely, ABPDiscover (HABPD), Hierarchical ABPDiscover (HABPD), and RNN-ABPDiscover (RNN-ABPD) using random forests (RFs) and RNNs. The Deep-ABPpred classifier [3] is a biLSTM-based model built for ABP classification. However, algorithms like RNN or bi-LSTM have sequential modes of operation. So, training and re-training these models consumes a lot of time and resources. The StaBle-ABPpred [44] uses a stacked ensemble method along with the attention mechanism [39] based biLSTM framework, whose output is utilized by an ensemble of three machine learning algorithms to perform an ABP classification. This stacked framework requires the biLSTM part to train for fewer epochs, decreasing the overall training time. However, a common drawback of StaBLe-ABPpred and all the approaches listed before is that they do not adapt to the new data points that emerge over time, which would affect their performance in the coming years (the static nature of the models will make them lose their relevance in a few years). Even if StaBle-ABPpred and Deep-ABPpred had this ability, re-training them would affect the real-time deployability of the apps based on them because they are based on the bi-LSTM algorithm (which would consume a lot of time while re-training).

There is a need for a model that can adapt to new data points while classifying and

discovering novel ABPs. In this chapter, one such continual learning (CL)-based model called **M**ulti-**S**cale **T**emporal **C**onvolutional **N**etworks-based **A**nti**B**acterial **P**eptide **pred**iction (MSTCN-ABPpred) has been proposed to serve this purpose. It takes peptide sequences (each of which is a chain of standard amino acids (AAs) represented by all the English letters except *B, J, O, U, X,* and *Z*) and uses temporal convolutional networks (TCNs) [51, 52] to classify them. TCNs are dilated one-dimensional convolutional neural networks (CNNs)-based deep learning frameworks used for sequence modeling. The MSTCN-ABPpred model uses a multi-scale variant of TCNs, where filters of varying sizes produce numerous feature maps. This modification enables the model to effectively capture both short and long-range dependencies within a given peptide.

A notable characteristic of this model is the use of CL for its adaptation to novel data points. CL is employed to enable a model to assimilate novel information (referred to as plasticity) while preserving its proficiency in previously acquired knowledge (referred to as stability) [53]. The model must be able to remain stable while showing plasticity to mitigate the risk of catastrophic forgetting, which refers to the phenomenon when the model becomes unstable and loses its previously gained information [54]. With the stability-plasticity trade-off as a guiding principle, we have put forth a baseline model, namely MSTCN-ABPpred (BL), and have explored four different techniques to enable its adaptability to new data using continual learning. After careful consideration, the most suitable technique was chosen to re-train the model, which was then named MSTCN-ABPpred (CL). In order to assist the researchers in the wet lab, a web application that is freely available online has been deployed and can be accessed at `https://mstcn-abppred.anvil.app/`. Its primary functionality includes the classification and identification of ABPs and non-ABPs. Due to the limited availability of experimentally confirmed ABPs in publicly accessible databases, a novel approach has been used for re-training the model that involves using synthetic

data points consisting of the peptides predicted within proteins by the model itself. In order to demonstrate the performance and functionality of the MSTCN-ABPpred (CL) model, we conducted re-training using ABPs and non-ABPs identified in the tail proteins of bacteriophages that target bacteria belonging to the ESKAPEE pathogens. This genera includes Enterococcus faecium, Staphylococcus aureus, Klebsiella pneumoniae, Acinetobacter baumannii, Pseudomonas aeruginosa, Enterobacter spp., and Escherichia coli [55]). These are critical to highly drug-resistant species as per the World Health Organisation (WHO) "priority pathogens list" [56]. Note that bacteriophages are those viruses that either kill or impede the activity of bacteria. The major contributions of the proposed work are as follows.

1. A novel multi-scale model, MSTCN-ABPpred (BL), has been proposed to classify and discover ABPs. Additionally, a novel continual learning module has been incorporated into the proposed model, enabling it to adapt and retrain on new data. The resulting model has been named MSTCN-ABPpred (CL).
2. A novel approach for the generation and utilization of novel data points has been proposed, wherein the peptides classified by the model within protein sequences are employed for its retraining.
3. The MSTCN-ABPpred (BL) model underwent retraining utilizing four distinct continual learning methodologies. Ultimately, the method that outperformed all others, including MSTCN-ABPpred (BL), when evaluated on an independent dataset, was selected. To the best of our current knowledge, this is a novel contribution to the area of *in silico* discovery of therapeutic peptides.
4. The statistical tests demonstrate that the MSTCN-ABPpred (CL) model does not exhibit significant catastrophic forgetting upon re-training. Furthermore, the performance of the proposed model is found to be better than the state-of-the-art models.
5. A web application developed using this framework has been implemented and

made available at `https://mstcn-abppred.anvil.app/`. This application serves the purpose of classifying and identifying ABPs and non-ABPs within protein sequences, which are subsequently employed to retrain the model.

The rest of this chapter is organized as follows. Section 3.2 presents the tools, techniques, and the dataset used in this work. The proposed work is elucidated in section 3.3. The results and outcomes have been elaborated in section 3.4. Lastly, the concluding remarks and future works are discussed in section 3.5.

3.2 Data and preliminaries

3.2.1 Dataset

The ABPs used to train, tune, and test the proposed model were collected from public repositories such as the collection of antimicrobial peptides (CAMP) [7], the StarPep database [34, 57, 58], ANTIMIC database [59], the data repository of antimicrobial peptides (DRAMP) [32], the antimicrobial peptide database (APD) [31], and the milk antimicrobial peptides (MilkAMP) [33] database. The data points for non-ABPs were taken from the UniProt database [35].

Further, the anionic peptides, the peptides containing non-standard amino acids (*B, J, O, U, X, Z*) and the ones having less than four and over thirty AAs were discarded. Hence, the final dataset comprised 5652 ABPs and 7284 non-ABPs. The data points were converted into strings of numbers after tokenization and then split into training (60%), validation (20%), and test (20%) sets. After this, the data points were used to train the skip-gram-based word embedding layer to generate feature vectors of size 200 corresponding to each AA. These were used later to generate a feature matrix corresponding to each peptide and used as an input to the proposed model.

3.2.2 Continual Learning

The concept of continual learning refers to the continuous adaptation and evolution of a model by learning on new tasks, new data, or both [60]. The mentioned approach

enables a model to preserve its learned abilities while also facilitating the gradual expansion of knowledge in response to novel data or tasks [61, 62]. The emergence of this concept can be attributed to the swift obsolescence of static machine learning models, as they lack the capacity to adapt and grow once deployed in an online environment.

A common case when CL is required the most is when there is a need to re-train an existing model to perform the same task on novel data points. As discussed before, maintaining relevance necessitates the model to update itself continuously using fresh data. Nevertheless, acquiring new abilities may occasionally impede the retention of knowledge obtained from previous data points, resulting in catastrophic forgetting or interference. In the most unfavorable scenario, acquiring new knowledge could potentially result in the entire displacement of the model's previously learned knowledge. Therefore, it is crucial to identify an appropriate balance between the stability of the model, which refers to its capacity to retain previously acquired skills, and its plasticity, which pertains to its ability to learn and adapt to new data.

Researchers have offered three potential solutions to the issue of catastrophic forgetting. These include regularisation, exact replay, and generative replay. Regularisation strategies aim to safeguard the crucial weights or parameters, which have a significant impact on the learning process from previous data points when retraining a model with fresh data [63, 64]. These strategies serve to mitigate the occurrence of catastrophic forgetting, but they also impose limitations on a model's capacity to adapt effectively to novel input. The exact replay methods entail the retention of the complete dataset used before and employing it repeatedly while retraining the model on new data points. Finally, the generative replay-based methodologies utilize an efficient generative model, which was trained along with the primary prediction model, to approximate the previously seen data points [65, 66]. These methodologies entail retraining the entire model using both the existing dataset and the newly acquired data. Nevertheless, when the dataset increases rapidly, these approaches may become impractical and computation-

ally burdensome. This phenomenon may result in significant delays in the real-time accessibility of the web applications that are based on these models. To address this matter, retraining a model using a limited portion of the previous dataset seems appropriate. In the studies done by [67, 68, 69, 70], the authors suggest utilizing a small subset of previous data points for memory rehearsal. This approach aims to make the real-time re-training and re-deployment of the model more practically viable. Previous research has demonstrated that this strategy exhibits superior performance compared to approaches based on regularisation [67]. But, the size of the subset has an impact on the balance between stability and plasticity, as well as the potential scalability of the solution in larger-scale environments [71].

3.2.3 Temporal Convolutional Networks (TCNs)

The TCNs are a class of deep learning algorithms used for sequence modeling, which are built upon the CNNs. In contrast to LSTM-based models, the operations of TCN-based models can be parallelized. Also, TCNs have reduced memory demands, effectively recording long-distance dependencies between the elements within a sequence. A TCN block consists of many one-dimensional convolutional (1D-CONV) layers arranged sequentially, where the output of one layer serves as the input for the subsequent layer. When employing residual connections, a layer encompasses two 1D-CONV layers, collectively forming a residual block. Residual connections have a crucial role in enhancing the stability of the learning process and facilitating rapid convergence toward favorable outcomes, particularly in the context of language modeling. According to earlier research, it has been observed that TCNs exhibit limited sensitivity to variations in hyper-parameters, provided that the receptive field size (R_{field}, which refers to the range of the input space to which a specific unit in a 1D-CONV layer is exposed) is adequately large [72]. The application of filter taps on the steps/units of a 1D-CONV layer is done in an interleaved or dilated manner to expand the receptive field. The size of the receptive field is determined by two parameters, namely the dilation size (d) and

the kernel size (k). The calculation of the receptive field size is described by Eq. 3.1 [73].

$$R_{field} = 1 + 2 \cdot (k-1) \cdot \sum_{i} d_i \tag{3.1}$$

In this context, the variable d_i represents the dilation employed within a specific TCN block. There are two forms of TCNs: causal and acausal. Causal TCNs only utilize the preceding layer's prior units or timesteps (namely, 0 through $t-1$) to calculate the value of a given timestep t in the current layer. In essence, the predictions made at a certain stage can only rely on the knowledge obtained from preceding steps. The equation for the dilated causal convolution operation, denoted as $C(t)$, executed at the t^{th} step is provided in Eq. 3.2 [72].

$$C(t) = (x *_d f)(t) = \sum_{i=1}^{k} f(i).x_{t-d.(i-1)} \tag{3.2}$$

In the given context, the variable x is used to denote the input. The symbol $*_d$ denotes the convolution operation. The filter is denoted by $f = f(0), f(1), \cdots, f(k)$. Here, k represents the size of the kernel, and d indicates the dilation size. The convolution is performed from the time step $x_{t-d.(k-1)}$ to x_t to exclude the timesteps beyond t from the analysis. In contrast, acausal TCNs incorporate both past and future timesteps to make predictions at t. The acausal convolutional operation executed for timestep t, denoted as $A(t)$, is defined by Eq. 3.3 [52]. In this scenario, the convolution operation executes from $x_{t-d.\lceil (k-1)/2 \rceil}$ to $x_{t+d.\lfloor (k-1)/2 \rfloor}$.

$$A(t) = (x *_d f)(t) = \sum_{i=1}^{\lfloor k/2 \rfloor} f(\lceil k/2 \rceil + i) \cdot x_{t-(d \cdot i)} + \sum_{i=\lfloor k/2 \rfloor + 1}^{k} f(k-i+1) \cdot x_{t+d.(i-\lfloor k/2 \rfloor + 1)} \tag{3.3}$$

3.3 Proposed Work

The model trained and tested on the dataset described in the previous section is named MSTCN-ABPpred (BL). Subsequently, after integrating the module for continual learning (CL), it is referred to as MSTCN-ABPpred(CL). The description of the two phases of the proposed work, namely the construction of the baseline model and the incorporation of the CL component, have been elucidated in this section.

3.3.1 MSTCN-ABPpred (BL) model

The MSTCN-ABPpred model, illustrated in Figure 3.1, comprises many components, including an embedding layer, spatial dropout layers, TCN blocks, a global average pooling layer, dense layers, a concatenation layer, and dropout layers. The embedding layer uses an embedding matrix formed using the skip-gram technique [74] to transform each input sequence into a matrix consisting of feature vectors, each with a dimensionality of 200. Subsequently, the feature matrix is input into a spatial dropout layer, which aims to identify potential correlations among the various columns of the feature matrix. This acquired information is then utilized to discard certain columns in order to reduce the risk of overfitting. Following this, four parallel extended temporal convolutional network (xTCN) blocks are employed. Each block consists of a TCN residual block, a global average pooling layer, a dense layer with a size of 16, and a dropout layer with a dropout rate of 20%. Each xTCN block produces a set of $f = 100$ feature maps (where f is the number of filters).

The model is called multi-scaled due to the utilization of filters of varying sizes in each xTCN block, as depicted in Figure 3.1. This approach enables the extraction of feature maps, which are subsequently transformed into a one-dimensional vector by a global average pooling layer. The subsequent layers consist of dense and dropout layers, as stated previously. The output vectors of each xTCN block are consolidated into one vector by a concatenation layer, which is subsequently connected to a dense

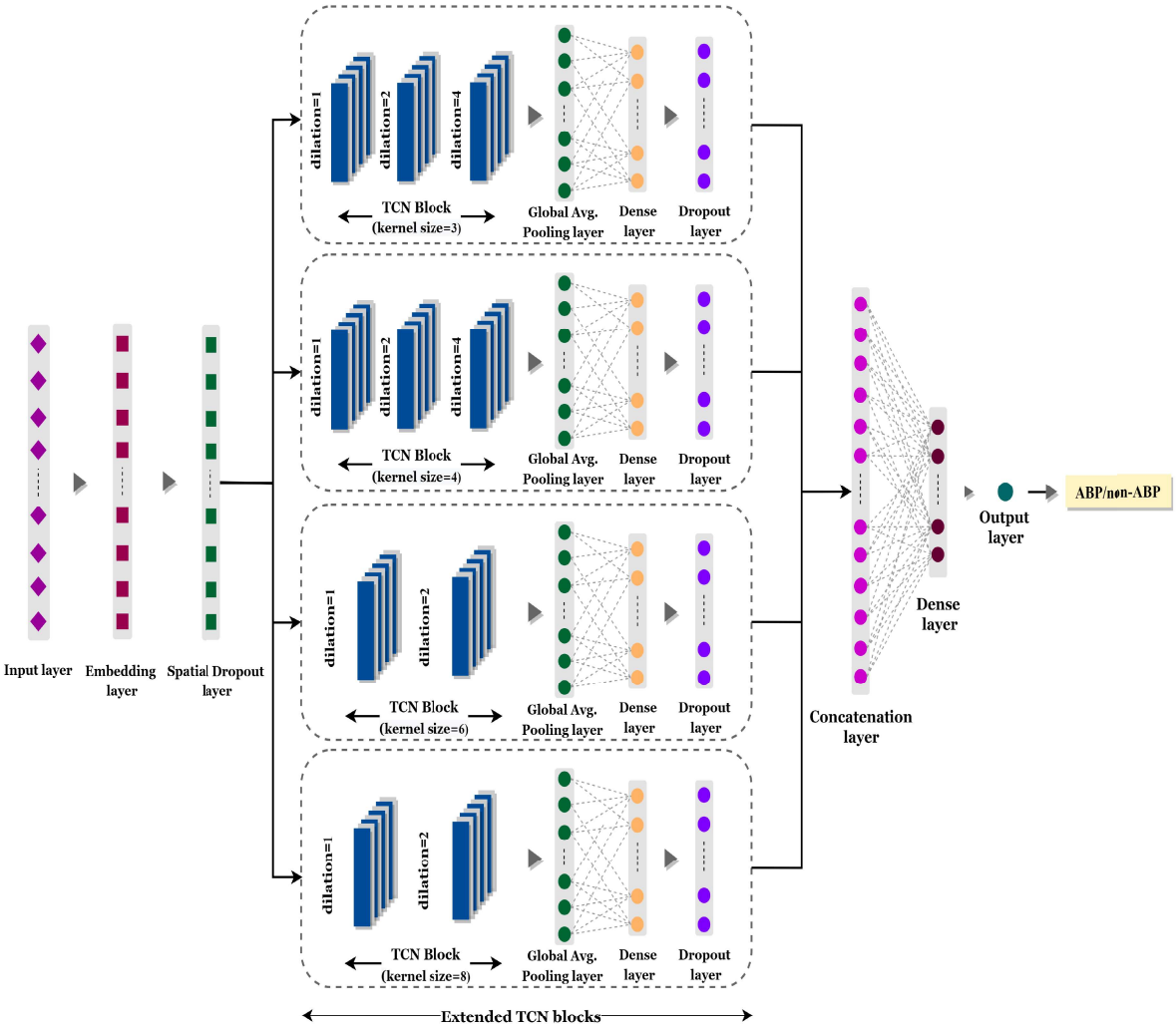

Figure 3.1: Architecture of MSTCN-ABPpred model

layer containing eight neurons. Finally, an output layer is employed. The logistic function is utilized in the output layer to determine the predicted probability value (*score*) of a given peptide. If the value is equal to or greater than 0.5, the peptide is classified as an ABP.

3.3.2 MSTCN-ABPpred (CL) model

Integrating a continual learning (CL) module in the MSTCN-ABPpred (BL) model required addressing two critical questions, which are as follows.

1. **Does the model require re-training on the previously seen data points?**

The model can be adapted to incorporate information about the additional data points, with or without memory rehearsal, on the previously seen datasets. This study examines two potential approaches: re-training the model with and without memory rehearsal. The composition of the re-training set (R) being utilized for memory rehearsals is critical in determining its effectiveness. To mitigate the occurrence of catastrophic forgetting, it is advisable to re-train the model using a combined dataset which consists of the original training set, denoted as T (which was initially used to train MSTCN-ABPpred (BL)), as well as the data points discovered during all the rounds of re-training, denoted as D. Additionally, the new data points should also be included in this combined dataset. As the model undergoes re-training, the size of dataset D is expected to grow, potentially leading to a significant rise in the memory and time required to store and re-train the model on the combined datasets T and D. The aforementioned concern holds significant importance in cases where the re-training process could potentially impact the real-time accessibility of an application constructed using such a model. Therefore, employing a limited number of data points from the sets T and D is a more favorable approach.

2. **Does the model require to re-train in its entirety?** There are two distinct approaches for retraining the model: one involves complete recalibration, while the other entails freezing certain layers, rendering them untrainable, and retraining the remaining layers. It is imperative to thoroughly consider and solve this issue to prevent the model from experiencing a sudden decline in its performance when trained on novel data points [75]. In our research, both alternatives have been thoroughly examined to identify an appropriate compromise between the model's stability and plasticity.

Considering the aforementioned factors, four methods for re-training the model based on continual learning (as depicted in Figure 3.2) were considered, outlined as

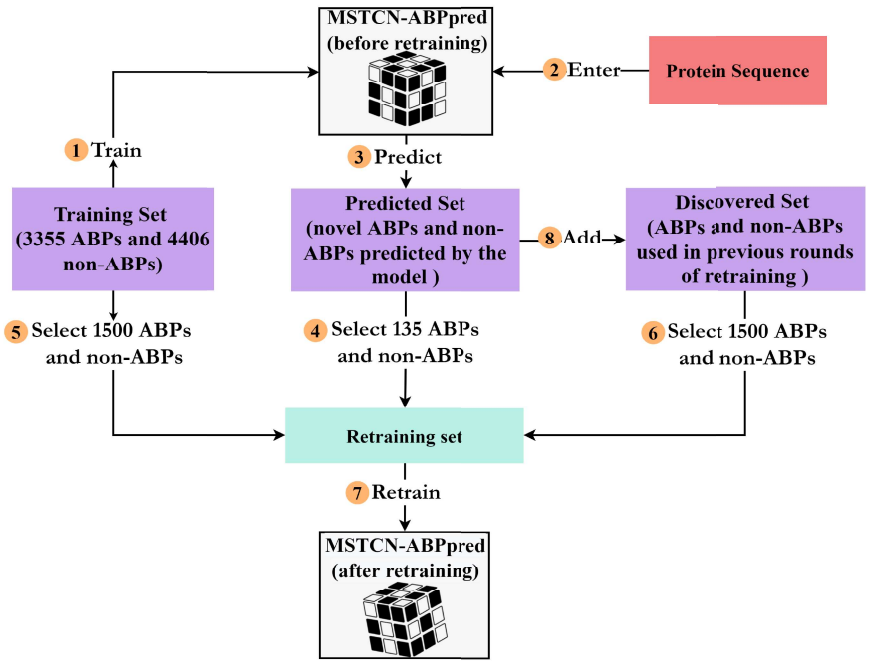

Figure 3.2: The working of the continual learning module. Step 1 is performed only once on the initial training set to build the MSTCN-ABPpred (BL) model. After that, steps 2-8 are repeated for further re-training (Steps 5 and 6 are not executed in case of re-training using approaches 1 and 3). To predict peptides in step 3 from the protein entered in step 2, the latest model (which has been re-trained on all the protein sequences entered before the current one) is always used.

follows.

1. **Approach-1**: Re-training the complete model using fresh data points without employing memory rehearsal on subsets of training and discovered sets.
2. **Approach-2**: Re-training the complete model using fresh data points with memory rehearsal on subsets of training and discovered sets.
3. **Approach-3**: Re-training some layers of the model using fresh data points without employing memory rehearsal on subsets of training and discovered sets.
4. **Approach-4**: Re-training some layers of the model using fresh data points with memory rehearsal on subsets of training and discovered sets.

Algorithm 1 Forming re-training dataset

Input: $ABPP, nABPP, ABPT, nABPT, ABPD, nABPD$

Output: R ▷ the dataset used for re-training

procedure RE-TRAIN($ABPP, nABPP, ABPT, nABPT, ABPD, nABPD$)
  $plen \leftarrow size(ABPP)$ ▷ number of predicted ABPs
  $nlen \leftarrow size(nABPP)$ ▷ number of predicted non-ABPs
  $R \leftarrow$ SELECT-RANDOM($ABPT, 1500$) $\cup$ SELECT-RANDOM($nABPT, 1500$)
  if $plen > 135$ **then**
    $R = R \cup$ SELECT($ABPP, 135$)
    $R = R \cup$ SELECT-RANDOM($ABPD, 1500$)
  else
    $R = R \cup ABPP$
    $R = R \cup$ SELECT-RANDOM($ABPD, 1500 + (135 - plen)$)
  end if
  if $nlen > 135$ **then**
    $R = R \cup$ SELECT($nABPP, 135$)
    $R = R \cup$ SELECT-RANDOM($nABPD, 1500$)
  else
    $R = R \cup nABPP$
    $R = R \cup$ SELECT-RANDOM($nABPD, 1500 + (135 - nlen)$)
  end if
return R
end procedure
procedure SELECT($pep, length$)
  $chosen \leftarrow \phi$
  $x = \lceil length/27 \rceil$
  while $4 \leq i \leq 30$ **do**

25: pep_i = select x peptides of length i from pep

26: $chosen = chosen \cup pep_i$

27: **end while**

28: **return** *chosen*

29: **end procedure**

The strategy employed to get the fresh data points for re-training the model is as follows. The model utilizes protein sequences to predict the presence of ABPs and non-ABPs. It then identifies ABPs with *score* equal to or more than 0.99 and non-ABPs with *score* equal to or less than 0.01. The process of picking ABPs and non-ABPs has been comprehensively elucidated in Algorithm 1. The approach utilizes the notation $ABPT, ABPD$, and $ABPP$ to represent the ABPs of the training, discovered, and predicted sets, respectively. In a similar manner, the non-ABPs belonging to the training, discovered, and predicted sets are represented by the variables $nABPT, nABPD$, and $nABPP$, respectively. It should be noted that the data points included in $ABPP$ and $nABPP$ are those that are not previously included in our training (T), validation (V), test (Te), or discovered (D) sets (as described in Eq. 3.4). In step 4, the algorithm selects 1500 peptides randomly from both $ABPT$ and $nABPT$. Subsequently, steps 5-18 were implemented in order to generate the re-training set, which will be utilized for the current round of re-training. In the case when the number of ABPs in the sequence $ABPP$ exceeds 135, a selection process is implemented to choose a maximum of 5 ABPs of all the length values within the range of 4 to 30. This ensures that a total of 135 ABPs are selected, with 5 ABPs representing each length value within the specified interval. Occasionally, a scenario may occur wherein the number of anticipated ABPs for a length value is insufficient. In this scenario, the algorithm chooses the ABPs with a high score in order to get a total of 135 selected ABPs. After selecting ABPs from the $ABPP$ dataset, the algorithm chooses 1500 ABPs from the $ABPD$ dataset. Alternatively, in the event where the number of ABPs in $ABPP$ is fewer than 135, it

will proceed by choosing all ABPs contained inside $ABPP$ and selecting an additional 1500 or more ABPs from $ABPD$, as elaborated in line 10. The selection process for non-ABPs follows a similar technique, as described in lines 12-18. It should be noted that the lines labeled as 4, 7, 10, 12, and 18 are only run during the implementation of approaches 2 and 4 for the purpose of re-training the model.

$$\begin{aligned} T &= ABPT \cup nABPT \\ D &= ABPD \cup nABPD \\ P &= ABPP \cup nABPP \\ P \cap D &= \phi \\ P \cap T &= \phi \\ P \cap V &= \phi \\ P \cap Te &= \phi \end{aligned} \tag{3.4}$$

After the model is re-trained on R, the new data points (those selected from $ABPP$ and $nABPP$) are added to the discovered set (D). When the size of the discovered set increases beyond a certain limit, some sequences are randomly removed every time the model undergoes re-training so that D does not become unmanageable.

3.4 Experiments, Results, and Discussions

Extensive experimentations have been conducted on the proposed model, MSTCN-ABPpred, which was coded using the Python programming language. The trials were conducted on a graphics processing unit (GPU) node with an NVIDIA V100 GPU core. The compute node also included 16 GB of random access memory. In order to execute the deep learning model, the Keras framework with Tensorflow was employed as the underlying computational engine [40].

3.4.1 Evaluation Criteria

Several evaluation metrics, like accuracy, f1-score, specificity and area under the receiver operating characteristic curve (AUC), were used to evaluate the proposed model. These metrics are as described in Eqs. 3.5-3.11.

$$Accuracy\ (Acc) = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

$$Precision\ (Pr) = \frac{TP}{TP + FP} \tag{3.6}$$

$$Recall\ (Rec)\ (\text{or}\ True\ Positive\ Rate\ (TPR)) = \frac{TN}{TN + FP} \tag{3.7}$$

$$F1\text{-}score\ (Fs) = \frac{2 \times Pr \times Rec}{Pr + Rec} \tag{3.8}$$

$$Specificity\ (Sp) = \frac{TN}{TN + FP} \tag{3.9}$$

$$False\ Positive\ Rate\ (FPR) = 1 - \frac{TN}{FP + TN} \tag{3.10}$$

$$AUC = \int TPR.\ \mathrm{d}(FPR) \tag{3.11}$$

3.4.2 Performance Evaluation

Initially, a comparative analysis of the proposed model's performance against several contemporary models, namely AMAP [46], StaBle-ABPpred [44], Deep-ABPpred [3], iAMPpred [4], AMPFUN [48], ABPD, RNN-ABPD, and HABPD [45], has been done using the test set. Subsequently, an examination is conducted on the efficacy of the four CL-based models derived after the re-training of MSTCN-ABPpred (BL) using four distinct techniques for continual learning, as elucidated in section 3.3.2. Subsequently, to maintain a good balance between stability and plasticity, the most appropriate technique for re-training is chosen for incorporation into the MSTCN-ABPpred (BL) model, after which the adaptive model has been referred to as MSTCN-ABPpred (CL). To es-

Table 3.1: Performance of various models on the test set

Model	Accuracy(%)	F1-score(%)	Specificity(%)	AUC(%)
MSTCN-ABPpred (BL)	97.91	97.65	98.05	99.72
Deep-ABPpred	96.40	95.86	96.00	99.10
RNN-ABPD	88.64	88.40	81.51	89.53
HABPD	87.06	87.02	78.53	88.13
ABPD	85.54	85.77	75.53	86.81
iAMPpred	80.95	80.98	72.62	82.00
AMAP	78.28	80.18	61.78	80.12
AMPFUN	70.79	74.97	48.64	73.58

tablish the stability of the MSTCN-ABPpred (CL) model, the analysis of variance (ANOVA) test is employed, which confirms that its performance does not exhibit a statistically significant decline even after undergoing multiple rounds of re-training [41, 42].

3.4.2.1 Comparative analysis

The performance of MSTCN-ABPpred (BL) was compared with other contemporary models on the test set, utilizing metrics such as accuracy, f1-score, specificity, and AUC. The findings are presented in Table 3.1. The performance was also assessed using box and whisker plots. The dataset was divided ten different times to form ten pairs of training and test sets. The model has been individually trained and tested on all the pairs to obtain its performance on each test set. Subsequently, the performance of state-of-the-art models on the test sets was also acquired. The obtained outcomes were used to construct box and whisker plots, which might be understood through a five-number summary, as listed below.

- Median (Q_2): The median performance.
- First Quartile (Q_1): It is the median of performance values that are lower than the Q_2.
- Third Quartile (Q_3): It is the median of performance values that are higher than the Q_2.
- Minimum value (Min): It is the lowest value of performance calculated as per the interquartile range (IQR), as given in Eq. 3.12.

- Maximum value (Max): It is the highest value of performance calculated as per IQR.

$$\begin{aligned} IQR &= Q_3 - Q_1 \\ Min &= Q_1 - 1.5 \times IQR \\ Max &= Q_3 + 1.5 \times IQR \end{aligned} \tag{3.12}$$

The following can be deduced using Figure 3.3.

1. Small whiskers for the proposed model indicate that its minimum, maximum, and median performances are closer to each other.
2. The absence of fliers suggests that the proposed model does not exhibit erratic performance.
3. The proposed model's IQR is lesser than that of others, implying more consistency in performance than others.

The performance gap between MSTCN-ABPpred and StaBLe-ABPpred is less pronounced. Nevertheless, the major advantage of the former over the latter is its reliance on TCNs, which facilitate parallel computation. The lack of parallelism in implementing the bi-LSTM algorithm significantly increases the time required for re-training a model. Hence, the application built upon StaBle-ABPpred will have prolonged unavailability to its end-users due to the re-training process. Meanwhile, the MSTCN-ABPpred would have a much faster re-training while having minimal impact on the real-time availability and accessibility of its application.

3.4.2.2 Analysis of the continual learning module

Following the development of the MSTCN-ABPpred (BL) model on the training set (T), its adaptability to new data points has been checked after implementing the four continual learning methodologies previously discussed in section 3.3.2. Using the proposed model, ABPs and non-ABPs were predicted in 63 tail proteins, each belonging

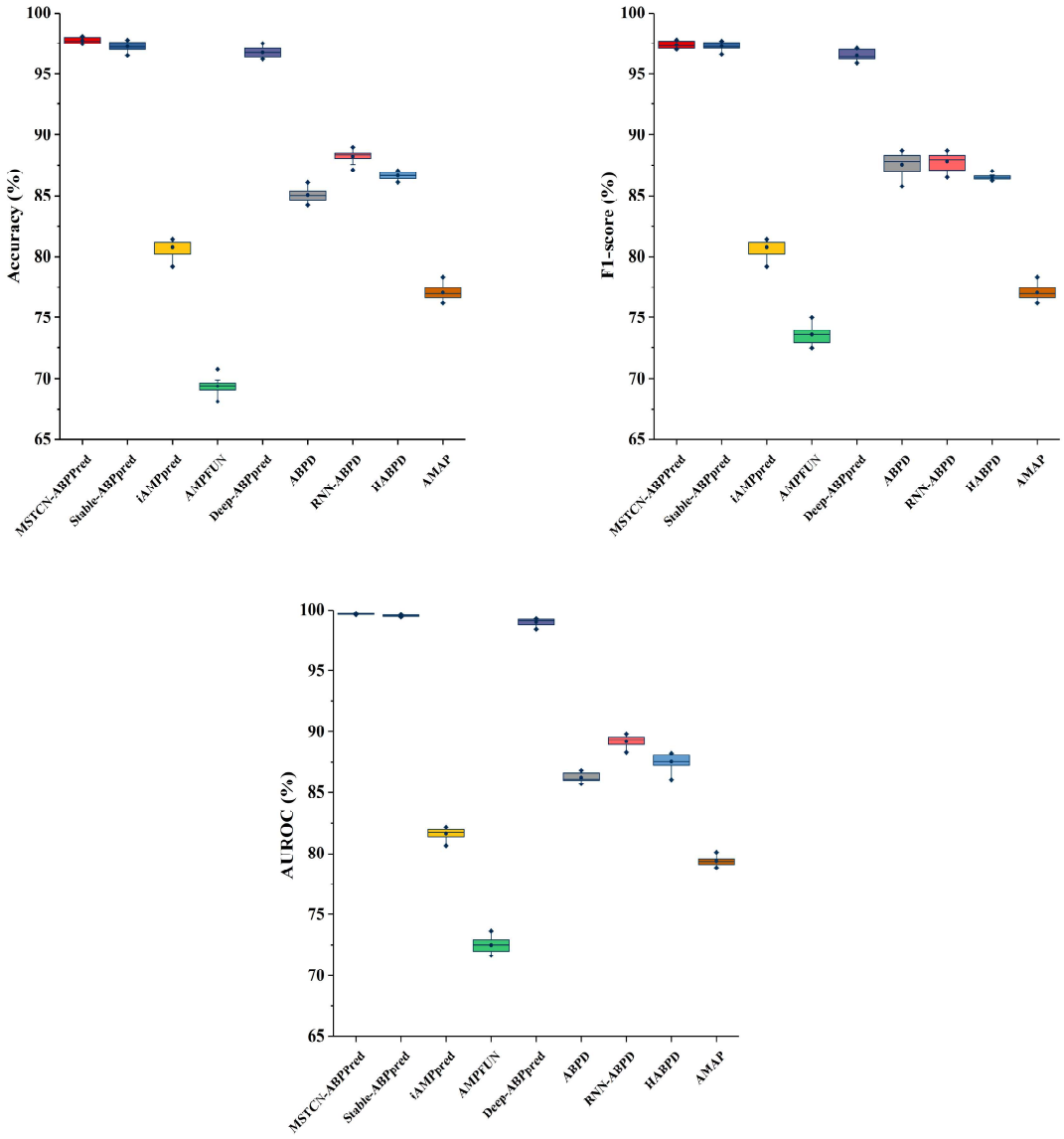

Figure 3.3: Box and whisker plots depicting performance of different models on test set based on accuracy(%), f1-score(%), and AUC(%)

to bacteriophages targeting the ESKAPEE pathogens. It is important to highlight that the most recent model, i.e., the model that underwent re-training on i proteins, was used to identify peptides in the $(i+1)^{th}$ protein. The ABPs and non-ABPs obtained by employing Algorithm 1 were utilized to retrain our model using the four continual learning approaches. After incorporating these approaches, the model's performance was analyzed and interpreted based on plasticity and stability.

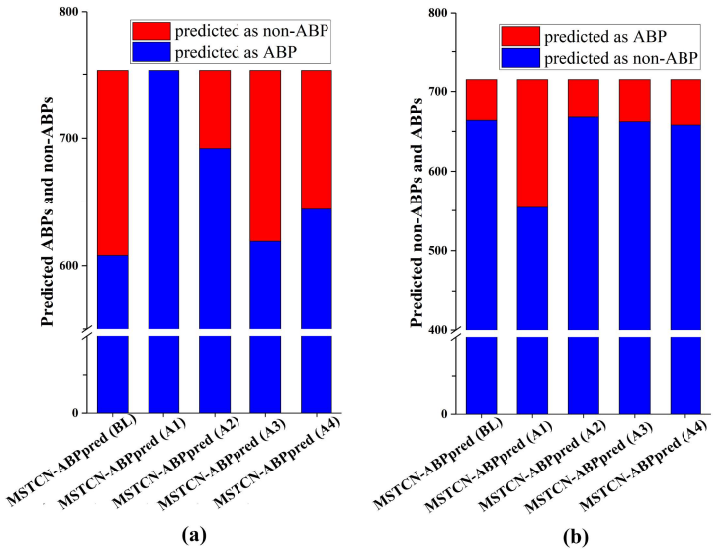

Figure 3.4: Performance of MSTCN-ABPpred (BL), MSTCN-ABPpred (A1), MSTCN-ABPpred (A2), MSTCN-ABPpred (A3), and MSTCN-ABPpred (A4) on an independent set of (a) ABPs active against ESKAPEE pathogens, and (b) general non-ABPs

1. **Plasticity**: The primary objective underlying the integration of continual learning is to facilitate the adaptation of a model to novel data. However, it is necessary to measure the performance of the re-trained model to determine whether the model is improving in terms of performance or not. In order to achieve this objective, a collection of experimentally verified ABPs that target the ESKAPEE pathogens has been used. The aim is to investigate the variations in performance when the model is retrained using four different approaches, namely approaches 1, 2, 3, and 4. Consequently, four distinct models were obtained, which are referred to as MSTCN-ABPpred (A1), MSTCN-ABPpred (A2), MSTCN-ABPpred (A3), and MSTCN-ABPpred (A4), respectively. The classification performance of MSTCN-ABPpred (A1) is demonstrated in Figure 3.4(a), where it is observed that all the ABPs were accurately classified. The performance of MSTCN-ABPpred (A2) was likewise notable when compared to the MSTCN-ABPpred

Figure 3.5: Performances (on the test set (Te)) of MSTCN-ABPpred (BL), MSTCN-ABPpred (A1), MSTCN-ABPpred (A2), MSTCN-ABPpred (A3) and MSTCN-ABPpred (A4) that were re-trained using approaches 1,2,3 and 4, respectively on (a) 7 sequences, (b) 14 sequences, (c) 21 sequences, (d) 28 sequences, (e) 35 sequences, (f) 42 sequences, (g) 49 sequences, (h) 56 sequences, (i) 63 sequences

(BL) model. Nevertheless, there is no significant difference in the performances of MSTCN-ABPpred (A3) and MSTCN-ABPpred (A4) when compared to the MSTCN-ABPpred (BL) model.

2. **Stability**: Maintaining previous knowledge while assimilating new information is equally important for a model undergoing re-training. An optimal balance between stability and plasticity is crucial for ensuring the long-term sustainability of a continual learning-based framework. Figure 3.5 demonstrates that during the re-training process on 63 proteins, the model that uses approach-1 experienced a significant decrease in performance on the test set (Te). Nevertheless, the decline in performance for MSTCN-ABPpred (A2), MSTCN-ABPpred (A3), and MSTCN-ABPpred (A4) is low. Furthermore, when evaluating these models on a separate dataset consisting of non-ABPs (as depicted in Figure 3.4(b)), it was noted that the MSTCN-ABPpred (A2) achieved the highest accuracy in classifying the majority of non-ABPs.

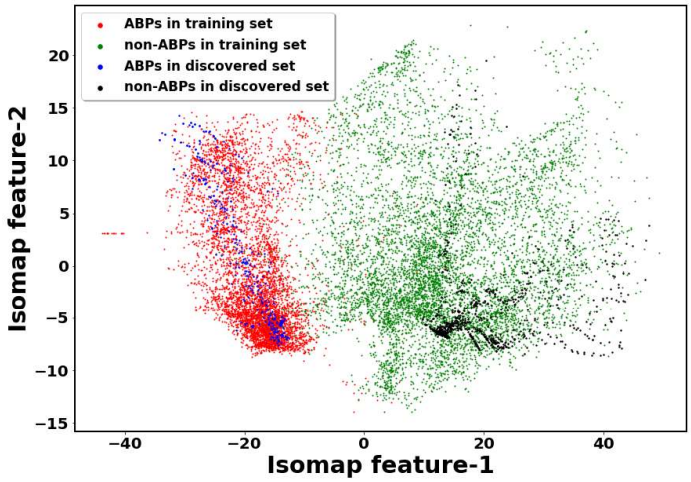

Figure 3.6: Visualization of peptides in the discovered set (D), and the initial training set (T) using the isomap technique.

To achieve a favorable balance between stability and plasticity, the MSTCN-ABPpred (A2) has been selected as the proposed continual learning-based model, which involves training the entire model on novel data while performing a memory rehearsal

Table 3.2: Performance of the model on the test set after re-training on proteins

Model	Accuracy(%)	F1-score(%)	Specificity(%)	AUC(%)
MSTCN-ABPpred (BL)	97.91	97.65	98.05	99.71
After re-training on 7 proteins	97.91	97.49	97.50	99.74
After re-training on 14 proteins	97.76	97.44	97.57	99.74
After re-training on 21 proteins	97.72	97.44	97.64	99.74
After re-training on 28 proteins	97.72	97.44	97.64	99.74
After re-training on 35 proteins	97.72	97.44	97.63	99.74
After re-training on 42 proteins	97.72	97.34	97.63	99.74
After re-training on 49 proteins	97.64	97.34	97.50	99.74
After re-training on 56 proteins	97.64	97.34	97.50	99.74
After re-training on 63 proteins	97.64	97.34	97.50	99.74

Table 3.3: ANOVA on accuracy (%) of the model after several rounds of retraining

(a) Input summary

Group	Count	Sum	Average
Before re-training	10	979.10	97.91
After re-training on 7 proteins	10	978.35	97.84
After re-training on 14 proteins	10	977.44	97.74
After re-training on 21 proteins	10	977.16	97.72
After re-training on 28 proteins	10	976.96	97.69
After re-training on 35 proteins	10	977.04	97.70
After re-training on 42 proteins	10	976.88	97.69
After re-training on 49 proteins	10	976.64	97.66
After re-training on 56 proteins	10	976.40	97.64
After re-training on 63 proteins	10	976.40	97.64

(b) ANOVA result

Source of Variation	SS	df	MS	F-stat	P-value	F-crit
Between Groups	0.67	9	0.071	1.98	0.07	68.94
Within Groups	0.09	90	0.001	-	-	-
Total	0.77	99	-	-	-	-

on a subset of training and discovered sets. This model is referred to as MSTCN-ABPpred (CL). The model was implemented as a web application and made available at `https://mstcn-abppred.anvil.app/`. The distribution of peptides in D, acquired after re-training on the peptides predicted in 63 bacteriophages, with the ABPs in T, is shown using the isometric mapping (isomap) approach, as depicted in Figure 3.6. The overlap is clearly substantial. Similar conclusions may be drawn regarding non-ABPs identified by this framework and those found in the original dataset T.

The performance outcomes of the proposed model at various stages, namely after

re-training on 7, 14, 21, $\cdots$, 63 sequences, on the test set (Te) are presented in Table 3.2. The MSTCN-ABPpred (CL) model exhibits consistent performance throughout multiple rounds of re-training. This re-training process was repeated ten times on a set of 63 proteins. Each repetition involved re-training the MSTCN-ABPpred (BL) model (from the beginning) after inputting the sequences into the web application in random order. The accuracy values obtained from the model at various stages of re-training were subjected to the ANOVA test. The null and alternative hypotheses are expressed as Eqs. 3.13 and 3.14, respectively.

$$H_0 : \mu_1 = \mu_2 \cdots = \mu_{10} \tag{3.13}$$

$$H_1 : \mu_1 \neq \mu_2 \cdots \neq \mu_{10} \tag{3.14}$$

The variables $\mu_1, \mu_2, \cdots, \mu_{10}$ denote the average performance of the model following re-training on 7, 14, 21, $\cdots$, and 63 sequences, respectively (as seen in Table 3.3). If the null hypothesis H_0 holds, it can be inferred that the model exhibits stability. To proceed, it is necessary to compute the value of the F-statistic as outlined in the equations provided by Eqs. 3.15-3.21.

$$df(Between) = s - 1 \tag{3.15}$$

$$df(Within) = n - s \tag{3.16}$$

$$SS(Between) = \sum_{i=1}^{10} (n_i \dot{(}\bar{x} - \bar{\bar{x}})) \tag{3.17}$$

$$MS(Between) = \frac{SS(Between)}{df(Between)} \tag{3.18}$$

$$SS(Within) = \sum_{b=1}^{s} \sum_{c=1}^{n_i} (x_c - \bar{x_b})^2 \tag{3.19}$$

$$MS(Within) = \frac{SS(Within)}{df(Within)} \tag{3.20}$$

$$Fstat = \frac{MS(Between)}{MS(Within)} \tag{3.21}$$

In this context, the symbol $\bar{x}$ represents the average accuracy of the model at a specific stage, such as after being re-trained on seven proteins. On the other hand, the symbol $\bar{\bar{x}}$ represents the overall average, calculated by taking the average of the mean accuracy values. The variable s represents the number of instances at which results were considered for comparison (here, 10). The variable n_i represents the number of results taken at each instance (here, 10). Lastly, the variable n represents the overall number of results taken (here, 200). The symbol df represents the degrees of freedom, SS represents the sum of squares, MS represents the mean square, and $Fstat$ represents the value of the F-statistic. The p-value quantifies the probability of observing a value of the F-statistic due to random chance. Therefore, to reject the null hypothesis, the p-value should be less than the chosen alpha or significance level (α=0.05), and the F-statistic should exceed the F-critical value. In this case, it was found that both of these prerequisites were confirmed to be false. Therefore, based on our analysis, we do not have sufficient evidence to reject the null hypothesis. This leads to the conclusion that there is no statistically significant decrease in the performance of the proposed model, even after undergoing multiple re-training iterations. In other words, this model does not exhibit catastrophic forgetting.

3.5 Conclusion

This research study introduces a model named MSTCN-ABPpred (CL) that is based on continual learning for predicting and identifying ABPs. During the testing phase, the classifier achieved an accuracy of 98%, which surpasses several contemporary classifiers. The model can adjust and undergo re-training based on the ABPs/non-ABPs found

in proteins that are input via a publicly accessible web application built using this model at `https://mstcn-abppred.anvil.app/`. The continuous learning module of the proposed model was demonstrated by conducting re-training on peptides identified in the tail proteins of ESKAPEE targeting bacteriophages. This analysis revealed that the model exhibited stability even after undergoing multiple re-training iterations with new data points, as evidenced by the absence of any statistically significant decline in its performance. In order to ascertain the plastic nature of the model, an evaluation of the retrained model was conducted using a separate test dataset consisting of ABPs that have demonstrated promise in combating ESKAPEE infections. It was found that the re-trained model exhibited superior performance compared to the baseline model, namely MSTCN-ABPpred (BL).

One shortcoming of the proposed study is that it does not focus on providing information regarding the target of the discovered ABP, as well as its potential haemotoxicity, cytotoxicity, and minimum inhibitory concentration (MIC) against several bacterial strains. Therefore, it is possible to expand upon this study by developing a multi-label classification model that incorporates information regarding the target bacterium as well as these factors. Several potential avenues for future research on this model can be explored. Also, it is worth considering various notable architectures like BERT to develop a better prediction model. Furthermore, techniques such as generative replays may be considered for approximating old data points for the purpose of memory rehearsals.