cAPTured: Neural Reflex Arc-Inspired Fuzzy Continual Learning for Capturing *in Silico* Aptamer-Target Protein Interactions

Aviral Chharia
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA
achharia@andrew.cmu.edu

Runjhun Saran
Waterloo Institute of Nanotechnology
University of Waterloo
Waterloo, ON, Canada
rsaran@uwaterloo.ca

Apurva Narayan
Department of Computer Science
Western University
London, ON, Canada
apurva.narayan@uwo.ca

***Abstract*—Aptamers are oligonucleotides or peptides with unique binding properties for specific target molecules, and they have shown great potential in diagnostics, therapeutics, and biosensing. However, the current *in vitro* SELEX-based method for discovering new target-selective aptamers is challenging, time-consuming, and often unsuccessful in finding high-affinity aptamers. Recently, *in silico* methods have gained immense attention. However, since labeled interaction-pair data collection is expensive and needs highly trained specialists, available data is sparse. Further, since acquiring positive-class samples is even more challenging, available datasets showcase high-class imbalance. This makes designing deep learning models incredibly challenging, as they require a sufficiently large training set and are biased towards the dominant class. Additionally, current models cannot be updated in real-time, and end-to-end re-training is necessary for each new aptamer-target interaction pair discovery. The present work is the *first* to address both these challenges. We present *cAPTured*, a *novel* fuzzy continual learning method for predicting aptamer-target protein interaction pairs in a continual learning environment. *cAPTured* continually updates its learned feature space on a non-stationary interaction-pair data stream. We performed extensive evaluation studies and experiments to establish the effectiveness of the proposed approach. *cAPTured* outperforms existing methods on the benchmark dataset by a significant margin.**

***Index Terms*—Continual Learning, Machine Learning, Fuzzy Classification, Aptamer-Protein interactions**

I. Introduction

A. Background

Aptamers are short single-stranded sequences of RNA [1], DNA [2], and other oligonucleotides or peptides. First discovered in the 1990s, aptamers have demonstrated significant promise in diagnostics, therapies, and bio-sensing. Aptamers have a high potential to serve as superior alternatives to antibodies due to their molecular recognition capability. Compared to traditional antibodies, aptamers offer numerous advantages: (1) Aptamers exhibit higher stability at high temperatures. Upon renaturation, aptamers regain their natural structure and attach to targets. (2) Aptamers can be screened *in vitro* using an artificial library instead of cell lines/ animals required for antibody screening. (3) After screening, aptamers can be manufactured on a large scale with high purity by elementary polymerase-chain reactions. (4) Aptamers' spatial conformations allow numerous functional groups addition (compared to antibodies), giving them ability to bind to specific biomolecular targets [3], including lipids [4], viruses [5], nucleic acids [6], cytokines [7], ions [8], with high specificity and affinity.

B. Challenges

In industry, aptamers are identified, screened, and selected *in vitro* using the iterative process of Systematic Evolution of Ligands by EXponential enrichment (SELEX) [9], which consists of several repeated rounds of binding, partition, and amplification. This experimental "lab-based" method is challenging, costly, labor-intensive, time-taking (several weeks), and often fail to find high-affinity aptamers [10, 11]. Therefore, *in-silico* methods have come into the limelight in recent years to develop a simple and cost-efficient computational alternative for designing a more effective aptamer. However, developing computational methods still faces many challenges: (1) high class-imbalance (2) limited data availability (3) models becoming irrelevant over very short times as new interactions pairs are discovered. Due to this, very few models have been developed in the past decade (five in number).

C. Motivation

Continual Learning is a potential strategy for addressing these issues. Even though we humans are continually exposed to new and varied types of vision data, we do not forget the characteristics of previous objects we have seen. However, training neural networks in a continual learning setting is a very challenging task. Despite attempts to imitate the human brain, it has been observed that models in continual learning environments are prone to catastrophic forgetting/ interference, i.e., inability to retain what they previously learned when new class samples are presented.

D. Contributions

This work is the *first* to present a continual learning model for *in silico* prediction of aptamer-target protein interaction

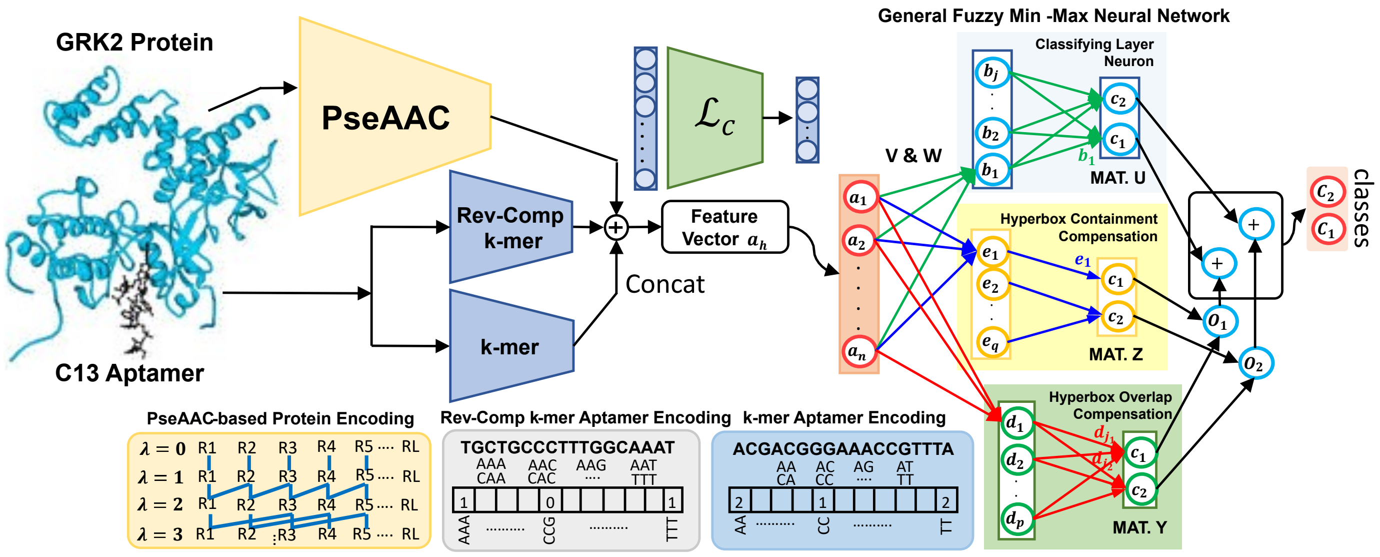

Fig. 1. Model architecture of *cAPTured* illustrating GRK2 protein (shown in blue) interacting with C13 aptamer (shown in black).

pairs. We encode aptamers using two different strategies: k-mer and reverse complement k-mer (revck-mer) frequency. Amino Acid Composition (AAC) and Pseudo-Amino Acid Composition (PseAAC) [12] were applied to represent target information using 24 physicochemical and conformational properties of proteins. To handle the skewness in data, we applied the neighborhood cleaning algorithm (NCL) [13]. Next, neural reflex arc-inspired fuzzy classification is performed. Further, we also illustrate the model's ability to train on limited data in a continual learning environment. The proposed method outperforms the current state-of-the-art (SOTA) models trained on the same benchmark dataset. The results indicate that *cAPTured* helps to identify novel aptamer–protein interaction pairs and build more-efficient biological insights into understanding the relationship between aptamers and target proteins. The contributions of this paper include:

- The present work is the *first* to eliminate model re-training for each new aptamer-target protein interaction pair discovery by formulating the problem as a continual learning task. Real-time architecture update enables the model to learn continually. (Section III)
- We propose a *de novo* neural reflex-arc inspired fuzzy continual learning method which significantly outperforms current SOTA models. A parametric analysis is also conducted to quantify model hyperparameters which is important for hyperparameter tuning since there is no such strategy for high-dimensional feature mappings. (Section IV-D, IV-E)
- The challenge of limited labeled data availability is addressed. *cAPTured*'s ability to attain high classification accuracy despite being trained on limited data samples, is demonstrated experimentally. Moreover, unlike conventional models, cAPTured only requires single-pass training. (Section IV-C). Another unique feature of the model lies in its robustness to high-class imbalance, demonstrated through experimental studies. (Section IV-C)

II. Related Works

A. Conventional Machine Learning (ML) methods

In past years, traditional ML approaches have been used to develop aptamer-screening methods, due to their strong learning capacity and versatility. Aptamer-target protein interaction pair prediction was firstly framed by [14] as an *in silico* task. Their method was based on Random Forest (RF) and used nucleotide acid composition (NAC) for encoding aptamers, while the target proteins were encoded using AAC and PseAAC [12]. Although somewhat successful, the model faced class-imbalance problem. This resulted in high prediction accuracy (87.13%) for large/ dominant non-interaction class and low accuracy (48.28%) for the small sample class, thus biasing the model towards the dominant class. This was due to high skewness and limited data samples, since acquiring positive-class samples is relatively more challenging. Moreover, with every new aptamer-target protein binding pair discovery, model re-training was required.

B. Low Performance by Ensemble methods

To deal with this problem, first studies used 3 RFs in the ensemble to minimize the effect of class imbalance [15]. They used Pseudo k-tuple nucleotide composition (PseKNC) to encode aptamers and discrete cosine transform (DCT), bi-gram position-specific scoring matrix (PSSM), and disorder information (DI) for encoding the target protein. However, the negative samples of each RF are less in their model due to train data split, which decreased the accuracy (from 77.4% to 71.9%) rather than increasing it and thus defeating the purpose [15]. [16] proposed using 03 support vector machines (SVMs) in an ensemble to address the class imbalance. They used NAC and PseKNC for aptamer encoding, and a sparse autoencoder (SAE) to represent the targets. [17] developed

TABLE I
NEW APTAMER-TARGET PROTEIN INTERACTION PAIRS OVER THE YEARS USED IN VARIOUS STUDIES

Year, Methods	New interaction pairs over the years	
	Dataset	Samples
2014 [14, 15, 16, 17]	Aptamer Base	725 positive, 2175 negative
2021 [18]	Aptamer Base, Aptagen [19]	850 positive, 2554 negative

a web server to predict aptamer-target protein interactions using an integrated framework AdaBoost and RF. Aptamers were represented by NAC, PseKNC, and normalized Moreau-Broto auto-correlation coefficient (NMBAC). However, proteins were characterized by AAC, PseAAC, grouped amino-acid composition, C/T/D composition, and sequence order-coupling number (SOC). Although these methods have generated good results, they had low accuracy and MCC scores that have restricted their wide use.

C. Failure of Deep Learning Models

Recently proposed, *AptaNet* [18] used a multi-layered perceptron (MLP) model, and applied NCL to address the class imbalance. AptaNet depicted an oscillation in loss and accuracy, confirming that the available data is less for developing a deep learning (DL) model. The availability of a limited number of training samples affects the development of DL models, since they are data-intensive and require a large number of training samples to tune their parameters. Thus, DL model design for aptamer-target protein interaction prediction is incredibly challenging and tends to overfit. *Second*, current models face the challenge of 'continuity'. All proposed approaches presume a 'closed-world assumption,' due to which models become outdated frequently and need to be re-trained from scratch for every new aptamer discovery. Table I confirms the increase in new interaction pair data over the recent years.

III. Proposed Methodology

A. Problem Formulation

Aptamer-target protein binding can be described in two distinct groups: 'bound' and 'unbound' sites. Therefore, the problem can be framed as follows:

$$f(x,y) = \begin{cases} 1 & \text{if aptamer } a \text{ binds to target } t \\ -1 & \text{otherwise} \end{cases}$$

Here, x represents target t, and y for an aptamer a.

B. Aptamer and target Protein Sequence Encoding

In this study, we have used k-mer and revck-mer to encode aptamer sequences. For encoding target protein sequences, we have utilized AAC (Amino Acid Composition) and PseAAC (Pseudo Amino Acid Composition) [12, 20]. Complete explanation of both encoding techniques have been given in great detail in Appendix A and B.

C. Feature Selection

In this study, 24 physicochemical and biochemical properties of amino acids are used. This includes hydrophobicity, hydrophilicity, mass, polarity, molecular weight, melting point, transfer-free energy, buri-ability, bulkiness, solvation free energy, relative mutability, residue volume, volume, amino acid distribution, hydration number, iso-electric point, compressibility, chromatographic index, unfolding entropy change, unfolding enthalpy, unfolding Gibbs free energy change power to beat the N terminal, C terminal and middle of alpha helix. These 24 properties were retrieved from [21, 22]. To generate protein characteristics, the *iFeature*[1] package was used [23]. Next, RF-based feature selection was performed to select the most important features statistically. RF ranks all features based on the improvement of node purity to select the important features.

D. Undersampling for targetting class imbalance

Acquiring positive-class labeled aptamer-target protein interaction pairs is highly challenging and costly. Thus, the available datasets have limited training samples and a high bias towards the negative class. This makes conventional DL model design incredibly challenging as they require a large training set and would over-fit or be biased towards the dominant class.

The proposed model design is capable of learning on limited data. The architectural reason for the same has been explained in detail in the following section. Further, since the proposed model can learn on restricted data, we use under-sampling [24] to balance the dataset by reducing the majority class size with replacement. Here, Wilson's edited nearest neighbor (ENN) rules [25] are used to identify noisy data and reduce the majority class by removing them. It is to be noted that only those instances that differ from at least 2 of their 3 nearest neighbors are removed. Furthermore, neighborhood cleaning is used to intensify the size reduction of the dominant class.

E. Neural Network-based Feature Vector Reduction

The features from the formed 640-dimensional feature vector are reduced to 32-dimensional feature vector representation using a feature reduction network. This is primarily done to reduce the dimensionality since the number of features formed surpasses the number of training samples. This reduces the computational time as the dominant features are extracted and redundant features are discarded. For this, a neural network was trained and used as a feature extractor. The network architecture is depicted in Figure 1. The network consists of 5 layers fully-connected dense layers that reduce the feature vector size from 208 to 32. ReLU non-linearity ($\texttt{ReLU}(x) = x$ if $x \geq 0$, else $= 0$ if $x < 0$) was used as the activation function. To prevent overfitting on the training dataset, a dropout of 0.3 was given after each layer. In the end, a sigmoid activation ($\texttt{Sigmoid}(x) = \frac{1}{1+e^{-x}}$) was used with the output layer. 'Binary cross-entropy' loss function

[1]iFeature python implementation is available at https://github.com/Superzchen/iFeature/

Algorithm 1: Training algorithm

```
Input: hyper-parameters $\theta, \gamma$, Trained $FEN$, New Experimental Data $(a_h, C_i)$
Output: Updated model M
 1: procedure: TRAIN($\theta$, $\gamma$, $FEN$, ($a_h$, $C_i$))
 2:     Load $FEN$, M ▷ Pre-trained $FEN$ and Current model
 3:     for $i$ in num_samples do
 4:         $a_h \leftarrow FEN(a_h)$
 5:         if $b_j$ accommodates $a_h$ ▷ H Isolation, Contraction Test
 6:             expand $b_j$
 7:             if $b_j$ is isolated ▷ H Isolation Condition
 8:                 continue
 9:             else
10:                 if $b_j$ is contained ▷ H Containment Condition
11:                     add $\mathcal{CCN}$
12:                 else
13:                     add $\mathcal{OCN}$
14:                     continue
15:         else
16:             Create H with $V$ = $W$ = $a_h$
17:     end for
18:     return M ▷ Updated model
19: end procedure
```

($\mathcal{L}_c = \frac{-1}{n}\sum_x\sum_t[ylna + (1-y)ln(1-a)]$) was used to train the model. The model was trained for a total of 260 epochs. Two model callbacks, model checkpoint and reduce learning rate on plateau, were used to help in the training process. The model checkpoint would save the best performing model after each epoch, while the reduce learning rate on plateau, which had a patience value of 10 and a reduction factor of 0.1, would reduce the learning rate by a factor of 0.1 if the accuracy did not increase for 3 consecutive epochs. Here, the final classification layer is discarded, and the remaining network is used as a feature extractor. The final dataset denoted by $\mathcal{S}$, consists of a labeled stream of samples, i.e., $\mathcal{S}^0$, $\mathcal{S}^1$, . . ., where $\mathcal{S}^{(t)} = (a_j^{(t)}, C_j^{(t)})_{j=1}^{N}$, where $t = 0, ..., K, (K+1), ..., L$.

F. Point Hyperbox Creation

The 32−dimensional feature vector representation for each sample is sent to the classifying neurons ($\mathcal{CLN}$), which use min-max hyperboxes to classify the learnt data. In this space, discontinuous decision boundaries in the form of 'hyperboxes ($\mathcal{H}$)' are created during model training [26]. Defining this decision boundary requires a **min** coordinate $\mathcal{V}_j = (v_{j1}, v_{j2}, ..., v_{jn})$, and a **max** coordinate $\mathcal{W}_j = (w_{j1}, w_{j2}, ..., w_{jn})$ [27]. However, in order to reduce the model hyperparameters, a single parameter, i.e., $\theta \in (0, 1)$ that represents the hyperbox size is defined. During training, for the first input, a point $\mathcal{H}$ is created. Subsequently, the model tries to accommodate remaining training samples $\{a_h, C_i\}$ in the previously constructed hyperboxes belonging to the same class C_i, provided the $\mathcal{H}_{size}$ does not exceeds a specified maximum limit ('θ'). A new $\mathcal{H}$ node is created in the $\mathcal{CLN}$ section, at times when the model encounters a training sample that does not belong to the classes it has seen so far, i.e., for a new training sample $\{a_h, C_i\}$, a hyperbox $\{b_j, C_j\}$ is found such that $C_j = C_i$ or $C_j = C_o$ which has the highest membership value and satisfying [27]-

$$(1)\ \theta_{\max} \geq \frac{1}{n}\sum_{i=1}^{n}(\max(w_{ji}, a_{hi}) - \min(v_{ji}, a_{hi})) \qquad (1)$$

$$(2)\ b_j \text{ is not associated with any compensation node} \qquad (2)$$

$$(3)\ \text{if } C_i = C_0 \text{ or } C_j = C_0 \text{ then } \mu_j > 0, \qquad (3)$$

where μ_j is membership of b_j. The coordinates of b_j are adjusted, as,

$$V_{ji}^{new} = \min(V_{ji}^{old}, a_{hi}), \qquad (4)$$

$$W_{ji}^{new} = \max(W_{ji}^{old}, a_{hi}), \text{ where } i = 1, 2, ..., n \qquad (5)$$

$$\text{and if } C_j = C_0 \text{ and } C_i \neq C_0 \text{ then } C_j = C_i \qquad (6)$$

Following, the min-max coordinates of b_j are adjusted such that, $\mathcal{V}_{ji}^{new} = \min(\mathcal{V}_{ji}^{old}, a_{hi})$; $\mathcal{W}_{ji}^{new} = \max(\mathcal{W}_{ji}^{old}, a_{hi})$, where $i = 1, 2, ..., n$, and *third*, if $C_j = C_0$ and $C_i \neq C_0$ then $C_j = C_i$. If no suitable b_j is present then a novel hyperbox $\mathcal{H}$ for class C_i is created with $\mathcal{V}_j = \mathcal{W}_j = a_h$; i.e., a point hyperbox is created.

In CLNs, the neuron b_j represents hyperbox fuzzy set i.e., $= A_h, \mathcal{V}_j, \mathcal{W}_j, f(A_h, \mathcal{V}_j, \mathcal{W}_j)\ \forall\ (A_h \in I_n)$ [27]. The input nodes and hyperbox nodes are coupled in the intermediate layer of the classifier. These connections represent the $n-$dimensional hyperbox fuzzy set's min-max coordinates, i.e., $\mathcal{V}$ and $\mathcal{W}$. The neurons in the intermediate layer are dynamically produced during training. Connection between the hyperbox node b_j to a class node C_j is represented by matrix $\mathcal{U}$, where, $u_{ij} = 1$ if $b_j \in C_j$ else $u_{ij} = 0$. In CLN nodes, to compute the class memberships, activation function by [28] is used to assign membership value $= 1$ when the test sample falls within the hyperbox. When the test sample is located outside $\mathcal{H}$, the model determines the membership-value based on its distance from $\mathcal{H}$'s extreme coordinates. More fuzziness in classification is achieved by increasing fuzziness control parameter (γ), whereas crisp classification is achieved by reducing it.

G. Neural Reflex Arc-Inspired Fuzzy Continual Learning

Since high-dimensional feature space contains all aptamer-target protein feature characteristics, hyperbox overlap is a possibility. Overlap Compensation Neurons ($\mathcal{OCN}$) and Containment Compensation Neurons ($\mathcal{CCN}$) are biologically-inspired error minimization sections which activate only when an instance of $\mathcal{H}$ overlap/ containment is found [29].

H. Decision Boundary Overlap

The Reflex mechanism is naturally inspired by the human reflex mechanism, which automatically takes control of the body in dangerous situations [29]. It contains the overlap compensation neurons ($\mathcal{OCN}$) and containment compensation neurons ($\mathcal{CCN}$), which assists in achieving more explainable class memberships with high accuracy. When a situation of hyperbox overlap and confinement is found, these neurons become active. A hyperbox ($\mathcal{H}$) of size equal to the overlapping space between two hyperboxes belonging to distinct classes is represented by $\mathcal{OCN}$. Only if the test data is within the overlap space does the $\mathcal{OCN}$ section become active. It produces two compensation outputs, one for each overlapping class. The $\mathcal{CCN}$ section, which solves the hyperbox containment problem, depicts a hyperbox ($\mathcal{H}$) with the same size as the overlapping space between the two classes. When a test sample enters the overlapping space, $\mathcal{CCN}$ is activated [29]. To see if there is any overlap, the Hyperbox overlap test is employed. δ^{old} starts with a value of 1.

C1: $v_{ji} < v_{ki} < w_{ji} < w_{ki}$ then, $\delta^{new} = \min(w_{ji} - v_{ki}, \delta^{old})$
C2: $v_{ki} < v_{ji} < w_{ki} < w_{ji}$ then, $\delta^{new} = \min(w_{ki} - v_{ji}, \delta^{old})$
C3: $v_{ji} < v_{ki} \le w_{ki} < w_{ji}$ then, $\delta^{new} = \min(\min(w_{ki} - v_{ji}, w_{ji} - v_{ki}), \delta^{old})$
C4: $v_{ki} < v_{ji} \le w_{ji} < w_{ki}$ then, $\delta^{new} = \min(\min(w_{ki} - v_{ji}, w_{ji} - v_{ki}), \delta^{old})$

If overlaps exist, i.e., one of the above conditions is true, and $(\delta_{new} - \delta_{old}) > 0$, then, $\Delta = i$ **else** $\Delta = -1$.

The contraction test is used to check for hyperbox contraction, i.e., if there is overlap and it is minimal along the Δ dimension, the hyperboxes are contracted using the provided conditions:

C1: $v_{j\Delta} < v_{k\Delta} < w_{j\Delta} < w_{k\Delta}$ then, $v_{k\Delta}^{new} = w_{j\Delta}^{new} = \frac{w_{j\Delta}^{old} + v_{k\Delta}^{old}}{2}$
C2: $v_{k\Delta} < v_{j\Delta} < w_{k\Delta} < w_{j\Delta}$ then, $v_{k\Delta}^{new} = w_{j\Delta}^{new} = \frac{w_{k\Delta}^{old} + v_{j\Delta}^{old}}{2}$
C3: $v_{k\Delta} < v_{j\Delta} \le w_{j\Delta} < w_{k\Delta}$ and $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $v_{j\Delta}^{new} = w_{k\Delta}^{old}$ else $w_{j\Delta}^{new} = v_{k\Delta}^{old}$
C4: $v_{j\Delta} < v_{k\Delta} \le w_{k\Delta} < w_{j\Delta}$ and $w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta}$ then $w_{j\Delta}^{new} = v_{k\Delta}^{old}$ else $v_{j\Delta}^{new} = w_{k\Delta}^{old}$

I. Decision Boundary Containment

The hyperbox node is dynamically formed in the reflex section's intermediate layer if a condition of overlap/ partial or complete containment of hyperbox ($\mathcal{H}$) is observed [29]. Overlap/ containment between a labelled hyperbox $B_j \in C_i \forall i > 0$ and an unlabeled hyperbox $B_k \in C_i \forall i = 0$ is permitted and does not result in the creation of any $\mathcal{OCN}$ or $\mathcal{CCN}$ nodes. The total number of classes learnt by the model is represented by the number of output layer nodes in the $\mathcal{CLN}$ section. The final membership value for the i^{th} class node is computed as $\mu_i \leftarrow C_i + O_i$, where C_i and O_i are the membership and the compensation values computed for the i^{th} class, respectively. The reflex mechanism uses the $\mathcal{OCN}$ and $\mathcal{CCN}$ neurons to create compensatory outputs, unlike fuzzy min-max neural networks (FMNNs), which constrict a hyperbox in the event of overlaps [27].

The hyperbox isolation condition is checked, i.e., if $(\mathcal{V}_{ki} < \mathcal{W}_{ki} < \mathcal{V}_{ji} < \mathcal{W}_{ji})$ or $(\mathcal{V}_{ji} < \mathcal{W}_{ji} < \mathcal{V}_{ki} < \mathcal{W}_{ki})$ is true for any $i \in 1, ..., n$, then, (b_k, b_j) are isolated [27]. The feature space must be verified for containment if the isolation condition does not hold. The hyperbox confinement condition is used to detect probable containment situations [29]. According to it, if $(\mathcal{V}_{ki} < \mathcal{V}_{ji} < \mathcal{W}_{ji} < \mathcal{W}_{ji})$ or $(\mathcal{V}_{ji} < \mathcal{V}_{ki} < \mathcal{W}_{ki} < \mathcal{W}_{ji})$ is true for any $i \in 1, ..., n$, then hyperboxes are contained and a $\mathcal{CCN}$ node is formed dynamically. A $\mathcal{OCN}$ node is formed if hyperboxes are not contained.

IV. Experiments and Results

A. Datasets

We evaluate our method *cAPTured* on the benchmark dataset by [18], which consists of all currently discovered aptamer-target protein interaction pairs from two popular databases: *Aptagen*[2] and *Aptamer Base* [19], containing 554 (477 RNA/ DNA aptamers and 241 target proteins) and 1638 (1381 RNA/ DNA aptamers and 211 target proteins) interaction entries respectively. Since both databases contain the target protein's identifier names, the sequence of each target protein was obtained from UniProtKB/ Swiss-Prot[3]. Aptagen has 269 interactions with the 241 protein targets out of the 477 aptamers present. As a result, the 269 pairs are regarded positive samples. Similarly, only 725 interactions with 164 proteins were found among 1381 aptamers for *Aptamer Base*; hence 725 pairs are deemed to be positive samples. Thus 850 pairs targeting 452 proteins were obtained. Negative pair instances were created using random pairs that had no overlap with the positive examples. Finally, the total number of instances was 3404, which contained 850 positive and 2554 negative instances.

B. Experimental Details

We briefly summarize the implementation details in this section.

Setup. The dataset was divided into the standard 80-20% train-test set. The normalisation method was '*zscore*', computed as $z = (x - u)/s$. All methods were implemented in the python programming language using Python 3.6. When training baselines for comparison, the same parameter settings and dataset was used to ensure a fair evaluation. The results obtained after 5-fold cross-validation are reported in this study.

Evaluation Metrics. We followed confusion matrix-based metrics to evaluate model performance. This includes $Acc = (\frac{TN+TP}{TN+FN+TP+FP}), Prec = (\frac{TP}{TP+FP}), Rec = (\frac{TP}{TP+FN}), F1 = 2(\frac{Prec \times Rec}{Prec + Rec})$. Since in case of class imbalance, accuracy may not be a good evaluation metric alone, we also estimated Matthew's Correlation Coefficient, i.e., $MCC = \frac{TP \times TP - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.

Platform. We carried out the studies on an NVIDIA K80 GPU workstation with 12GB RAM with Tensorflow [30] as a backend. The implementation is available with the supplementary material and will be open-sourced.

[2]*Aptagen* dataset is available at https://www.aptagen.com/
[3]*UniProtKB/ Swiss-Prot* datasets are available at: https://www.uniprot.org/uniprot/

TABLE II
RESULTS OF THE CONTINUAL LEARNING ABILITY OF THE PROPOSED MODEL ON APTAMER-TARGET PROTEIN BINDING PROBLEM. HERE, HYPERPARAMETERS ARE $\theta = 0.50$, $\gamma = 2$. HERE 1: IS THE BINDING PAIR CLASS AND 2: REPRESENTS NON-BINDING PAIR.

	$R1$ (@50)				$R2$ (@100)				$R3$ (@150)				$R4$ (@200)				$R5$ (@250)			
	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup
1	1.00	1.00	1.00	5	1.00	1.00	1.00	7	0.93	1.00	0.97	14	0.86	1.00	0.93	19	0.85	1.00	0.92	23
2	1.00	1.00	1.00	5	1.00	1.00	1.00	13	1.00	0.94	0.97	16	1.00	0.86	0.92	21	1.00	0.85	0.92	27
ACC			**1.00**	10			**1.00**	20			**0.97**	30			**0.93**	40			**0.92**	50
Macro AVG	1.00	1.00	1.00	10	1.00	1.00	1.00	20	0.97	0.97	0.97	30	0.93	0.93	0.92	40	0.93	0.93	0.92	50
Weighted AVG	1.00	1.00	1.00	10	1.00	1.00	1.00	20	0.97	0.97	0.97	30	0.94	0.93	0.92	40	0.93	0.92	0.92	50

TABLE III
APTAMER-TARGET PROTEIN INTERACTION PREDICTION RESULTS FOR MODEL TRAINED ON LIMITED DATA SUBSET CONFIGURATION.

	$S1$ (@100)				$S2$ (@200)				$S3$ (@300)			
	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup	Prec	Rec	F1	Sup
1	0.89	0.89	0.89	9	0.93	0.87	0.90	15	0.95	0.91	0.93	22
2	0.91	0.91	0.91	11	0.89	0.94	0.91	17	0.94	0.97	0.95	30
Acc			**0.90**	20			**0.91**	32			**0.94**	52
Mac Avg	0.90	0.90	0.90	20	0.91	0.90	0.91	32	0.94	0.94	0.94	52
Wt Avg	0.90	0.90	0.90	20	0.91	0.91	0.91	32	0.94	0.94	0.94	52

C. Evaluation Protocol

We evaluate the performance of *cAPTured* method under three circumstances – (a) in an continual learning environment, (b) in a limited data environment, and (c) on the full benchmark dataset. These 03 evaluating situations answer – (a) how well the model adapts to novel data samples, (b) how well the model is able to preserve the accuracy (mitigating catastrophic interference) of previous data embeddings, and (c) how well the model performs in limited data availability, (d) how well the model performs compared to present SOTA methods on the benchmark dataset. The baseline models were trained on the benchmark datasets.

1) Evaluation in Continual Learning Environment: In the first experiment, the model's continual learning ability was evaluated. The model was trained on 1164 samples (80% samples), while the remaining 292 samples (20% samples) were input as an continual-data stream ($R1$, $R2$, $R3$, $R4$, $R5$ each containing 50 samples) during testing for the model to learn in a continual learning setting. It is to be noted that the samples in the continual learning (lab setting) were unseen previously by the model and were not used during training the classifying layer neuron ($\mathcal{CLN}$) or the feature extraction networks. The results obtained are shown in Table II. The experiment demonstrates the model's ability and consistency of performance in predicting aptamer-target protein interaction pairs while updating its feature space in real-time over new aptamers that are being continuously discovered. Moreover, since *cAPTured* is the *first* continual learning method for Aptamer-target protein interaction prediction, it serves as the baseline for continual learning tasks in the domain. We also discuss the hyperparameters chosen during these experiments. From the obtained results, we can conclude that the model has consistent performance, and it overcomes catastrophic forgetting.

2) Evaluation in Limited Data Environment: The limited availability of aptamer-target protein interaction datasets is the major challenge in developing *in situ* methods for their screening. Currently only 2 databases exists – *Aptamer Base* and *Aptagen*, both of which we have utilized in our study. Since, physicochemical properties (e.g., hydrophilicity, hydrophobicity, average accessible surface area, and polarity) and biochemical contacts (residues contacts, atom contacts, salt bridges, and hydrogen bonds) affect protein interactions, we have used 32 structure-based and sequence-based properties from protein sequences. Most previous studies [14, 15, 16, 17] have not utilized these features. In the second experiment, the model was trained on limited data subset configuration $S1$, $S2$ and $S3$ which had $n = 100$, $n = 200$ and $n = 300$ samples respectively. Here 'n' represents the number of samples taken. The model's classification ability was tested on each subset configuration to establish the generalizability of the model's ability to make predictions on limited subsets of data. Table III shows the prediction performance of the model and compare the obtained results with varying data subset configurations.

3) Evaluation on Benchmark dataset: In contrast to the previous experiment, in which the model's classification performance was assessed on a limited data subset, here we analyze the model on the entire benchmark dataset. The classification results are presented in Table IV. We compare our results with all existing aptamer-target protein interaction prediction methods. While these methods are based on a 'closed-world assumption', we propose a continual learning model and significantly outperform them (3%).

D. Analysis and Performance Comparison

For encoding aptamer sequences, we relied on using k-mer frequency due to its simplicity and wide-acceptance. k-mer encodes more sequence information compared to other encoding techniques. Extending the same for aptamers has yielded successful results. Similarly, we used PseAAC to represent target-protein sequences. We did not use the commonly used AAC strategy since it leads to a loss in protein-sequence information. Moreover, in previous methods, PseAAC has been utilized and had successful outcomes. Therefore, we used PseAAC for representing protein target sequences.

TABLE IV
Comparison of obtained aptamer-target protein interaction prediction with baselines and state-of-the-art methods on Benchmark data. Here, "A" is the dataset by [14] and "A+B" is the benchmark dataset by [18]. Here, IDE: Imbalanced dataset environment, LDE: Limited data environment, CLE: Continual learning environment.

	Methods	Dataset	Evaluation Metrics					IDE	LDE	CLE	Feature encoding		Predictor
			ACC	Prec	Rec	F1	MCC				Aptamer encoding	Protein encoding	
Baseline	Shallow Net	A+B	0.687	0.382	0.212	0.272	0.101	-	-	✗	k-mer, revck-mer	PseAAC	-
	K-NN	A+B	0.601	0.380	0.136	0.200	0.011	-	-	✗	k-mer, revck-mer	PseAAC	-
	RF	A+B	0.795	0.396	0.240	0.291	0.196	-	-	✗	k-mer, revck-mer	PseAAC	-
	SVM	A+B	0.548	0.390	0.032	0.058	-0.020	-	-	✗	k-mer, revck-mer	PseAAC	-
SOTA	[14]	A	0.774	-	0.483	-	0.372	✗	✗	✗	NAC	AAC, PseAAC	RF
	[15]	A	0.719	-	0.738	-	0.398	✗	✗	✗	PseKNC	DCT, DI, PSSM	3 RF ens.
	[16]	A	0.745	-	0.773	-	0.450	✓	✗	✗	AAC, PseKNC	SAE	3 SVM ens.
	[31]	A	0.819	-	0.287	-	0.467	✗	✗	✗	PseKNC	TPC	Monte Carlo Tree
	[17]	A	0.842	-	0.641	0.664	0.557	✓	✗	✗	AAC, PseAAC, C/T/D, SOC	NAC, PseKNC, NMBAC	Adaboost+RF
	[18]	A+B	0.913	0.909	0.890	0.899	0.824	✓	✗	✗	k-mer, revck-mer	PseAAC	MLP
Ours	**cAPTured**[a]	A+B	**0.942**	**0.940**	**0.940**	**0.940**	**0.884**	✓	✓	✓	k-mer, revck-mer	PseAAC	-

[a]Hyper-parameters $\theta = 0.25$, $\gamma = 2$.

Table IV summarizes the evaluation results of all baselines and compares *cAPTured* to existing state-of-the-art methods. From Table IV, we can observe that:

(1) Past approaches [16] have mainly used ML models in ensemble as classifiers. Since number of features (i.e., dimensionality of feature vectors) far exceeds the number of training samples, most ML models (including SVM, RF) fails to achieve the required performance levels. Using SVM does not yield high accuracy since SVM is not convenient on noisy dataset where target classes are overlapping. Further, SVM has low explainability since for classification above and below the hyperplane, there is no probabilistic explanation.

(2) Deep learning (DL) approaches are ineffective due to limited data availability. [18] proposed *AptaNet* based on DL based multi-layered perceptron. However, study reported an oscillation in loss and accuracy. This was due to the limited number of training samples, since DL models require large volumes of data. To address this issue, *cAPTured* uses shape-morphing architecture which makes it effective in achieving performance even on limited datasets as demonstrated through the set of experiments.

E. Further Analysis and Insights

1) Impact of the Hyperbox Expansion Coefficient (θ): To assess the influence of model parameters, an in-depth parametric study was performed where the effect that variations in hyperbox expansion coefficient (θ) and fuzziness control parameter (γ) have on (1) the number of hyperboxes ($\mathcal{H}$) formed during training, and (2) the model training time (sec) were analyzed. Obtained results indicate that, as the hyperbox expansion coefficient (θ) increases, the number of hyperboxes formed during training increases in exponentially (rather than in a linear manner) [32, 33]. Figure 2(a) depicts this graphically.

2) Temporal Complexity Analysis: In addition to the parametric study, a temporal complexity analysis is performed. The

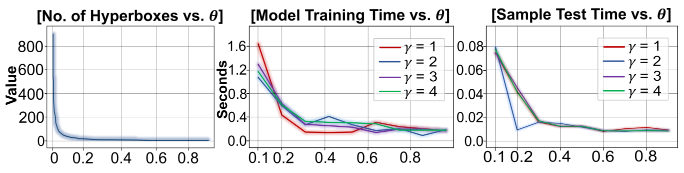

Fig. 2. Plots obtained from the parametric study and temporal complexity analysis on the benchmark dataset (a) Effect of hyperbox expansion coefficient (θ) on number of hyperboxes formed (b) Effect of hyperbox expansion coefficient (θ) and fuzziness control parameter (γ) on the sample testing time (c) Effect of hyperbox expansion coefficient (θ) and fuzziness control parameter (γ) on model training time

sample testing time (sec) for the prediction task is plotted graphically by adjusting the hyperparameters. The experiment is performed several times, varying the hyperparameters to determine the total model training time. The obtained study indicates that the model's training time on the overall dataset is comparable (i.e., 0.2 to 2 sec) to the sample test time (which ranges from 0.01 to 0.1 sec). Usually, such a substantial gap is not found in low-dimensional data categorization tasks [34-35]. As shown in Figure 2(b), the model training time is higher until 0.2, after which it reduces 'exponentially' for varying γ.

3) Visualization of post-embedding and final configuration: T-SNE plots representing feature spaces before and after the

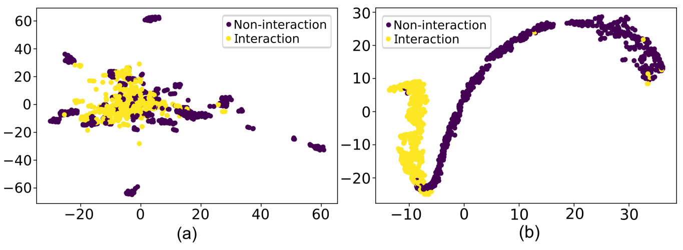

Fig. 3. t-SNE plots: (a) Before (b) After. From the plot, it can be inferred that our method aligns both classes by minimizing the gap.

changes brought about by our training methodology are shown in Figure 3. The t-SNE plots give a visual of how the classification task is performed by the proposed model. As evident from Figure 3, during incremental setup, *cAPTured* refines weights efficiently to push the classifier weights away from uncertain areas, resulting in better decision boundary.

V. Conclusion and Future Work

Interactions between aptamers and proteins are significant in physiological activities as well as molecular identification. In this paper, we propose a fuzzy continual learning method called *cAPTured*, which combines continual learning as a novel bioinformatics tool for screening aptamer-target protein interactions while saving on training data. *cAPTured* first performs feature encoding (1) k-mer and revck-mer based aptamer-seq encoding, (2) AAC and PseAAC based target protein-seq encoding, and then fuses the encoded features and performs feature embedding in a low-dimensional latent space, preserving more statistically significant features and finally, screens for potential aptamer-target protein interaction pairs in a fuzzy continual learning environment. Experimental results on real interaction data indicate the superiority of *cAPTured* both over baselines and state-of-the-art methods. In addition, the performance of *cAPTured* on both limited data subset configuration and continual learning environment demonstrate that *cAPTured* is effective, robust and prevents the model from becoming outdated with time. Future work includes extending the proposed algorithm to assign aptamer sequences to their corresponding families on the basis of their conserved primary sequences and secondary structures [36]. Secondly, fuzzy continual learning based bioinformatics methods can be developed for other domains like EEG [37], Brain-Computer Interface [38], visual aids [39], etc.

References

[1] A. D. Ellington and J. W. Szostak, "In vitro selection of RNA molecules that bind specific ligands," Nature, vol. 346, no. 6287, pp. 818–822, 1990, doi: 10.1038/346818a0.

[2] D. L. Robertson and G. F. Joyce, "Selection in vitro of an RNA enzyme that specifically cleaves single-stranded DNA," Nature, vol. 344, no. 6265, pp. 467–468, 1990, doi: 10.1038/344467a0.

[3] A. B. Iliuk, L. Hu, and W. A. Tao, "Aptamer in bioanalytical applications," Anal. Chem., vol. 83, no. 12, pp. 4440–4452, 2011, doi: 10.1021/ac201057w.

[4] M. Ashrafuzzaman, "Aptamers as both drugs and drug-carriers," Biomed Res. Int., vol. 2014, p. 697923, 2014, doi: 10.1155/2014/697923.

[5] J. M. Binning et al., "Development of RNA aptamers targeting Ebola virus VP35," Biochemistry, vol. 52, no. 47, pp. 8406–8419, 2013, doi: 10.1021/bi400704d.

[6] M. E. Jaax et al., "Complex formation with nucleic acids and aptamers alters the antigenic properties of platelet factor 4," Blood, vol. 122, no. 2, pp. 272–281, 2013, doi: 10.1182/blood-2013-01-478966.

[7] P. Wang, Y. Yang, H. Hong, Y. Zhang, W. Cai, and D. Fang, "Aptamers as therapeutics in cardiovascular diseases," Curr. Med. Chem., vol. 18, no. 27, pp. 4169–4174, 2011, doi: 10.2174/092986711797189673.

[8] T. Tŏpala et al., "New sulfonamide complexes with essential metal ions [Cu (II), Co (II), Ni (II) and Zn (II)]. Effect of the geometry and the metal ion on DNA binding and nuclease activity. BSA protein interaction," J. Inorg. Biochem., vol. 202, no. 110823, p. 110823, 2020, doi: 10.1016/j.jinorgbio.2019.110823.

[9] C. Tuerk and L. Gold, "Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase," Science, vol. 249, no. 4968, pp. 505–510, 1990, doi: 10.1126/science.2200121.

[10] M. Flamme, L. K. McKenzie, I. Sarac, and M. Hollenstein, "Chemical methods for the modification of RNA," Methods, vol. 161, pp. 64–82, 2019, doi: 10.1016/j.ymeth.2019.03.018.

[11] C. Zhu, G. Yang, M. Ghulam, L. Li, and F. Qu, "Evolution of multi-functional capillary electrophoresis for high-efficiency selection of aptamers," Biotechnol. Adv., vol. 37, no. 8, p. 107432, 2019, doi: 10.1016/j.biotechadv.2019.107432.

[12] Y.-S. Ding, T.-L. Zhang, and K.-C. Chou, "Prediction of protein structure classes with pseudo amino acid composition and fuzzy support vector machine network," Protein Pept. Lett., vol. 14, no. 8, pp. 811–815, 2007, doi: 10.2174/092986607781483778.

[13] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in Artificial Intelligence in Medicine, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 63-66, doi:10.1007/3-540-48229-6_9

[14] B.-Q. Li, Y.-C. Zhang, G.-H. Huang, W.-R. Cui, N. Zhang, and Y.-D. Cai, "Prediction of aptamer-target interacting pairs with pseudo-amino acid composition," PLoS One, vol. 9, no. 1, p. e86729, 2014, doi: 10.1371/journal.pone.0086729.

[15] L. Zhang, C. Zhang, R. Gao, R. Yang, and Q. Song, "Prediction of aptamer-protein interacting pairs using an ensemble classifier in combination with various protein sequence attributes," BMC Bioinformatics, vol. 17, no. 1, p. 225, 2016, doi: 10.1186/s12859-016-1087-5.

[16] Q. Yang, C. Jia, and T. Li, "Prediction of aptamer-protein interacting pairs based on sparse autoencoder feature extraction and an ensemble classifier," Math. Biosci., vol. 311, pp. 103–108, 2019, doi: 10.1016/j.mbs.2019.01.009.

[17] J. Li, X. Ma, X. Li, and J. Gu, "PPAI: a web server for predicting protein-aptamer interactions," BMC Bioinformatics, vol. 21, no. 1, p. 236, 2020, doi: 10.1186/s12859-020-03574-7.

[18] N. Emami and R. Ferdousi, "AptaNet as a deep learning approach for aptamer-protein interaction prediction," Sci. Rep., vol. 11, no. 1, p. 6074, 2021, doi: 10.1038/s41598-021-85629-0.

[19] J. Cruz-Toledo et al., "Aptamer Base: a collaborative knowledge base to describe aptamers and SELEX experiments," Database (Oxford), vol. 2012, p. bas006, 2012, doi: 10.1093/database/bas006.

[20] B. Liu, F. Liu, L. Fang, X. Wang, and K.-C. Chou, "repDNA: a Python package to generate various modes of feature vectors for DNA sequences by incorporating user-defined physicochemical properties and sequence-order effects," Bioinformatics, vol. 31, no. 8, pp. 1307–1309, 2015, doi: 10.1093/bioinformatics/btu820.

[21] S. Kawashima, P. Pokarowski, M. Pokarowska, A. Kolinski, T. Katayama, and M. Kanehisa, "AAindex: amino acid index database, progress report 2008," Nucleic Acids Res., vol. 36, no. Database issue, pp. D202-5, 2008, doi: 10.1093/nar/gkm998.

[22] M. M. Gromiha, "A statistical model for predicting protein folding rates from amino acid sequence with structural class information," J. Chem. Inf. Model., vol. 45, no. 2, pp. 494–501, 2005, doi: 10.1021/ci049757q.

[23] Z. Chen et al., "iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences," Bioinformatics, vol. 34, no. 14, pp. 2499–2502, 2018, doi: 10.1093/bioinformatics/bty140.

[24] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," IEEE Trans. Syst. Man Cybern. B Cybern., vol. 39, no. 2, pp. 539–550, 2009, doi: 10.1109/TSMCB.2008.2007853.

[25] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," IEEE Trans. Syst. Man Cybern., vol. SMC-2, no. 3, pp. 408–421, 1972, doi: 10.1109/tsmc.1972.4309137.

[26] B. Alpern and L. Carter, "The hyperbox," in Proceeding Visualization '91, 2002, pp. 133–139, doi: 10.1109/VISUAL.1991.175790

[27] P. K. Simpson, "Fuzzy min-max neural networks. I. Classification," IEEE Trans. Neural Netw., vol. 3, no. 5, pp. 776–786, 1992, doi: 10.1109/72.159066.

[28] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," IEEE Trans. Neural Netw., vol. 11, no. 3, pp. 769–783, 2000, doi: 10.1109/72.846747.

[29] A. V. Nandedkar and P. K. Biswas, "A General Reflex Fuzzy Min-Max Neural Network," Engineering Letters, vol. 14, no. 1, pp. 195–205, 2007.

[30] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, doi: 10.48550/ARXIV.1603.04467.

[31] G. Lee, G. H. Jang, H. Y. Kang, and G. Song, "Predicting aptamer sequences that interact with target proteins using an aptamer-protein interaction classifier and a Monte Carlo tree search approach," PLoS One, vol. 16, no. 6, p. e0253760, 2021, doi: 10.1371/journal.pone.0253760.

[32] A. Chharia and N. Kumar, "Foreseeing survival through 'fuzzy intelligence': A cognitively-inspired incremental learning based de novo model for breast cancer prognosis by multi-omics data fusion," in Predictive Intelligence in Medicine, Cham: Springer International Publishing, 2021, pp. 231–242, doi: 10.1007/978-3-030-87602-9_22.

[33] A. Chharia and A. Narayan, "A novel fuzzy approach towards in silico B-cell epitope identification inducing antigen-specific immune response for Vaccine Design," in 2021 IEEE 21st International Conference on Bioinformatics and Bioengineering (BIBE), IEEE, 2021, pp. 1–6, doi: 10.1109/BIBE52308.2021.9635292.

[34] A. Chharia, R. Upadhyay, and V. Kumar, "Novel fuzzy approach to Antimicrobial Peptide Activity Prediction: A tale of limited and imbalanced data that models won't hear," 2021. In Proceedings of the NeurIPS 2021 AI for Science Workshop, Vancouver, BC, Canada, 13 December 2021, https://openreview.net/pdf?id=x0tzOYvapDl

[35] A. Chharia et al., "Deep-Precognitive Diagnosis: Preventing future pandemics by novel disease detection with biologically-inspired Conv-fuzzy network," IEEE Access, vol. 10, pp. 23167–23185, 2022, doi: 10.1109/access.2022.3153059.

[36] J. Perez Tobia, P.-J. J. Huang, Y. Ding, R. Saran Narayan, A. Narayan, and J. Liu, "Machine learning directed aptamer search from conserved primary sequences and secondary structures," ACS Synth. Biol., vol. 12, no. 1, pp. 186–195, 2023, doi: 10.1021/acssynbio.2c00462.

[37] N. Grover, A. Chharia, R. Upadhyay, and L. Longo, "Schizo-Net: A novel Schizophrenia Diagnosis framework using late fusion multimodal deep learning on Electroencephalogram-based Brain connectivity indices, IEEE Trans. Neural Syst. Rehabil. Eng., vol. PP, pp. 1–1, 2023, doi: 10.1109/TNSRE.2023.3237375.

[38] J. Kalra et al., "How Visual Stimuli evoked P300 is transforming the brain-computer interface landscape: A PRISMA compliant systematic review," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 31, pp. 1429–1439, 2023, doi: 10.1109/TNSRE.2023.3246588.

[39] A. Chharia and R. Upadhyay, "Deep recurrent architecture based scene description generator for visually impaired," in 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, 2020, pp. 136–141, doi: 10.1109/ICUMT51630.2020.9222441.

Appendix A
k-mer & revck-mer based Aptamer Encoding

k-mers are k-length DNA sub-sequences (A, T, C, and G). Since T (Thymine) in DNA is similar to U (Uracil) in RNA, each RNA sequence was converted to DNA by replacing U to T. In this work, each aptamer was encoded into 84-dimension (for $k = 3$) and 339-dimension (for $k = 4$). To estimate the revck-mer, *firstly*, the reverse complement of a DNA sequence is estimated. This is done by replacing T and A, exchanging G and C, and reversing the letters. The total possible number of reverse complement k-mer is 2^{2k-1} (for $k = 1, 3, 5, ...$) and $2^{2k-1} + 2^{k-1}$ (for $k = 2, 4, 6, ...$). In this study, $k = 3, 4$ and each aptamer is encoded as 44-dimensional vector (for $k = 3$) and 179-dimensional vector (for $k = 4$). *repDNA* (python implementation available at https://github.com/liufule12/repDNA) is utilized for generating aptamer characteristics [20].

Appendix B
PseAAC based target Protein Sequence Encoding

AAC is a protein sequence feature based on protein attributes that includes folding types, secondary structure, domain, sub-cellular location, etc. It estimates each amino acid's frequency in a protein sequence. For a protein sequence with N amino acid residues, residue frequency was considered as follows: $f(t) = \frac{N(t)}{N}$, where $N(t)$ is the number of amino acid type t. Similarly, PseAAC [12] is a group of descriptors for prediction of protein sub-cellular properties and has been used as an effective feature extraction method especially in bioinformatics. Given a protein chain P with N amino acid residues: $P = R_1R_2R_3...R_N$. The sequence order effect of protein can be represented by discrete correlation factor set:

$$\begin{cases} \theta_1 = \frac{1}{L-1}\sum_{i=1}^{L-1}\Theta(R_i, R_{i+1}) \\ \theta_2 = \frac{1}{L-2}\sum_{i=1}^{L-2}\Theta(R_i, R_{i+2}) \\ \theta_3 = \frac{1}{L-3}\sum_{i=1}^{L-3}\Theta(R_i, R_{i+3}) \\ ... \\ \theta_\lambda = \frac{1}{L-\lambda}\sum_{i=1}^{L-\lambda}\Theta(R_i, R_{i+\lambda}), (\lambda < L) \end{cases} \tag{7}$$

where θ_1, θ_2, ..., θ_λ are 1-tier, 2-tier, and λ^{th}-tier correlation factors respectively. The overall correlation function is,

$$\Theta(R_i, R_j) = \frac{1}{3}\Big\{\big[H_1(R_j) - H_1(R_i)\big]^2 + \big[H_2(R_j) - H_2(R_i)\big]^2 + \big[M(R_j) - M(R_i)\big]^2\Big\} \tag{8}$$

where, $H_1(R_i)$, $H_2(R_i)$, and $M(R_i)$ are some properties (e.g., physicochemical, conformational, and energetic) value for the amino acid R_i; and also $H_1(R_j)$, $H_2(R_j)$, and $M(R_j)$ are the corresponding values of the amino acid R_j which is estimated as follows. Here, $H_1(i)$, $H_2(i)$, and $M(i)$ represent the original values of amino acid properties.

$$\begin{cases} H_1(i) = \dfrac{H_1^0(i) - \sum_{i=1}^{20}\frac{H_1^0(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[H_1^0(i) - \sum_{i=1}^{20}\frac{H_1^0(i)}{20}\right]}{20}}} \\ H_2(i) = \dfrac{H_2^0(i) - \sum_{i=1}^{20}\frac{H_2^0(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[H_2^0(i) - \sum_{i=1}^{20}\frac{H_2^0(i)}{20}\right]}{20}}} \\ M(i) = \dfrac{M^0(i) - \sum_{i=1}^{20}\frac{M^0(i)}{20}}{\sqrt{\frac{\sum_{i=1}^{20}\left[M^0(i) - \sum_{i=1}^{20}\frac{M^0(i)}{20}\right]}{20}}} \end{cases} \tag{9}$$

Overall, the vector representation of PseAAC for a protein sequence is given by $[V_1, V_2, ..., V_{20}, V_{21}, ..., V_{20+\lambda}]^T$. Here, T represents the transpose operator.

$$f(x) = \begin{cases} \frac{f_u}{\sum_{i=1}^{20} f_i + \omega\sum_{j=1}^{\lambda}\theta_j} & (1 \le u \le 20) \\ \frac{\omega\theta_{u-20}}{\sum_{i=1}^{20} f_i + \omega\sum_{j=1}^{\lambda}\theta_j} & (20+1 \le u \le 20+\lambda) \end{cases} \tag{10}$$

Here, f_i represents the frequency of occurrences of 20 amino acids and θ_j represents j^{th} tier sequence correlation factor found using Eqn. 1. ω shows the weight factor of the sequence order effect. For the study, $\omega = 0.05$. The first 20 components shows the AAC effect whereas the remaining components ($20+1$ to $20+\lambda$) show sequence order. The whole of $20+\lambda$ will be PseAAC. Thus $\lambda = 30$.