[1] **Hufsa Manzoor**

[2]**Arif Wani**

Selective Gradient Dropout in Deep Neural Networks for Linear B cell Epitope Prediction

Journal of Electrical Systems

Abstract: - This study focused on determining linear B-cell epitopes using a dataset of 50,000 experimentally validated linear epitopes from IEDB. The analysis showed that sequence-derived features were crucial for predicting BCEs. A deep learning framework was developed to improve feature representation and efficiency. Despite the power of deep neural networks, they require regularization technique to perform well. However, applying dropout to crucial recurrent connections in networks like LSTM can disrupt information flow. To address this, we introduce a novel method based on selective gradient dropout for deep fully connected layers. This novel method selectively freezes specific connections during training, improving network sparsity and efficiency while maintaining performance. Experimental results confirm its effectiveness in mitigating overfitting and enhancing computational efficiency. Experimental results show that this sparse network outperforms traditional methods in classification tasks achieving an average 86.3% accuracy. Thus being a significant improvement compared to existing tools.

Keywords: B Cell Epitopes; Dropout; Gradient Descent Dropout; LSTM

I. INTRODUCTION

B-cell epitopes (BCEs) are antigen surface regions that trigger immune responses by binding to designated antibodies, therefore vital in immunological research, including developing peptide-based vaccines, production of antibodies, and helping tackle diseases. B-cell epitopes are of two types: linear epitopes, composed of sequential amino acid residues, and conformational epitopes, formed by folded protein structures. Linear epitopes are epitopes with continuous sequence of amino acids and are relatively easier to predict [1]. They are valuable in designing peptide vaccines against viral diseases and targeting antibody responses in autoimmune conditions. Conformational epitopes, on the other hand, consist of spatially contiguous but non-continuous amino acid sequences. These epitopes are more challenging to decipher due to their complex structural nature but provide valuable insights into antigen-antibody interactions. Traditionally, experimental methods like peptide microarrays, X-ray crystallography, and ELISA have been used for BCE identification, but they are resource-intensive. Alternatively, computational approaches have shown promise in predicting linear BCEs from primary protein sequences or 3D structures [2]. Numerous computational methods have been developed, utilizing physicochemical properties like hydrophobicity, surface accessibility, and antigenicity. Various techniques, such as hidden Markov models, Naïve Bayes, SVM, BLOSUM62 scoring and ensemble methods have also been explored. Recent advancements include using more complex neural networks, such as deep maxout networks, deep ridge neural networks, and bidirectional LSTM networks.Advancements in computational capabilities and data processing have facilitated the creation of faster and more precise networks. These networks often feature dense and recursive architectures, enabling them to handle increasingly complex data. Deeper networks, in particular, stand out due to their significantly higher number of trainable parameters, offering exceptional adaptability to diverse input data. However, this advantage comes with a caveat. The networks are extremely flexible in terms of learning from the data which can lead to overfitting.In order to get the best performance from a model and simultaneously mitigate this inherent shortcoming, various regularization techniques such as cross validation, bagging[3], boosting weights[4], data augmentation[5], early stopping[4], and weight decay[6] have been introduced. Dropout [4] is one of the most effective regularization techniques. It mitigates the overfitting by removing random nodes such that they do not have any effect during training. It introduces randomness by temporarily deactivating some of the hidden units in a network. This intentional disruption prevents the network from becoming overly reliant on specific units, ensuring that each unit contributes independently to detecting features. While dropout typically enhances model performance by preventing over-reliance on certain activations, it does impact convergence time significantly. The frequent alteration of the loss descent path [7] extends the training duration as the model navigates towards a comprehensive representation, utilizing different components in varied combinations throughout iterations.

[1]*Hufsa Manzoor: Department of Computer Sciences, University of Kashmir, Hazratbal Srinagar, J&K 190006, India, hufsamanzoorbhat@gmail.com

[2]Department of Computer Sciences, University of Kashmir, Hazratbal Srinagar, J&K 190006, India, awani@uok.edu.in

This study introduces a novel adaptive training approach aimed at creating efficient models capable of achieving accurate and rapid generalization. Diverging from the conventional random dropout technique, this method involves freezing a portion of the network's parameters during each training step by nullifying specific gradients based on input information. The selection of gradients to be deactivated relies on networks with trainable parameters. Through experiments conducted on widely used IEDB [8] classification datasets, the effectiveness of this approach was demonstrated. The findings indicated that a general dropout based network was surpassed by a modified variant employing sparsity. This approach seeks to augment the conventional functionality of dropout by precisely dropping specific nodes in each training step. Unlike the conventional dropout technique, which might indiscriminately discard potentially crucial components of the network, this method aims to enhance dropout's efficacy by selective freezing of parameters.

II. CHALLENGES IN APPLYING DROPOUT TO RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNN) are specialized nerual networks tailored especially for sequential data processing, rendering them highly proficient for predictive tasks. RNNs are characterized by their ability to maintain an internal memory, allowing them to consider previous inputs when making predictions [6]. This feature is crucial for tasks where the sequence of data is more important than individual items. The diversity of applications for RNNs is highlighted by their ability to handle different types of input and output, making them a versatile tool in the field of data processing.Extensions of RNNs have been developed to address some of the limitations of standard RNNs, particularly the vanishing gradient problem [9]. These extensions introduce mechanisms for controlling data flow through the network, allowing them to better understand the underlying dependencies of the data. One such example being Long Short-Term Memory (LSTM). Despite their effectiveness, LSTMs face challenges such as lack of regularization using dropout [10].

Research has shown that using dropout, a regularization technique that turns off certain neurons during training to prevent overfitting, in Recurrent Neural Networks (like LSTMs) can lead to problems, especially when applied to the recurrent connections. This is because adding dropout to these connections can cause the model to become unstable and affect the data flow through these connections [11], making it difficult for the model to maintain the context and dependencies between them over various stages of sequence processing. In LSTM networks, the recurrent connections are crucial for passing information from one cell to the next. When dropout is applied to these connections, it can randomly turn off some of these connections during training, leading to a loss of information flow. This can result in the model being unable to effectively learn and transfer knowledge from one LSTM cell to another, thereby affecting the model's ability to accurately predict or understand sequences. Instead, it's recommended to apply dropout to feed-forward connections only [6], which are the connections that move data from one layer to the next without looping back.

From the findings, dropout cannot be effectively applied to a large number of parameters within the recurrent layers, leaving them without regularization. In simpler terms, applying dropout to RNNs, especially in the recurrent connections, can cause the model to become unstable. Instead, it's better to apply dropout to feed-forward connections.This highlights the importance of carefully considering where and how to apply dropout in LSTMs to avoid negatively impacting the model's performance [10]. It also underscores the ongoing efforts to develop new techniques that can effectively regularize RNNs without causing them to become unstable.

III. RELATED WORK

Lately, there has been a concerted effort in research to address the challenge of overfitting in complex artificial networks. Various adaptations and expansions of the dropout technique have emerged. For instance, DropConnect [12] induces sparsity within a network by randomly nullifying weights in a fully connected layer, rather than the output vectors of its activations. In a separate study [4], researchers introduced a regularization method that involves randomly dropping fractions of layers in highly complex residual networks. This strategy aims to promote sparsity during training while preserving the network's complexity during testing, thereby enhancing generalization without compromising performance. DropBlock [7], an extension of Cutout, employs a data augmentation approach to enhance performance. It aims to increase generalization by using feature maps within convolution filters. Other research endeavors focus on enhancing the generalization capability of networks with multiple branches [7]. They achieve this by preventing various paths to adapt to each other, either through random dropout of network segments or by altering activation functions.Certain extensions for dropout were introduced in various methods such as Maxout [13], which introduces new layers that are used for averaging dropout. In another study [14], noise is injected into gradients for regularizing the dropout to enhance the model.

Such variations include introducing some sort of randomness in the dropout. During training, a hidden unit's weight, activity, or gradient is probabilistically set to zero using some sort of Bernoulli distribution. Conversely, the approach discussed in [11] is similar to the discussed work. Here, the neuron is not dropped randomly rather it is affected by the input and the activation of the unit. In [15], units and weights were given an importance level. The aim was to reduce reliance between significant and insignificant network features and to develop meaningful connections between the unit's whileas at the same time minimizing the impact of dropping a unit.In [7], a learning-rate based dropout was proposed. At each step, the learning of a random unit is temporarily stopped and hence the learning rate is dropped to zero. While sharing similar conceptual grounds with this paper, it differs notably in the approach—prioritizing random freezing of network nodes over selective masking of less crucial units.

IV. DATASETS AND PREPROCESSING

A. The Dataset

B-cell epitopes were primarily sourced from the IEDB database. Machine learning techniques require a fixed pattern length for training, but mostly B-cell epitopes vary in length. Some of the previous studies suggest that the majority (approximately 90%) of epitopes are less than 20 amino acids long. Therefore, we chose to focus on epitopes with a maximum sequence length of 20, not as an optimized length but as a practical way to standardize the pattern size.Details of datasets used in the present study shows in Table 1.

Table 1: Details of datasets used in the study

Dataset	Source	Epitopes	Non Epitopes	Length
Benchmarking 1	Bcipep	700	700	20
Final	IEDB	50000	50000	20

To ensure the fairness of our prediction model, we removed identical epitopes from the dataset. The initial dataset comprised of 700 unique epitopes and non-epitopes each. This dataset constituted our benchmarking 1 dataset.

In the first iteration we used this benchmarking 1 dataset consisting of 700 positive and negative epitopes each. In the subsequent iterations, all the epitopes and non-epitopes were downloaded from IEDB. Following steps were taken to streamline the dataset:

• Both positive and negative epitopes of similar length were grouped together. A threshold length of 20 was chosen to form dataset of uniform length.

• Sequences with length greater than 20 were truncated using the technique implemented by DLBEpitope method in which same length is achieved by trimming or extending the sequence.

• 50,000 positive and negative epitopes were used to construct the final dataset. However, there is scope for using a bigger number. Also a non uniform dataset consisting of variable sequence lengths can also be taken into consideration.

• Only the sequences were taken into consideration and all other columns were cleaned up. This is to ensure that only sequences are used for feature extraction.

• Enhanced Dipeptide Composition was used in addition to feature clustering with k-means clustering for feature extraction.

B. Feature representation of peptides

For feature extraction, we used iFeature, a python library which extracts a wide range of sequence-derived features that capture different aspects of the protein or peptide. These features may include physicochemical properties, composition-based descriptors, structural properties, and more. Feature can be used for multiple numerical encoding schemes, feature extraction, clustering, ranking and reduction of dimensionality as well. We have used the following descriptors in the study:

a) Amino acid composition (AAC)

It is defined as presence of each amino acid with respect to each other in the sequence. Consider a sequence with N number of peptides. The Amino acid Composition for a jth amino acid can be calculated as:

AAC = number of occurences of jth amino acid / Total number of peptides (N)

b) Dipeptide composition (DPC)

It is defined as how often a given dipeptide will appear in a protein sequence P. It is a vector consisting of 400 (20x20) features and it is the multiple of standard 20 amino acids, hence it consists of 400 features of all possible dipeptides.

Consider a sequence with N number of peptides. For a dipeptide dij, the dipeptide composition is calculated as:

DPC = number of occurences of dij in the sequence / N – 1

c) AAP antigenicity scale (AAP)

It is defined as the ratio of how often the peptides occur in the positive set compared with the negative set. For each dipeptide, its antigenicity value is calculated by log of the ratio of occurences in positive and negative sets.

d) Evaluation Parameters

Sensitivity (SEN), Specificity (SPE), Positive Predictive Value (PPV), Accuracy (ACC) and Mathew's correlation coefficient (MCC) [16, 17] have been used to measure the performance of the prediction model. They have been defined in the following equations:

$$SEN=\frac{TP}{TP+FN}*100\%$$

$$SPE=\frac{TN}{TN+FP}*100\%$$

$$PPV=\frac{TP}{TP+FP}*100\%$$

$$ACC=\frac{TP+TN}{TP+FP+TN+FN}*100\%$$

$$MCC=\frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}*100\%$$

where TP is True Positive

TN is True Negative. This is the count of epitopes and non epitopes respectively which have been correctly predicted by the model.

FP is False Positive meaning the count of non-epitopes falsely identified as epitopes.

FN is False Negative meaning the count of epitopes falsely identified as non-epitopes by the model.

V. PROPOSED METHODOLOGY

In the initial model, we constructed a standard LSTM model with no complex call-backs or regularization techniques. This served as our baseline model, allowing us to understand the performance of the architecture without any additional measures. The model did not use any dropout and was a standard model and all of them were used for training the model.

A. Simple Dropout (0.5)

In the second model, we introduced a simple dropout [15] of 0.5 to our LSTM layers. The primary purpose of this was to mitigate overfitting and encourage generalization. The dropout was introduced to check the effect on the prediction accuracy.

Mathematically, first define a dropout probability: dropout_prob = 0.5. Then For each mini-batch of training data, apply dropout as follows:

For each neuron i in the layer:

Generate a random number r^i from a uniform distribution between 0 and 1: $r^i \sim U(0, 1)$.

If r^i<dropout_prob, retain the neuron: neuron_output=neuron_output_without_dropout.

If $r^i \geq$ dropout_prob, drop the neuron:

neuron_output = 0.

B. Adaptive Dropout

The third model introduced an adaptive dropout [18] mechanism. Here, we dynamically adjusted the dropout rate based on the validation loss. If the validation loss increased, we reduced the dropout rate, and vice versa. However, the adjustments were random i.e.; we just used some predefined value to add or subtract from the dropout values. This model included the dropout and the adaptive dropout function based on validation loss. The purpose of the Adaptive Dropout is to increment or decrement the dropout of the subsequent layer based on improvement

in validation loss. If the validation loss improves i.e; there is less loss in the model, then the dropout is adjusted accordingly by a predefined random incremental or detrimental factor. Layers and their inputs for adaptive dropout based model is shown in Fig. 1.

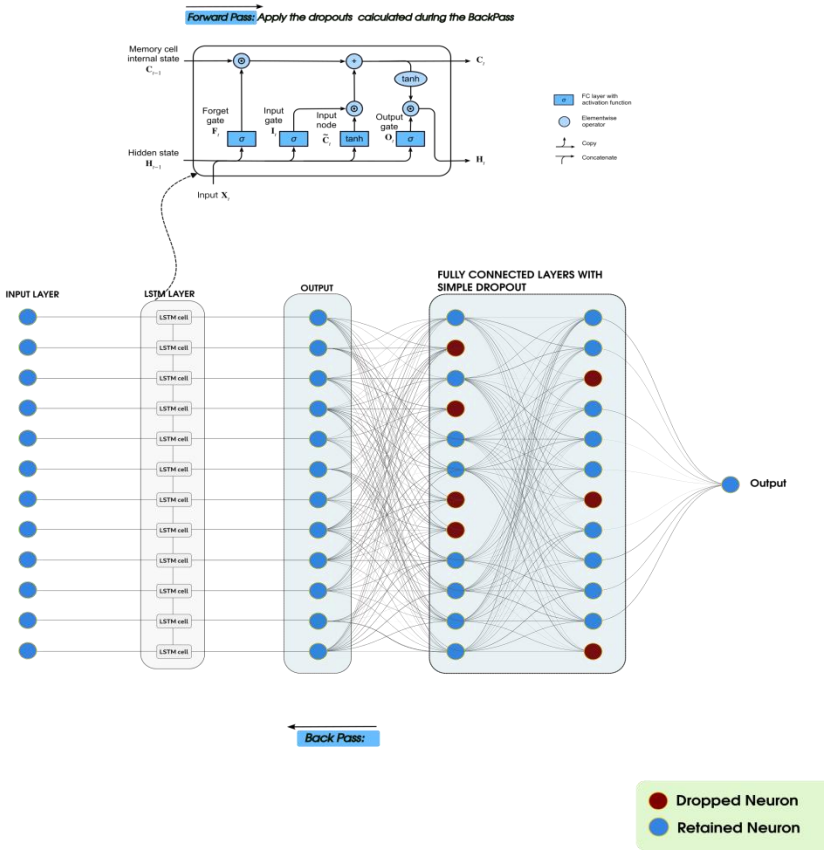

Fig. 1: Layers and their inputs for adaptive dropout based model.

We consider a neural network with N hidden layers. Let n ϵ 1, ..., N represent the hidden layers of the network and i represents hidden unit. Each connection between neurons is associated with weight and bias.

Calculate dropout rates for each neuron based on its output. For each neuron i, calculate p_i as a function of its output is given in equation (1):

$$Pi = 1 - (y_i^{(l+1)} / h) \quad (1)$$

Where h = constant and would be a hyperparameter that you can tune to control the dropout behaviour.

Generate random numbers $r_i^{(l)}$ from a uniform Bernoulli distribution between 0 and 1 for each neuron. Decide whether to keep or drop each neuron based on $r_i^{(l)}$ and Pi. If

$r_i^{(l)}$<Pi, retain the neuron;

Otherwise, drop it.

C. Gradient Descent Dropout

In the fourth and final version, we implemented a gradient decent dropout mechanism based on the value of each gradient given below. This iteration showcased the potential for improvements by optimizing dropout rates in response to the model's performance, resulting in improved accuracy and lower validation loss. This iteration defines and trains a recurrent neural network (RNN) model with Long Short-Term Memory (LSTM) layers using Tensor Flow/Keras. The model starts by preparing the data. It combines positive and negative examples and splits them into training, validation, and test sets using a standard 60-20-20 split ratio. It also standardizes the input features using the `Standard Scaler' [19]. A custom function 'GradientDropout' is defined to calculate the new dropout rate. This pre-defined function is used to determine the new dropout rate for each epoch. In this model, callbacks are used to aid in the prediction namely Gradient Descent. The gradient descent calculates the decay based on the epoch iteration.This callback is used together while training the model.In this variant, both RNN and LSTM are combined to make a deeper model and to enhance the prediction. Also, RMSProp optimizer [20] is used while compiling the model. Fig. 2 shows the layers and their inputs for Gradient Descent based model.

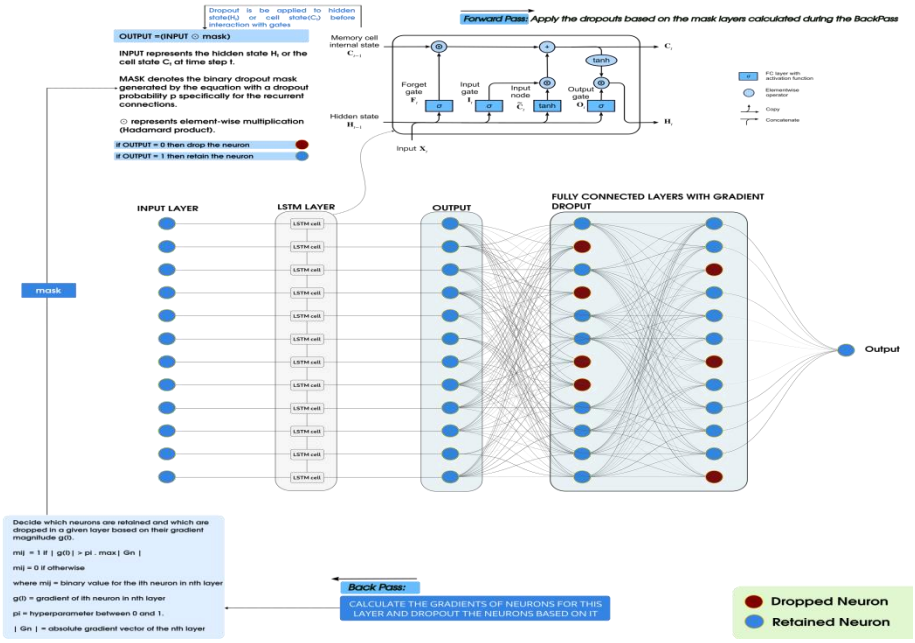

Fig. 2: Layers and their inputs for Gradient Descent based model.

Dropout is a widely used technique in training deep neural networks wherein a some percentage of neurons are dropped during each training iteration. This process helps mitigate the issue of overfitting. Reasons why we use dropout:

Co-Adaptation: Dropout's effect is often explained by its ability to prevent co-adaptation between nodes in the network. Co-adaptation refers to the situation where certain neurons become highly specialized in their responses and overly rely on the behaviour of neighbouring neurons.

Optimization technique [21]: an additional explanation for why dropout is effective. It suggests that dropout acts as an optimization technique that guides the input toward the saturation area of a nonlinear activation function. The saturation area of an activation function is where the function's output is close to its maximum or minimum value. The idea is that dropout accelerates the flow of gradient information even in this saturation area during backpropagation. While saturation areas are helpful for forward propagation and generalization, training neural networks in these areas is challenging because gradient information does not flow effectively, potentially causing the backpropagation process to struggle due to zero gradients.

The neural network comprises many hidden layer with many neurons and an output layer for binary classification. The method determines which nodes need to be leftout, thereby possibly improving training time and memory usage. This function dynamically computes the dropout rate for each neuron based on the gradient value during each training epoch. Following are the steps to selectively drop out gradients;

Consider a neural network with any number of hidden layers. Let it be denoted by N such that n ϵ 1, ...l, N

The connection between each of these neurons is associated with weight and bias. We implement the descent based dropout in the back propagation step. We use the gradient descent to train the network and every iteration updates the parameters w(l), as below:

First we calculate loss and expected output value y'.Loss FunctionL= (y'- yi)2

Then calculate the first derivative as given in equation (2) and (3):

$$\delta L / \delta w_i^{(l+1)} = \delta L / \delta yi * \delta yi / \delta w_i^{(l+1)} \quad (2)$$

Let's denote $\delta L / \delta w_i^{(l+1)}$ as g(l) , where g(l) is the gradient that is;

$$g(l) = \delta L / \delta w_i^{(l+1)} \quad (3)$$

We define a masking process to decide which neurons are retained and which are dropped in a given layer based on their gradient magnitude g(l).

$$mil = 1 \text{ if } | g(l)| > pi. \max|Gn|$$

mil = 0 if otherwise

where mil = binary value for the ith neuron in lth layer

g(l) = gradient of ith neuron in lth layer

pi= hyperparameter between 0 and 1. It represents masking threshold and higher value means more aggressive masking and vice versa.

| Gn| = absolute gradient vector of the nth layer

Finally, the neurons retained are only processed.

This updates the network neurons in each iteration and helps form new connections for the data flow. The selective dropout helps mitigate the overfitting by creating these paths that add to the predictive ability of the model.

VI. RESULTS AND DISCUSSION

To evaluate the performance of gradient descent based dropout, we compare the results in four aspects. First of all, we have adapted the sequence based features for this model. For training and testing an average of five sets has been used. We implemented four variations of datasets for training and testing. In addition, four different methods each using these datasets were implemented. In summary, the performance of the LSTM model was significantly better when using the selective gradient descent dropout technique.Accuracy and precision values of the models onthe datasets using simple dropout, adaptive dropout and selective gradient descent dropout are illustrated in Table 2 and Fig. 3 and 4.Comparison of the methods along with their dataset source and core methodology are shown in Table 3.

Table 2: Accuracy and precision values of the models

Method	Accuracy	Precision
BepiPred-2.0	0.72	0.68
	0.72	0.68
	0.76	0.72
ABCpred	0.60	0.60
	0.65	0.60
	0.70	0.68
BCpred	0.70	0.73
	0.73	0.70
	0.76	0.74
LBtope	0.68	0.72
	0.72	0.68
	0.8	0.77
iBCE-EL	0.72	0.74
	0.74	0.72
	0.8	0.75
DLBEpitope	0.76	0.72

Method	Accuracy	Precision
	0.76	0.74
	0.8	0.78
EpitopeVec	0.74	0.70
	0.74	0.70
	0.78	0.75
Normal Dropout based LSTM	0.58	0.60
Adaptive Dropout based LSTM	0.63	0.65
Gradient Descent based LSTM	0.86	0.82

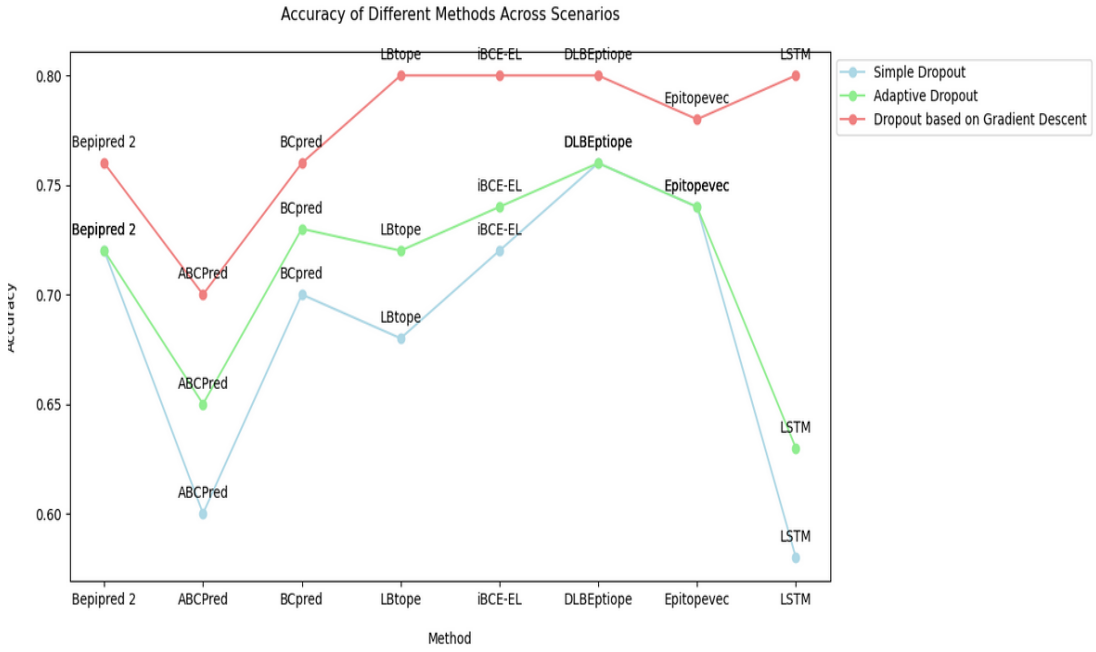

Fig. 3: Accuracy of these methods across different scenarios

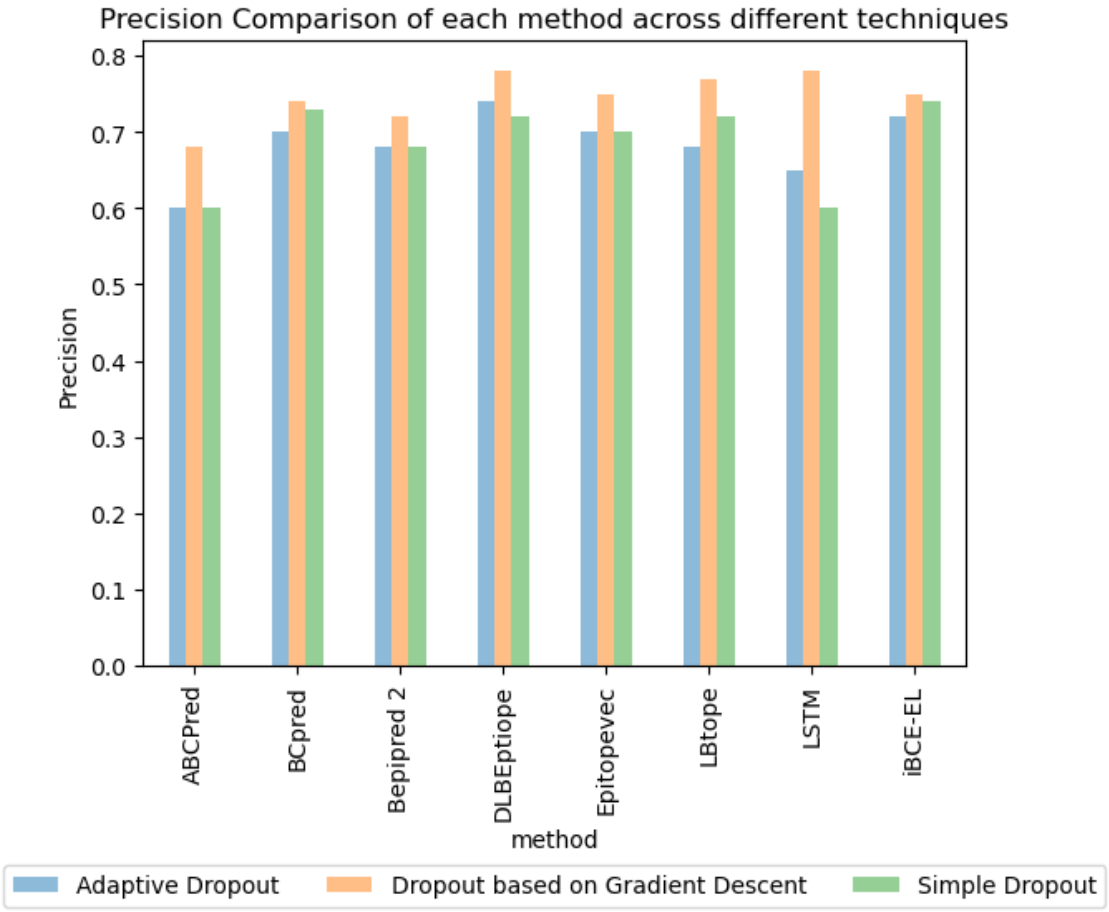

Fig. 4: Precision comparison of all these methods

Table 3: Comparison of the methods along with their dataset source and core methodology

Method Name	Core Methodology	Dataset Source	Dataset
BepiPred-2.0	Random Forest	Protein Data Bank	675 antigen-antibody crystal structures
ABCpred	FNN and RNN	BCIPep	700 epitopes
BCpred	String Kernel	Bcipep/ Swiss-Prot	701 epitopes 701 non-epitopes derived from Swiss-Prot
Lbtope	SVM	IEDB	14876 epitopes 23321 non-epitopes
iBCE-EL	Ensemble ERT and Gradient Boosting	IEDB	5550 validated Epitopes 6893 non BCE
DLBEpitope	Feed Forward Deep Neural Network	IEDB	25,884 linear B-cell epitopes 214,679 non-epitopes.
EpitopeVec	SVMs with the RBF kernel	Variable datasets	Variable
Gradient Descent based LSTM	LSTM with gradient Descent	IEDB	50000 epitopes 50000 non-epitopes

Numerous epitope prediction methods have been developed but achieving satisfactory accuracy remains a challenge [22]. To improve epitope prediction, efforts have focused on enhancing learning algorithms and identifying more effective features.

In this study, we introduced a novel approach by customizing the dropout of Long Short-Term Memory (LSTM) network for the prediction of B-cell epitopes. Using gradient descent dropout in LSTM offers a solution to address the problem of overfitting deep learning. While Support Recurrent Neural Networks (RNN) have been commonly used to enhance epitope prediction accuracy, our LSTM-based model demonstrated superior classification performance. This improvement can be attributed to the fact that existing network models often have a single structure and may not excel in text classification tasks. Additionally, feature selection has been a significant challenge in machine learning methods [23]. In our study, we used ifeature [24] for efficient feature calculations. Despite achieving very promising results in terms of accuracy for predicting epitopes, certain limitations exist and there is still room for model network optimization to further enhance predictive performance.Comparison of the various methods are illustrated in Table 4 and 5.

Table 4: Comparison of metrics of the methods

Method	ACC	SEN	SPE	PRE	F1 Score	AUC-ROC	MCC
BepiPred-2.0	High	High	Moderate	Moderate	High	High	High
ABCpred	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate
BCpred	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate
Lbtope	High	High	High	High	High	High	High
iBCE-EL	High	High	High	High	High	High	High
DLBEpitope	High	High	High	High	High	High	High
EpitopeVec	High	High	High	High	High	High	High
LSTM with gradient Descent	High	High	High	High	High	High	High

Table 5: Comparison of the methods

Method	Current Technology	Improvements
BepiPred-2.0	Utilizes Random Forest classifier and propensity scale.	Integration of additional features for enhanced prediction accuracy.
ABCpred	Utilizes recurrent neural network (RNN) for prediction.	Integration of attention mechanisms in RNN to improve feature extraction.
BCpred	Based on pair antigenicity and auto-correlation.	Integration of advanced machine learning algorithms for improved prediction performance.
Lbtope	Combines SVM and HMM algorithms for prediction.	Integration of deep learning models for enhanced prediction accuracy.
iBCE-EL	Utilizes ensemble learning approach for prediction.	Integration of new ensemble methods and feature engineering techniques for better performance.
DLBEpitope	Relies on deep learning, specifically convolutional neural networks (CNNs).	Integration of transfer learning and attention mechanisms in Deep Neural Networks for

Method	Current Technology	Improvements
		improved accuracy.
EpitopeVec	Utilizes word embedding technique and deep learning models.	Integration of contextual embeddings and transformer-based architectures for better epitope prediction.
LSTM with gradient Descent	Utilizes multiple feature extraction techniques along with selective gradient descent in LSTM network	Introduction of selective gradient descent that involves freezing a portion of the network's parameters during each training step by nullifying specific gradients based on input information.

VII. CONCLUSION

Owing to the impact of epitope prediction in mitigating various diseases and healthcare in general, the accurate prediction of epitopes is extremely essential. It has been a continous challenge for the researchers and multiple studies with numerous new methods have been dedicated towards it. We have introduced LSTM network with gradient based dropout for accurate prediction of epitopes. The study finds that using a customised selective dropout for the LSTM proves to be very promising in terms of performance for epitope predicition models developed on IEDB dataset.

CONFLICTS OF INTEREST

There are no conflicts to declare.

FUNDING DECLARATION

There is no funding agency for the present work.

REFERENCES

[1] Jespersen MC, Peters B, Nielsen M, Marcatili P. BepiPred-2.0: Improving sequence-based B-cell epitope prediction using conformational epitopes. Nucleic Acids Res. 2017; 45(W1):W24–9.

[2] Ponomarenko J, Bui HH, Li W, Fusseder N, Bourne PE, Sette A, et al. ElliPro: A new structure-based tool for the prediction of antibody epitopes. BMC Bioinformatics. 2008; 9:1–8.

[3] Gupta VK, Gupta A, Jain P, Kumar P. Linear B-cell epitopes prediction using bagging based proposed ensemble model. Int J Inf Technol. 2022 Dec 1;14(7):3517–26.

[4] Avgerinos C, Vretos N, Daras P. GradientDropout. Vol. 23, Sensors. MDPI; 2023.

[5] Perez L, Wang J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. 2017 Dec 13; Available from: http://arxiv.org/abs/1712.04621

[6] Semeniuta S, Severyn A, Barth E. Recurrent Dropout without Memory Loss. 2016 Mar 16; Available from: http://arxiv.org/abs/1603.05118

[7] Avgerinos C, Vretos N. Less Is More : Adaptive Trainable Gradient Dropout for Deep Neural Networks. 2023;

[8] Liu T, Shi K, Li W. Deep learning methods improve linear B-cell epitope prediction. BioData Min. 2020 Apr 17;13(1).

[9] Yuan X. BeeTLe: A Framework for Linear B-Cell Epitope Prediction and Classification. 2023 Sep 5; Available from: http://arxiv.org/abs/2309.02071

[10] Zaremba W, Sutskever I, Vinyals O. Recurrent Neural Network Regularization. 2014 Sep 8; Available from: http://arxiv.org/abs/1409.2329

[11] Hahn S, Choi H. Understanding Dropout as an Optimization Trick. 2018 Jun 25; Available from: http://arxiv.org/abs/1806.09783

[12] Park S, Song K, Ji M, Lee W, Moon I-C. Adversarial Dropout for Recurrent Neural Networks [Internet]. Available from: www.aaai.org

[13] Lian Y, Huang ZC, Ge M, Pan XM. An Improved Method for Predicting Linear B-cell Epitope Using Deep Maxout Networks. Biomed Environ Sci. 2015;28 (6):460–3.

[14] Ma L, Chen X, Wang B, Jin P. Gradient-based Dynamic Dropout. In: Proceedings - 2021 17th International Conference on Computational Intelligence and Security, CIS 2021. Institute of Electrical and Electronics Engineers Inc.; 2021. p. 137–40.

[15] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15:1929–58.

[16] Saha S, Raghava GPS. Prediction of continuous B-cell epitopes in an antigen using recurrent neural network. Proteins Struct Funct Genet. 2006 Oct 1;65 (1):40–8.

[17] Pai TW, Wang HW, Lin YC, Chang HT. Prediction of B-cell linear epitopes with a combination of support vector machine classification and amino acid propensity identification. J Biomed Biotechnol. 2011; 2011.

[18] Jimmy Ba Brendan Frey L. Adaptive dropout for training deep neural networks.

[19] Azim KF, Hasan M, Hossain MN, Somana SR, Hoque SF, Bappy MNI, et al. Introduction to Machine Learning with Python [Internet]. 2016. Available from: http://arxiv.org/abs/1906.11078

[20] Marca AFL, Lopes R da S, Lotufo ADP, Bartholomeu DC, Minussi CR. BepFAMN: A Method for Linear B-Cell Epitope Predictions Based on Fuzzy-ARTMAP Artificial Neural Network. Sensors. 2022;22 (11).

[21] Goodfellow IJ, Warde-Farley D, Mirza M, Courville A, Bengio Y. Maxout Networks. 2013.

[22] Angaitkar, P., Aljrees, T., Kumar Pandey, S. et al. Inferring linear-B cell epitopes using 2-step metaheuristic variant-feature selection using genetic algorithm. Sci Rep13, 14593 (2023). https://doi.org/10.1038/s41598-023-41179-1

[23] Bouchlaghem, Y., Akhiat, Y., and Amjad, S., "Feature Selection: A Review and Comparative Study", in E3S Web of Conferences, 2022, vol. 351. doi:10.1051/e3sconf/202235101046.

[24] Zhen Chen, Pei Zhao, Fuyi Li, André Leier, Tatiana T Marquez-Lago, Yanan Wang, Geoffrey I Webb, A Ian Smith, Roger J Daly, Kuo-Chen Chou, Jiangning Song, iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences, Bioinformatics, Volume 34, Issue 14, July 2018, Pages 2499–2502, https://doi.org/10.1093/bioinformatics/bty140