Chapter 5

Natural-Language Processing (NLP) based feature extraction technique in Deep-Learning model to predict the Blood-Brain-Barrier permeability of molecules

Summary:

Blood-Brain-Barrier (BBB) permeability is one of the critical factors in the success and failure of CNS drug development. The most accurate method of measuring BBB permeability involves clinical experiments, which are labour-intensive and time-consuming. Thus, numerous efforts were made to use artificial intelligence (AI) to predict molecules′ BBB permeability. Most of the previous models are based on calculated descriptors and molecular fingerprints. In the present work, we have developed an NLP-based feature extraction technique in Deep-Learning models to predict BBB permeability. We have used the B3DB database and generated SELFIES to extract features from the molecules. We have employed word level and N-gram tokenization to represent words into numeric vectors. The extracted features were fed into several Artificial Neural Network (ANN) and Bi-directional Long Short-Term Memory (LSTM) models. The model, ANN-10 built using ANN and 6-gram tokenization, performed best on the independent test set. The accuracy, precision, recall, F1, specificity and AUC of ROC scores were found to be 0.89, 0.91, 0.91, 0.91, 0.85 and 0.90. Thus, the developed model can be used for the early screening of CNS drugs.

5 Natural-Language Processing (NLP) based feature extraction technique in Deep-Learning model to predict the Blood-Brain-Barrier permeability of molecules

5.1 Introduction

The blood-brain-barrier (BBB) has been a bottleneck for discovering neurotherapeutics. Many drug molecules fail in the drug discovery and development phase due to the inability to cross the BBB. The two well-described barriers are the vascular BBB and the blood-cerebrospinal fluid (blood-CSF) barrier. The former consists primarily of the capillary bed, and the latter consists primarily of the choroid plexus [83]. The BBB regulates the entry of substances into the brain that are critical to CNS function. The barrier is created by the tight junctions between the cerebral endothelial cells, the epithelial cells of choroid plexus and the cells of the arachnoid epithelium [83]. The BBB only allows lipophilic molecules with a molecular weight of less than 500 Da from the bloodstream to enter the brain; other molecules require specific transport systems, which help move across membranes [84]. There are five basic mechanisms involved in the movement across the membranes. The first is passive (or simple) diffusion, in which the drug molecules travel based on the concentration gradient from a higher drug concentration to a lower concentration. Usually, molecules with high lipophilicity and low molecular weight can quickly diffuse across the BBB. Blood gases and several drugs, such as anaesthetics, crosses BBB in this manner [85]. Secondly, carrier-mediated transporters involve active and facilitated diffusion. These are ATP-binding transporters, such as P-glycoprotein (Pgp). Pgp is expressed naturally on the plasmatic membranes of endothelial cells. It protects the brain from harmful substances. Thirdly, the solute carriers (SLC) facilitate the movement of solutes bi-directionally across the cell membrane. SLC transport is carried out by concentration gradients or electrochemical gradients [86].

Some drugs, such as L-DOPA, enter the brain through such transporters [87]. Fourthly, receptor-mediated transcytosis (RMT) includes binding the macromolecules, such as proteins and peptides, to the receptors. This system utilizes the vesicular transport of endothelial cells to transport the substrates in the brain [88]. Finally, transendothelial migration or diapedesis of leukocytes, where the leukocytes penetrate the BBB directly through the cytoplasm without the disruption of tight junctions. Once the diapedesis of a leukocyte commits, it does not go back to the same cell [86] [89] **(Figure 5.1)**.

The *in-vivo* experiments to study the BBB permeability are one of the most accurate and reliable methods, but it is expensive and time-consuming. It is also challenging to perform such experiments for high throughput screening (HTS) in the early research stages.

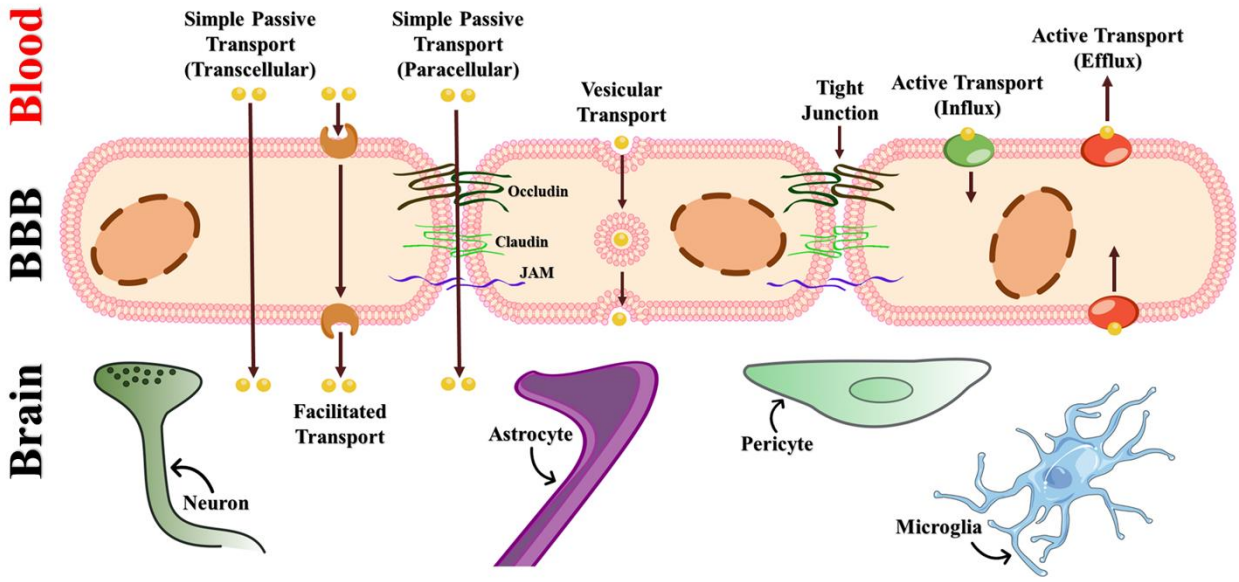

Figure 5.1 Basic mechanism involved in the transport of drugs across the BBB

Deep-Learning is a subset of machine learning which employs more than three layers of neural networks to learn from a massive amount of data. Deep-Learning eliminates most of the data preprocessing usually involved with machine learning. The neural networks can ingest and process unstructured data like texts or images and automate feature

extraction. In the present study, we have applied Artificial Neural Network (ANN) and Recurrent Neural Network (RNN) to predict the BBB permeability of a drug.

There are two types of *in-silico* models to predict BBB permeability, one is quantitative, and the other is qualitative. The quantitative models predict the logarithm of the drug's bain-plasma concentration ratio (LogBB), whereas the qualitative model predicts whether a compound is BBB permeable (BBB+) or not (BBB−). In the past several classification models have been build using machine learning and deep-learning algorithms. Kumar *et al*. developed the DeePred-BBB model using 3971 compounds with the help of physicochemical properties and substructure fingerprints. The model showed an accuracy of 98.07% using deep neural network (DNN), but the results were not validated on an external validation set [90].

Similarly, Alsenan *et al*. developed a Recurrent Neural Network (RNN) model using the dataset of 2350 compounds which was acquired from Wang et al. showed overall accuracy of 96.53% and a specificity score of 98.08 % [91]. A lightBBB model was developed by Shaker et al., which was trained using Light Gradient Boosting Machine (LGBM) on 7162 compounds and showed an overall accuracy of 93% and specificity of 77% [92]. The dataset used in the study consisted of several duplicates and molecules that could not be recognized by RDKit [93].

Another critical factor while developing predictive models is the use of molecular descriptors and fingerprints. The descriptors significantly affect the model's performance and pose a problem while deploying the model to the use case. It causes dependency on other packages and software to calculate the descriptors and then feed them into the model, thus making it difficult to use. One way to overcome this is to apply Natural Language Processing (NLP) on the chemical identifiers to extract meaningful information for the model. In a study performed by Parakkal *et al.*, they used a pre-trained model

Mol2Vec to generate the NLP-based descriptors. The AUC of the best model was found to be 0.96 [94]. In the present work, we have employed different NLP techniques to extract information from the molecules and then applied some deep-learning algorithms to train the model and get reliable predictions.

5.2 Methods

The deep-learning models were built on Python 3.10.7 using standard libraries and TensorFlow v2.11.0.

5.2.1 Dataset

The B3DB dataset, curated by Meng *et al.*, was used to develop the models. The dataset consists of 7807 ligands. This is the largest BBB dataset we know. The chiral specifications of the molecules were retained by using isomeric SMILES [93]. The dataset was checked for any missing or duplicated records. After careful curation, the dataset consisted of 4956 BBB permeable and 2851 BBB non-permeable compounds (Fig.2). The class label for BBB permeable was assigned as '1' and BBB non-permeable as '0'.

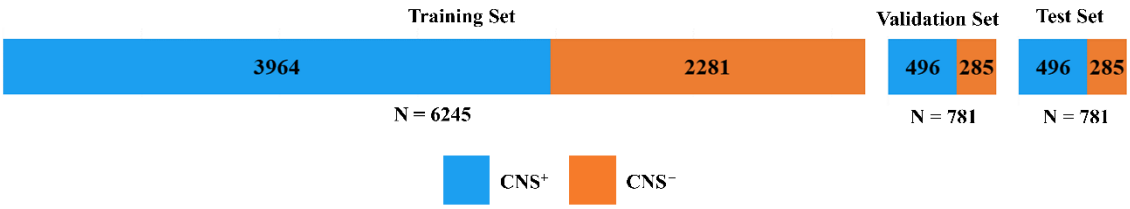

Figure 5.2 Distribution of dataset into training, validation and test set

5.2.2 Features extractions

The SMILES were converted to a more robust Self-Referencing Embedded String (SELFIES) to describe the molecular graph in a machine language. Each SELFIES corresponds to a valid molecule; every molecule can be described as SELFIES [95]. As deep-learning models work on differentiable functions which can process only the

numerical inputs in an array or tensor form. So the SELFIES need to be converted into numeric form before applying deep-learning algorithms. The SELFIES were converted into numerical tensors using vectorization. The SELFIES are split into units (called tokens), and every unit is indexed. The process of feature extraction from SELFIES has been summarized in Figure 5.3**.** For example, N-methylpropan-2-amine can be represented as 'CNC(C)C' in terms of SMILES. The same will be converted into SELFIES, which will be represented as '[C][N][C][Branch1][C][C][C]'. The SELFIES were split between '][' to generate units viz., '[C]','[N]','[C]','[Branch1]','[C]','[C]','[C]'. Now these individual units are tokenized. The tokenization of SELFIES was done in two ways (i) Word level tokenization and (ii) N-gram tokenization. In the case of word-level tokenization, the sequence of the string is retained; such types of models are also called 'sequence models'. On the other hand, in N-gram tokenization, the tokens are a group of N-consecutive words; for example, '[C] [N]' is a 2-gram token. Similarly, 1 to 6-gram tokens were generated. This approach treats the input as a set, discarding the original order; such models are called 'bag-of-words models. Once the SELFIES is split into tokens then they are hashed into a fixed binary vector. An index of all the unique tokens found in the training set is made and a number is assigned to them. If any new token comes up while tokenizing training set, it will be assigned as 'UNK'. In case of N-gram tokenization every combination is taken as individual vocabulary and the same is indexed. Now considering the above example the feature vocabulary will contain following unique tokens i.e. '[C]', '[N]' and '[Branch1]'. These tokens will be given a specific index number which will be used to assign the index corresponding to token present in the molecules of training and test dataset. The tokens were then one-hot encoded, or multi-hot encoded to create numeric vectors. More information was added by using the Term frequency-inverse document frequency (TF-

IDF) normalization. TF-IDF is a metric that weights a given term by taking term frequency, how many times the string pattern appears in a current molecule divided by the measure of frequency in the entire dataset [96]. Essentially, it measures the importance of a feature by comparing its frequency within a molecule with the entire training dataset.

$$TF(w) = \frac{No.of\ times\ the\ string\ w\ appears\ in\ a\ molecular\ representation}{Total\ number\ of\ string\ in\ a\ molecular\ representation}$$

$$IDF(w) = log\frac{Total\ number\ of\ molecules\ in\ training\ set}{Number\ of\ molecules\ comtaining\ the\ string\ w}$$

So, the weight of a string 'w' in molecule 'm' is given by the following TF-IDF weighting:

$$weight\ (w,m) = TF(w,m) * IDF(w)$$

There also exists an advanced string representation method called word embedding. Contrary to the previously mentioned techniques, word embeddings are low-dimensional and dense floating point vectors. In addition to being dense, word embeddings are structured representations, and their representation is learned from the data. They encode semantic relationships between features based on their contextual usage.

5.2.3 Neural Network Architecture

In the present study, we employed ANN and RNN to predict BBB permeability. ANN is a type of deep-learning model which employs multiple perceptrons at each layer. This is also called a feed-forward neural network because the direction of input processing is only in the forward direction [97]. On the other hand, RNNs are more complex; they save the output of a processing layer and feed the result back into the model. Bi-directional Long Short-Term Memory (LSTM) is a type of RNN model that can process the sequence information in both forward and backward directions.

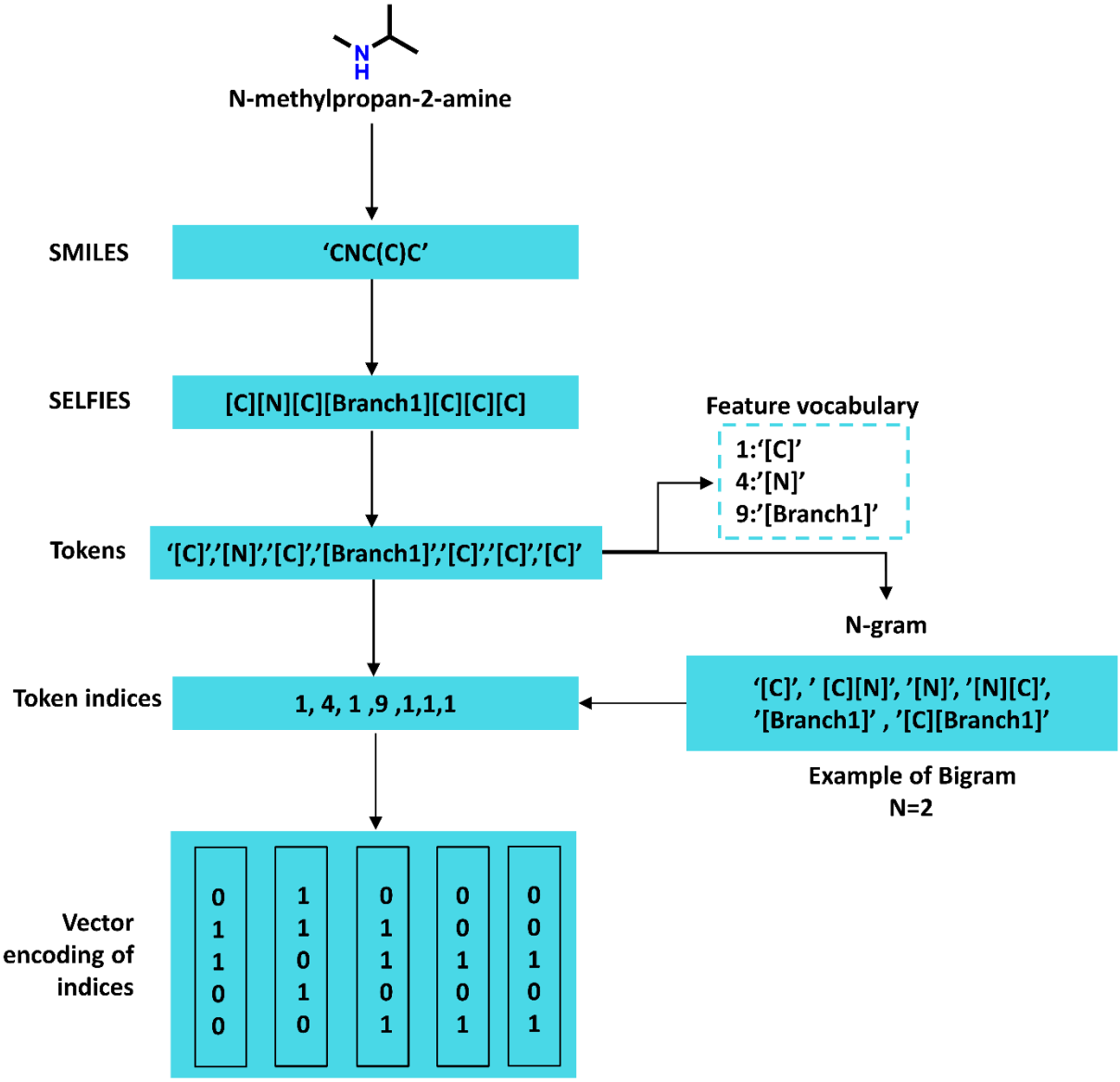

Figure 5.3 Steps of feature extraction from SELFIES

5.2.3.1 Input layer

The features extracted from the previous section were used as an input layer in the deep-learning models.

5.2.3.2 Hidden layer

The hidden layer in the case of ANN was made up of 6 layers, out of which three were dense with 512 units and three were dropout layers with a dropout rate of 0.2. The models were also tested with a dropout rate of 0.2, 0.3, 0.4 and 0.5. Dropout regularization was

used to prevent the models from overfitting. The dropout layers randomly drop the nodes as per the dropout rate. The Rectifier linear unit (ReLU) activation function was used for the dense layers **(Figure 5.4(a)).** The number of hidden layers was optimized (2-10 hidden layers); any reduction in the number of hidden layers led to a reduction in model performance, and an increment in the number of hidden layers did not show any improvements. Similarly, the number of neurons was also optimized (128, 256,512, 1024 and 2048).

In the case of the bidirectional LSTM model, the inputs were fed into an embedding layer which was then connected with a bidirectional LSTM layer of 128 units. The activation function in the forward direction was Tanh, and the recurrent activation function was selected as sigmoid. The LSTM layer was then connected with the dropout layer with a dropout rate of 0.2 **(Figure 5.4(b)).**

5.2.3.3 Output layer

The output layer comprised a dense layer with one neuron and 'sigmoid' as an activation function. The output layer gives predictions in the form of prediction probability which was then converted into classes (P>0.5 as BBB+ and P<0.5 as BBB-). The summary of parameters of both ANN and bi-directional LSTM model has been mentioned in **Table 5.1** and **Table 5.2,** respectively.

5.2.4 Model parameters

The models were trained with the RMSProp optimizer, and the learning rate was set to 0.001. Several learning rates were tried (0.01, 0.05, 0.001, 0.005 and 0.0001) but the best result was obtained with learning rate of 0.001. The optimizer minimizes the loss value from the binary cross-entropy cost function. Early stopping was used to prevent the model from overfitting. While performing early stopping, the patience was set to 25 epochs.

Patience is the number of epochs after which the training will stop if there is no improvement in the validation loss. The batch size was kept at 32.

5.2.5 Evaluating the performance of DL models

To evaluate the performance of binary classification models, we used the following metrics:

(a) Accuracy = $\frac{True\ positives + True\ negatives}{True\ positives + True\ negative + False\ positives + False\ negatives}$

(b) Precision=$\frac{True\ positives}{True\ positive + False\ positives}$

(c) Recall= $\frac{True\ positives}{True\ positive + False\ Negatives}$

(d) F1 score= $2\frac{Precision * Recall}{Precision + Recall}$

(e) Specificity= $\frac{True\ negatives}{False\ positive + True\ negatives}$

(f) The area under the ROC curve (AUC of ROC) - This is a standard binary classification evaluation metric. It demonstrates the model's ability to split the two classes through sensitivity and specificity.

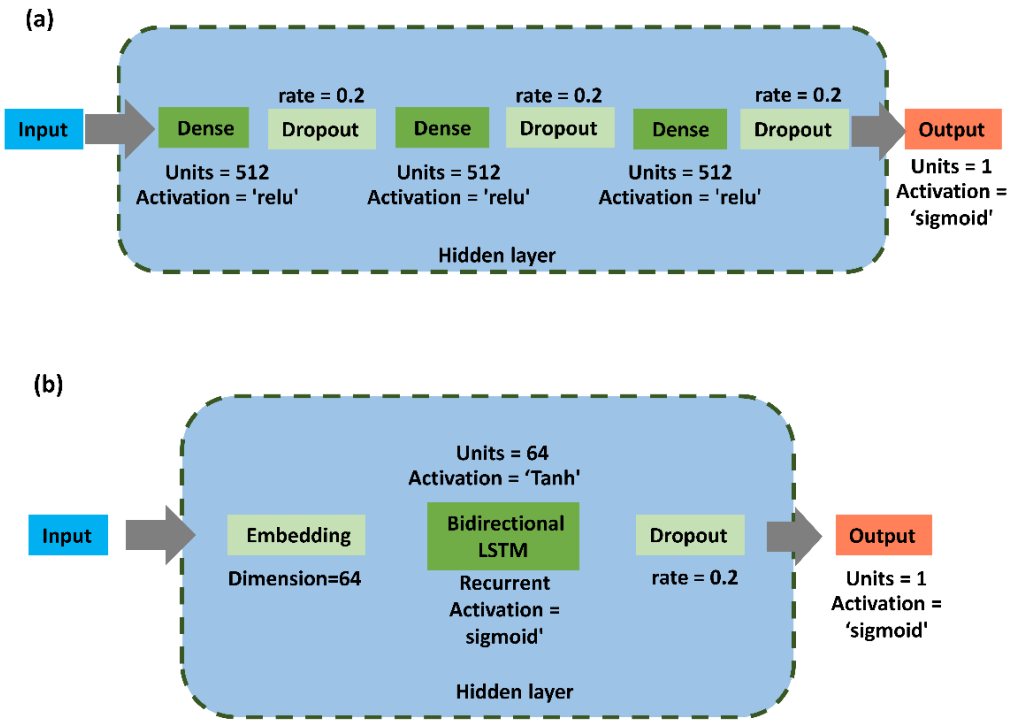

Figure 5.4 Neural network architecture of (a) ANN (b) LSTM

Table 5.1 Summary of the architecture of ANN model

Layer	Type	Output shape	No. of Parameters
Input	Input	(None, 235)	0
Hidden	Dense	(None, 512)	120832
	Dropout	(None, 512)	0
	Dense	(None, 512)	262656
	Dropout	(None, 512)	0
	Dense	(None, 512)	262656
	Dropout	(None, 512)	0
Output	Dense	(None, 1)	513

Table 5.2 Summary of the architecture of LSTM Model

Layer	Type	Output shape	No. of Parameters
Input	Input	(None, 235)	0
Hidden	Embedding	(None, 235, 64)	5952
	Bidirectional	(None, 128)	66048
	Dropout	(None, 128)	0
Output	Dense	(None, 1)	129

5.3 Result and Discussion

In the present work we have converted SMILES to SELFIES. SELFIES possess several advantages over SMILES, such as its robust molecular string representation. SELFIES never produces any invalid molecule, the combinations used in SELFIES alphabet map to a chemically relevant graph. In the case of SMILES, there is syntactic invalidity of unbalanced parentheses or ring identifiers. In the present work, if the feature were extracted based on SMILES, there could be a possibility that there may be a string with open parentheses but no corresponding closed parentheses, thus leading to an invalid molecular graph.

The text vectorization process led to the extraction of 91 unique features from the SELFIES representation. Any new feature which might appear in the test set will be assigned with 'UNK'. A list of all the extracted features has been summarized in a table (**Table S7 of appendix).** The performance of 17 deep-learning models (10-ANN and 7-bidirectional LSTM) was compared to select the best model and text representation technique.

Table 5.3 summarizes the result of loss and accuracy for the training set and test set of ANN models. It can be observed that the models with N-gram tokenization performed better than the models with word-level tokenization. The models viz., ANN-5, ANN-7, ANN-9 and ANN-10 showed an accuracy of 0.88 on the validation set. On the other hand, bidirectional LSTM models did not perform well as compared to the ANN models. The model LSTM-4, built using word-level tokenization with tf-idf encoding, showed the best accuracy among the bidirectional LSTM models. The accuracy of LSTM-4 on the validation set was found to be 0.84. Thus, the four models, ANN-5, ANN-7, ANN-9 and ANN-10, which performed better, were selected to evaluate their performance on an independent test set. The plot of loss and accuracy across epochs for the chosen models has been given in Figure 5.5**.** The plots for the remaining models have been given in the SI **(Figure S4-S10 of appendix)**. The summary of performance models on the test set has been summarized in **Table-5.** The raw confusion matrix has been shown in **Figure S10 of appendix**. The model ANN-10 showed the best accuracy, precision, and F1 score on the test set. The AUC of ROC plot has been represented in Figure 5.5**.** Thus, the model ANN-10 was selected from our study and used to make the predictions.

Table 5.3 Summary of the performance of ANN models on the Training set and Validation set

Model	Tokenization Layer	Output type	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
ANN-1	Word level	int	0.39	0.62	0.87	0.83
ANN-2	Word level	Multi-hot encoding	0.33	0.58	0.85	0.82
ANN-3	Word level	Count	0.25	0.41	0.87	0.86
ANN-4	Word level	tf-idf	0.24	0.41	0.88	0.85
ANN-5	**N-gram (N=1)**	**tf-idf**	**0.28**	**0.40**	**0.88**	**0.88**
ANN-6	N-gram (N=2)	tf-idf	0.25	0.68	0.97	0.82
ANN-7	**N-gram (N=3)**	**tf-idf**	**0.20**	**0.82**	**0.97**	**0.88**
ANN-8	N-gram (N=4)	tf-idf	0.28	0.55	0.92	0.85
ANN-9	**N-gram (N=5)**	**tf-idf**	**0.22**	**1.20**	**0.97**	**0.88**
ANN-10	**N-gram (N=6)**	**tf-idf**	**0.23**	**1.10**	**0.97**	**0.88**

Table 5.4 Summary of the performance of LSTM models on the Training set and Validation set

Model No.	Tokenization Layer	Output type	Training Loss	Validation Loss	Training accuracy	Validation Accuracy
LSTM-1	Word level	int	0.24	0.38	0.89	0.83
LSTM-2	Word level	Multi-hot encoding	0.64	0.66	0.8	0.77
LSTM-3		Count	0.34	0.44	0.84	0.83
LSTM-4		tf-idf	0.38	0.46	0.85	0.84
LSTM-5	N-gram (N=1)	tf-idf	0.37	0.45	0.84	0.83
LSTM-6	N-gram (N=2)	tf-idf	0.56	0.61	0.82	0.81
LSTM-7	N-gram (N=3)	tf-idf	0.49	0.49	0.82	0.80

Table 5.5 Summary of performance on the test set

Model	Accuracy	Precision	Recall	F1 Score	Specificity	AUC of ROC
ANN-5	0.87	0.87	0.93	0.90	0.87	0.92
ANN-7	0.88	0.90	0.90	0.90	0.85	0.88
ANN-9	0.86	0.85	0.94	0.89	0.89	0.86
ANN-10	**0.89**	**0.91**	**0.91**	**0.91**	**0.85**	**0.90**

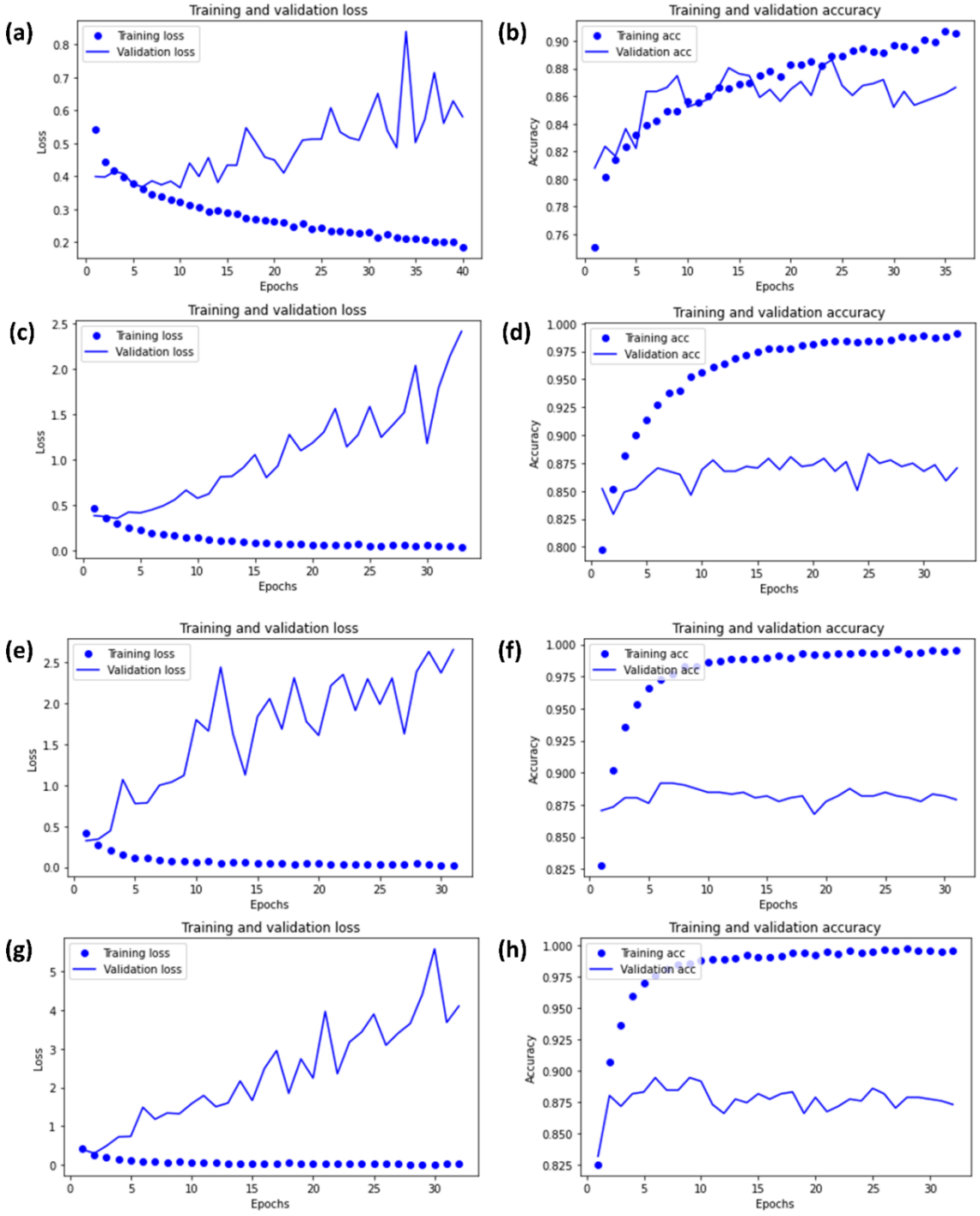

Figure 5.5 Performance of Model while training: (a) Training loss and validation loss for ANN-5 (b) Training and Validation accuracy for ANN-5 (c) Training loss and validation loss for ANN-7(d) Training and Validation accuracy for ANN-7 (e) Training loss and valid

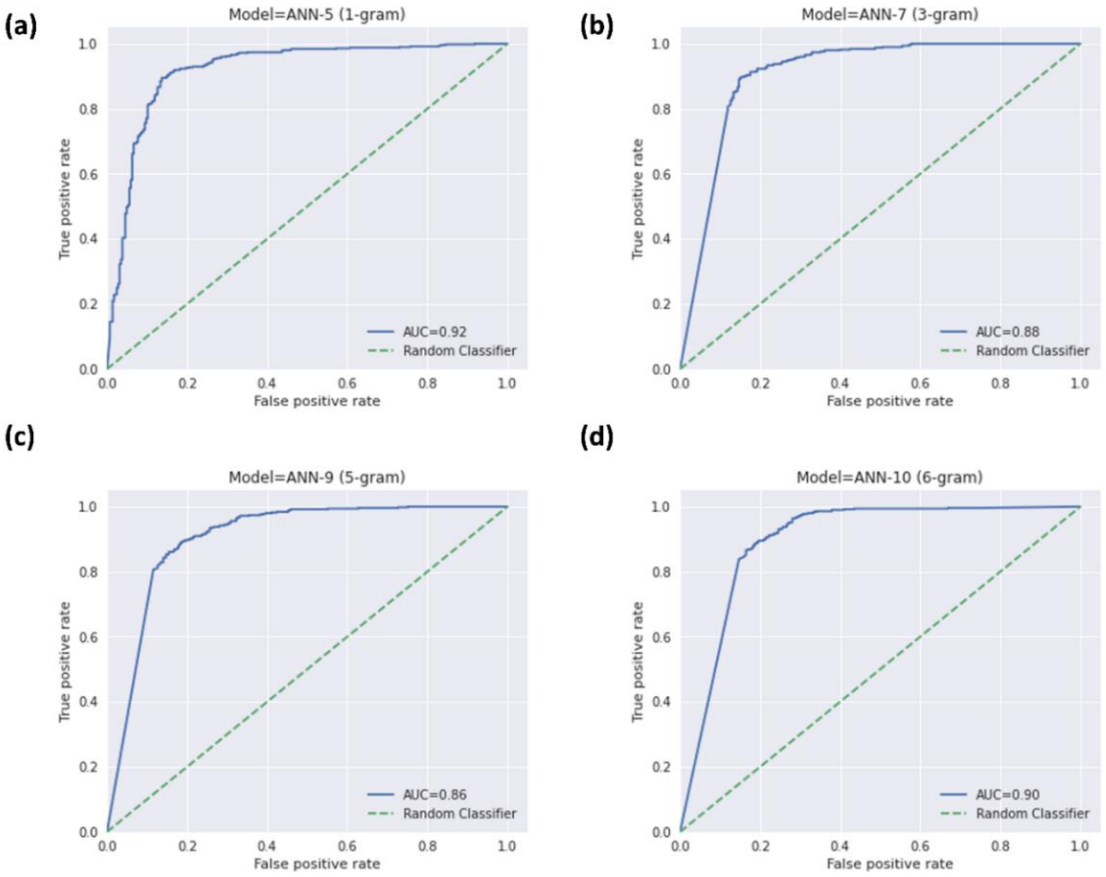

Figure 5.6 ROC curve of models on the test set. (a) ANN-5 (b) ANN-7 (c) ANN-9 (d) ANN-10

Fast model overtraining can be observed in the developed models, i.e. a difference in the model's performance on training and test set. This can be due to a large and underrepresented dataset and complex model. The implications of fast model overtraining can be significant such as poor generalization, increased sensitivity to noise and reduced model interpretability.

The performance of our best classification model was compared with the TEMPO model of Ghosh *et al.* [98]. The comparison in terms of accuracy has been summarized in Table 5.6**.** Our model was also compared with the BBB-score model developed by Gupta *et al.* [99]. Table 5.7 summarizes the accuracy score of our ANN-10 model and BBB-score

model. Shaker *et al.* developed a LightBBB model using the LightGBM algorithm to predict BBB permeability, having an accuracy of 89% [100].

Table 5.6 Accuracy of the TEMPO model and our best ANN-10 model

Model	**Accuracy on CNS test set**	**Accuracy on Non-CNS test set**
TEMPO	72.4%	71.5%
ANN-10	90.2%	87.6%

Table 5.7 Accuracy of BBB-score model and our best ANN-10 model

Model	**Accuracy on CNS test set**	**Accuracy on Non-CNS test set**
BBB-score	80%	72%
ANN-10	90%	88%

An online web server has been developed and is publicly available for predicting the BBB permeability of desired compounds. The user needs to enter the SMILES of the compound, and the server returns whether the query molecule is BBB permeable or not.

5.4 Conclusion

In the presented work, we have proposed an NLP-based deep-learning model built using ANN and LSTM to predict the BBB permeability of the drug. The B3DB dataset was used to build the model. Several NLP based feature extraction techniques were employed, and the models which were built with N-gram tokenization performed better than other models. The ANN models performed better than the bidirectional LSTM model. The accuracy, precision, recall, F1, specificity and AUC of ROC score of the best model (ANN-10) was found to be 0.89, 0.91, 0.91, 0.85 and 0.90, respectively. The data and code can be accessed from https://github.com/ravisingh15/BBB-Deeplearning. This tool can be helpful in early drug-discovery research to identify molecules which can cross the BBB.