Article

Discriminating Ion Channels from Non-Ion Channel Membrane Proteins Using Machine Learning and Deep Learning Classifiers with Protein Language Model Representations

Hamed Ghazikhani [1*] and Gregory Butler [2]

[1] Department of Computer Science and Software Engineering, Concordia University; hamed.ghazikhani@concordia.ca
[2] Centre for Structural and Functional Genomics, Concordia University; gregory.butler@concordia.ca
* Correspondence: hamed.ghazikhani@concordia.ca

Abstract: Ion channels are integral membrane proteins that facilitate the movement of ions across cell membranes, playing a key role in a range of biological processes. The high cost and time required for wet lab experiments to characterize ion channels has spurred the development of computational methods for this purpose. In our previous work, we demonstrated the effectiveness of protein language models for ion channel prediction, using a logistic regression classifier to distinguish ion channels from non-ion channels (TooT-BERT-C) and transporters from non-transporters (TooT-BERT-T). In this study, we build upon this approach by using a combination of classical machine learning classifiers and a Convolutional Neural Network (CNN) with fine-tuned representations from ProtBERT, ProtBERT-BFD, and MembraneBERT to discriminate ion channels from non-ion channels. The results of our experiments demonstrate that TooT-BERT-CNN-C, a combination of the representations from ProtBERT-BFD and a CNN, outperforms existing state-of-the-art methods for predicting ion channels, with a Matthews Correlation Coefficient (MCC) of 0.86 and an accuracy of 98.35% on an independent test set.

Keywords: Ion channels; Membrane proteins; Transmembrane proteins; Drug discovery; Protein language models; Convolutional Neural Network

1. Introduction

Ion channels are integral membrane proteins that play a vital role in the movement of ions across cell membranes, enabling cells to maintain their electrical properties and regulate their functions [1,2]. These proteins are involved in a range of biological processes, including muscle contraction, nerve impulse transmission, and cell signaling [3]. As such, ion channels have garnered significant attention in the field of membrane protein research, with extensive studies focused on their structure, function, and regulation [4,5]. In the realm of drug discovery, ion channels represent promising targets for therapeutic intervention, as their modulation can result in changes in cellular behavior [6,7]. However, the high cost and time required for wet lab experiments to characterize ion channels has spurred the development of computational methods for this purpose [8,9]. These methods can greatly accelerate the drug discovery process by providing efficient and cost-effective predictions of ion channel presence and function.

There has been a significant amount of research on predicting ion channel proteins in the past, with an emphasis on developing computational methods that can accurately differentiate ion channels from non-ion channels [8–14]. These methods have often utilized traditional machine learning techniques, such as support vector machines (SVM) and random forests (RF), which classify protein sequences based on features derived from their primary, secondary, and tertiary structures. These features can include information about the sequence itself, such as the presence of certain amino acid residues or motifs, as well

 doi:10.20944/preprints202301.0221.v1

as structural features, such as secondary structure elements or solvent accessibility [9,15]. The use of these features for ion channel prediction is thoroughly explained in Menke et al. [9] and Ashrafuzzaman [8]. However, the emergence of deep learning has provided new opportunities for ion channel prediction, with recent studies demonstrating the potential of these techniques to generate rich representations of protein sequences and enhance the performance of ion channel predictors [13,14]. These models are able to learn complex patterns in protein sequences and use this information to improve prediction performance, potentially overcoming limitations of traditional machine learning approaches.

Protein language models [16–20], which utilize deep learning techniques, have gained widespread interest among computational biologists due to their potential for accurately modeling biological phenomena. These models produce comprehensive representations of protein sequences that are useful for various applications in protein analysis, including predicting protein function, protein-protein interactions, and protein structure [16–22]. Unsal et al. [23] review the use of natural language models for protein representation from 2015 to the present.

Deep learning-based protein language models, can be pre-trained on large-scale protein datasets and then fine-tuned for specific downstream tasks [23]. This approach, known as transfer learning [24], allows the models to leverage the knowledge they have acquired during pre-training to more effectively perform the targeted task. By pre-training the models on a diverse and representative dataset, they are able to learn generalized patterns in protein sequences that can be applied to a wide range of tasks [19]. Fine-tuning the models on a task-specific dataset then allows them to further adapt and optimize their performance for the particular task at hand. This approach has been demonstrated to be effective in a range of protein analysis applications, including predicting protein function [25], protein-protein interactions [26], and protein structure [27,28].

In the ProtTrans project [19], Elnaggar et al. evaluated six Transformer-based models on the tasks of secondary structure prediction, subcellular localization, and water solubility. ProtBERT and ProtBERT-BFD, two BERT-based models pre-trained on the UniRef100 database (216 million protein sequences) [29] and the BFD database (2.1 billion protein sequences) [30], respectively, were among the transformer-based models examined. ProtBERT and ProtBERT-BFD each consist of 30 layers of 16 attention head transformer encoders with a total of 420 million parameters, and are capable of generating 1024-dimensional vectors for each amino acid in a protein sequence. MembraneBERT [31] is a protein language model that was developed by fine-tuning ProtBERT-BFD on a dataset of membrane proteins from the TooT-M project [32].

In our previous work, we introduced TooT-BERT-T [33] and TooT-BERT-C [34], a method for discriminating transport proteins from non-transport proteins and ion channels from non-ion channels, respectively, using a Logistic Regression (LR) classifier with fine-tuned representations from ProtBERT-BFD. This approach outperformed state-of-the-art transporter and ion channel predictors, highlighting the potential of using protein language models for these tasks.

In this study, we further explore the use of protein language models for ion channel prediction by employing a combination of classical machine learning classifiers, including SVM, RF, k-Nearest Neighbors (kNN), Feed-Forward Neural Networks (FFNN), and LR as well as a Convolutional Neural Network (CNN) to distinguish ion channels from non-ion channels. We utilize representations from ProtBERT, ProtBERT-BFD, and MembraneBERT in our classifiers, to exploit the power of these models for this task.

In this study, we make the following contributions:

1. We investigate the use of fine-tuned representations from ProtBERT, ProtBERT-BFD, and MembraneBERT for ion channel prediction, and analyze their potential for this task.
2. We evaluate a range of traditional machine learning classifiers, including SVM, RF, kNN, FFNN, and LR as well as a CNN for ion channel prediction using these representations.

 doi:10.20944/preprints202301.0221.v1

3. We design a new CNN architecture that is capable of fine-tuning the representations from ProtBERT, ProtBERT-BFD, and MembraneBERT and effectively discriminating ion channels from non-ion channels.
4. We propose TooT-BERT-CNN-C, a method for ion channel prediction that utilizes fine-tuned representations from ProtBERT-BFD and a CNN to outperform all previous approaches.

The remainder of this paper is structured as follows: Section 2 outlines the datasets and methods employed in our experiments, including details on the protein language models, traditional machine learning classifiers, and the proposed CNN architecture. Section 3 presents the results of our experiments, including the performance of the various classifiers on the ion channel prediction task, and provides a discussion of the implications of our findings and insights. Finally, Section 4 summarizes our contributions and suggests directions for future work.

2. Materials and Methods

2.1. Dataset

In this study, we utilize the same dataset as previous works, specifically the DeepIon and MFPS_CNN projects [13,14], which was compiled from the UniProt database [35]. To ensure a diverse and representative sample, Taju and Ou [14] used the BLAST algorithm [36] to eliminate protein sequences with more than 20% similarity, resulting in a dataset containing 4915 protein sequences. This dataset includes 301 ion channels, 351 ion transporters, and 4263 membrane proteins, and was divided into training and test sets to evaluate the generalizability of the model. However, ion transporters were not utilized in this study in order to match the DeepIon [14] methodology. In this paper, non-ion channel proteins refer to membrane proteins that do not function as ion channels. The distribution of sequences within the dataset is shown in Table 1.

Table 1. DS-C, the ion channel dataset. This table displays the distribution of sequences in the dataset used in this study, separated into the training and test sets.

Class	Training	Test	Total
Ion channel	241	60	301
Non-ion channel	3,413	850	4,263
Total	3,654	910	4,564

2.2. Protein language models

Transfer learning is a technique that allows a model to use knowledge gained from one task to improve its performance on a related, but different task [37]. This can be particularly useful when there is a limited amount of labeled data available for the target task, as the model can utilize the knowledge gained from the source task to enhance its performance. The process of transfer learning consists of two steps: Pre-training and fine-tuning (see Figure 1).

Pre-training involves training a model on a large, general-purpose dataset, allowing the model to learn general patterns and representations that are applicable to a variety of tasks [38]. Fine-tuning involves further training the pre-trained model on a specific task or dataset, allowing the model to adapt to the characteristics of that task or dataset and often resulting in improved performance [38].

BERT [24], or Bidirectional Encoder Representations from Transformers, is a transformer-based [39] language model that has achieved state-of-the-art performance on a variety of natural language processing tasks. BERT is able to generate contextualized representations of words in a sentence, taking into account the words that come before and after it [40]. This is achieved through the use of self-attention mechanisms [39], which allow the model to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

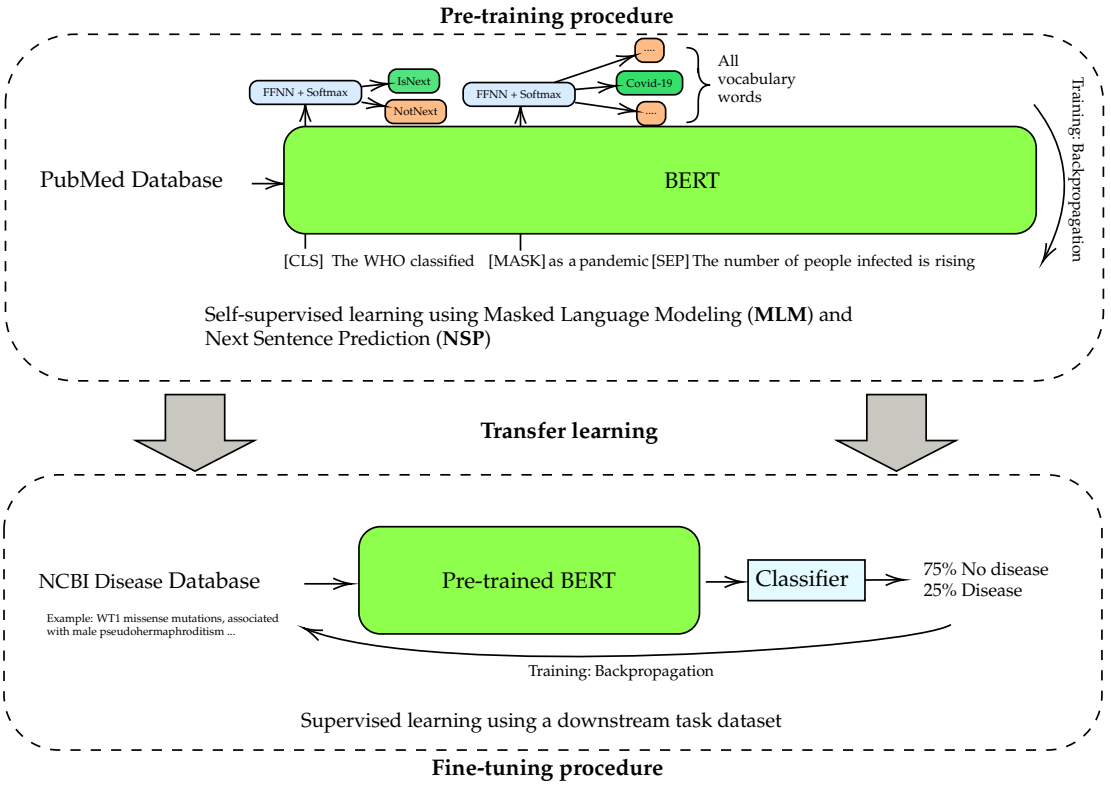

Figure 1. BERT training stages. The BERT model training process is depicted in this figure, with the pre-training phase involving the use of Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) objective to learn contextual representations of words from the PubMed Database. The fine-tuning phase involves adapting the pre-trained BERT model to the binary classification task of predicting whether a text from the NCBI Disease Database describes a disease or does not describe a disease.

attend to different parts of the input sequence and weight their importance for generating the final representation.

BERT (Figure 1) has a multi-layered architecture of self-attention and feed-forward networks in each layer and is trained using two pre-training tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In the MLM task, a percentage of the tokens in the input are randomly masked and the model is required to predict the original value of the masked tokens based on the context provided by the remaining tokens. The NSP task involves predicting whether a given sentence is the next sentence in a sequence or a randomly chosen sentence from the dataset. By performing these pre-training tasks, BERT is able to learn contextual relationships between words in a sentence and between sentences in a document. Once pre-trained, BERT can be fine-tuned for specific classification tasks by adding a classification head on top of the model.

Frozen (feature-based) and fine-tuned representations are two different types of representations that can be extracted from a BERT model. Frozen representations, are derived from the internal layers of a BERT model that has been pre-trained on a large, general-purpose dataset. These representations capture general patterns and structures in the data that are useful for a wide range of tasks. Fine-tuned representations, on the other hand, are derived from a BERT model that has been further trained on a specific task or dataset. These representations are tailored to the specific characteristics of the task or dataset, and can often provide better performance for that task or dataset compared to feature-based representations.

In self-attention, a sequence of n items is transformed into a sequence of n vectors. The i-th vector is computed by applying the self-attention mechanism to the i-th item and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

all other items in the sequence. The self-attention mechanism is defined by three functions: the query function Q, the key function K, and the value function V. The value function V is typically implemented as a linear transformation of the input item, such as a matrix multiplication followed by an optional non-linear activation function. Given an input sequence of items $x_1, x_2, ..., x_n$, the self-attention mechanism produces an output sequence of vectors $y_1, y_2, ..., y_n$ with the following formula:

$$y_i = \sum_{j=1}^{n} a_{i,j} V(x_j) \tag{1}$$

where $a_{i,j}$ is a weight that indicates the importance of item x_j to the computation of vector y_i. These weights are computed by taking the dot product of the query vector $Q(x_i)$ and the key vector $K(x_j)$ for all pairs of items i and j, and then normalizing these dot products using the softmax function:

$$a_{i,j} = \frac{\exp(Q(x_i) \cdot K(x_j))}{\sum_{k=1}^{n} \exp(Q(x_i) \cdot K(x_k))} \tag{2}$$

Protein language models, a subclass of natural language models, have gained significant attention in recent years due to their ability to generate highly expressive and informative representations of protein sequences. These representations have been widely applied to various tasks in computational biology, including predicting protein function, structure, and interactions [17,19,21,23,41,42].

ProtBERT [19] is a protein language model that was pre-trained on the UniRef100 database [43], which contains over 216 million protein sequences. It is based on the BERT model architecture [24] and uses the MLM pre-training task. ProtBERT-BFD [19] is a variant of ProtBERT that was pre-trained on the BFD database [30], which consists of over 2.1 billion protein sequences. MembraneBERT [31] is a variant of ProtBERT-BFD that has been fine-tuned on a dataset of membrane proteins [32], enabling it to specifically focus on the characteristics of these proteins.

2.3. Machine learning classifiers

In this study, we use a range of common classical machine learning classifiers in the field of protein-informatics [23,44] including logistic regression, support vector machine, random forest, k-nearest neighbor, and feed-forward neural network from the scikit-learn library [45] as well as a convolutional deep neural network implemented in Pytorch [46] to distinguish ion channels from non-ion channels using the representations generated by ProtBERT, ProtBERT-BFD, and MembraneBERT. To optimize the performance of each classifier, we perform a grid search over a set of hyperparameters on the training set (See Table 2). The optimized classifier is then applied to the test set to evaluate its performance. In the following sections, we provide a detailed description of these methods.

2.3.1. Logistic Regression (LR)

LR [47] is a linear model that models the probability that an input sample belongs to a particular class, based on the linear combination of the input features and the model weights. Given a set of input features $X = x_1, x_2, ..., x_n$ and their corresponding labels $Y = y_1, y_2, ..., y_n$, where $y_i \in 0, 1$, the LR model learns the weights $W = w_1, w_2, ..., w_n$ that minimize the cost function, which is defined as the negative log-likelihood of the predicted labels of the input data:

$$J(W) = -\frac{1}{m} \sum_{i=1}^{m} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right] \tag{3}$$

where m is the number of samples and $\hat{y}_i$ is the predicted label of the i-th sample, which is computed as the sigmoid function of the linear combination of the input features and the model weights:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

$$\hat{y}_i = \sigma(W^T \cdot x_i) = \frac{1}{1 + e^{-W^T \cdot x_i}} \tag{4}$$

The sigmoid function maps the output of the linear combination to the range $[0, 1]$, which represents the probability that the i-th sample belongs to the positive class. The weights of the LR model are learned using gradient descent, which iteratively updates the weights to minimize the cost function by computing the partial derivative of the cost function with respect to each weight.

2.3.2. Support Vector Machine (SVM)

SVM [48] is a supervised learning algorithm that works by finding a decision boundary that maximally separates the training data into their corresponding classes. The decision boundary is determined by the support vectors, which are the training data points that lie on the margin of the classes. The SVM model then predicts the label of the input data based on the location of the input data relative to the decision boundary.

Given a set of input features $X = x_1, x_2, ..., x_n$ and their corresponding labels $Y = y_1, y_2, ..., y_n$, where $y_i \in 0, 1$, the SVM model learns a decision boundary by solving the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^t w + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(w\phi(x_i + b)) + \xi_i - 1 \geq 0, \quad i = 1, ..., n \\ & \xi \geq 0, \quad i = 1, ..., n \end{aligned} \tag{5}$$

where w and b are the parameters of the decision boundary, ξ is the vector of slack variables, C is the penalty parameter, and $\phi(x_i)$ is the feature mapping of the input data. The decision boundary is given by the equation $w^T \phi(x) + b = 0$. The SVM model then predicts the label of the input data as:

$$\hat{y} = \text{sign}(w^T \phi(x) + b) \tag{6}$$

where $\text{sign}(z) = 1$ if $z \geq 0$ and $\text{sign}(z) = -1$ if $z < 0$.

The SVM model is effective, as it can learn complex decision boundaries that can accurately separate the training data into their corresponding classes. It is also efficient, as it only uses a small subset of the training data as support vectors to determine the decision boundary. However, the SVM model is sensitive to the choice of the kernel function, which is used to map the input data into a higher-dimensional space [49]. The kernel function allows the SVM model to learn non-linear decision boundaries by applying a non-linear transformation to the input data. This is known as the kernel trick, which allows the SVM model to learn complex decision boundaries without explicitly computing the feature mapping of the input data. This can improve the performance of the SVM model on non-linearly separable data and make it more efficient, as it avoids the computationally expensive operation of mapping the input data into a higher-dimensional space [50].

2.3.3. Random Forest (RF)

RF [51] is an ensemble model that consists of a collection of decision trees, where each tree is trained on a random subset of the input data and makes predictions based on the feature values of the input data. This allows the decision trees to make independent predictions and avoid overfitting the input data. The RF model uses the majority voting strategy to combine the predictions of the individual decision trees, which reduces the variance of the predictions and improves the performance of the model. Given a set of input features $X = x_1, x_2, ..., x_n$ and their corresponding labels $Y = y_1, y_2, ..., y_n$, where $y_i \in 0, 1$, the RF model learns a set of decision trees to predict the labels of the input data as:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

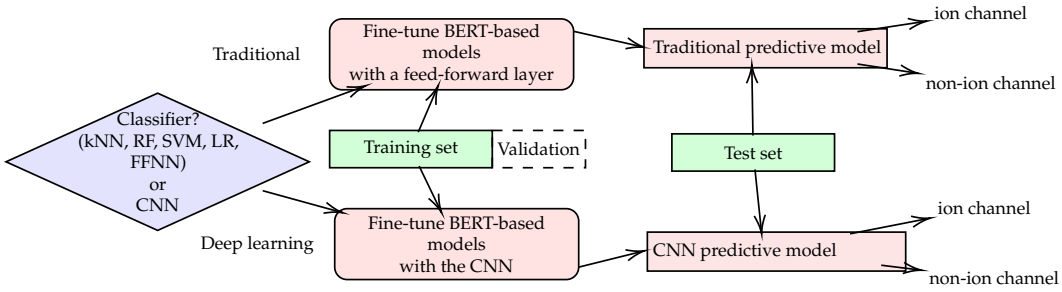

Figure 2. Proposed method for protein classification. The figure shows the proposed method for classifying ion channel proteins from non-ion channel membrane proteins using a variety of classifiers. The training and validation sets are used to fine-tune BERT-based models (ProtBERT, ProtBERT-BFD, and MembraneBERT) depending on the classifier chosen, which can be a traditional classifier or a deep learning-based classifier (CNN). These models are then evaluated on the test set to determine their performance in classifying the proteins.

$$\hat{y}_i = \frac{1}{T} \sum_{j=1}^{T} \hat{y}_{i,j} \tag{7}$$

where $\hat{y}_i$ is the final prediction for sample i made by the random forest, T is the number of decision trees in the random forest, and $\hat{y}_{i,j}$ is the prediction of the j-th decision tree for sample i.

2.3.4. k-Nearest Neighbor (kNN)

kNN [52] is a non-parametric supervised learning algorithm that works by calculating the distance between the input data and a set of labeled training data. The distance is calculated using a distance metric, such as the Euclidean distance, which measures the difference between the feature values of the input data and the training data. The kNN model then predicts the label of the input data by finding the k training data points that are closest to the input data and averaging their labels.

Given an input data point x with features $X = x_1, x_2, ..., x_n$ and a set of training data $T = (t_1, y_1), (t_2, y_2), ..., (t_m, y_m)$, where t_i is the feature vector of the training data and y_i is its corresponding label, the kNN model predicts the label of the input data as follows:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{8}$$

where y_i is the label of the i-th nearest neighbor to the input data and k is the number of neighbors.

The kNN model is simple and easy to implement, as it does not require training to learn the relationship between the input features and the output labels. However, the model does require training to learn the distance metric that is used to determine the similarity between the input data points and the reference points. It is also effective, as it can learn complex non-linear relationships between the input data and the output labels. However, it is computationally expensive, as it requires calculating the distances between the input data and the entire training set for each prediction. In addition, the kNN model is sensitive to the choice of the distance metric and the number of neighbors, which can affect the performance of the model.

2.3.5. Feed-Forward Neural Network (FFNN)

FFNN [53,54] consists of multiple layers of neurons, where each layer transforms the input data into a higher-dimensional space using a non-linear activation function. The final output of the FFNN is obtained by applying a linear activation function to the output of the last layer. Given a set of input features $X = x_1, x_2, ..., x_n$ and their corresponding labels

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

Table 2. Optimal hyperparameters of classifiers. The table lists the optimal hyperparameters of the classifiers determined through grid search.

Classifier	Hyperparameter	Value
KNN	algorithm	auto
	leaf size	10
	number of neighbors	5
RF	min samples leaf	1
	min samples split	2
	number of estimators	100
SVM	C	10
	gamma	0.1
	kernel	rbf
LR	C	1
	max iteration	100
	solver	lbfgs
FFNN	alpha	0.0001
	hidden layer sizes	(256, 32)
	learning rate	invscaling
CNN	epochs	10
	learning rate	$5e^{-5}$
	batch size	4

$Y = y_1, y_2, ..., y_n$, where $y_i \in 0, 1$, the FFNN model learns a set of weights and biases to predict the labels of the input data as follows:

$$\hat{Y} = f_{\text{out}}\left(\sum_{i=1}^{n} W_i f_{\text{act}}(W_i X + b_i) + b_{\text{out}}\right) \tag{9}$$

where W_i and b_i are the weights and biases of the i-th layer, f_{act} is the non-linear activation function, and f_{out} is the linear activation function applied to the output of the last layer.

The FFNN model learns the weights and biases of each layer using a gradient-based optimization algorithm, such as *Adam* [55]. The optimization algorithm minimizes the loss function, which measures the difference between the predicted labels $\hat{Y}$ and the true labels Y. The loss function is used to compute the gradients of the weights and biases with respect to the loss, which are then used to update the weights and biases of the model.

The FFNN model is known for its ability to learn complex non-linear relationships between the input features and the labels. It is also flexible, as it can be easily adapted to different datasets by changing the number of layers and the number of neurons in each layer. However, the FFNN model is prone to overfitting and requires a large amount of training data to learn the weights and biases accurately[53].

2.3.6. Convolutional Neural Network (CNN)

A CNN [56,57] is a type of neural network that that has been successfully applied to various tasks, including image classification and object detection [57,58] and it has also been used for protein analysis [13,59]. It consists of multiple layers of neurons, where each layer applies a convolution operation to the input data to extract local features. The convolution operation is applied using a set of filters, where each filter is a small matrix of weights that is learned by the CNN model. The final output of the CNN is obtained by applying a non-linear activation function to the output of the last layer.

The convolution operation is a mathematical operation that involves combining an input sequence with a set of filters using element-wise multiplication, sliding the filters across the input, and producing an output through element-wise summation. In the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

context of a CNN, this operation is used to extract features from the input sequence. The convolution operation can be defined mathematically in two ways:

1. Using the asterisk (*) as the convolution operator:

$$O = X * F \tag{10}$$

where O is the output of the convolution operation, X is the input sequence, and F is the set of filters. The output of the convolution operation is a feature map with dimensions $(l - f_l + 1) \times o$, where l is the length of the input sequence, f_l is the length of the filters, and o is the number of output channels.

2. Using a double sum:

$$O_i = f_{\text{act}}\left(\sum_{j=1}^{c}\sum_{k=1}^{f_l} F_{i,j,k} \cdot X_{i+k-1,j} + b_i\right) \tag{11}$$

where O_i is the output of the convolution operation at position i, f_{act} is the non-linear activation function, and b_i is the bias of the filter.

The CNN model, similar to FFNN learns the filters using a gradient-based optimization algorithm. Then the loss function is used to compute the gradients of the filters with respect to the loss, which are then used to update the filters of the model.

In this work, we used a CNN to classify ion channels and non-ion channels, as well as to fine-tune ProtBERT, ProtBERT-BFD, and MembraneBERT concurrently during training. The proposed CNN consists of multiple convolutional layers with various filter sizes and numbers, followed by a dropout layer, and three fully connected layers. Dropout [60] is a regularization technique in deep learning that helps prevent overfitting by randomly dropping out a certain percentage of neurons in the network during training. The first and second fully connected layers contain 128 and 64 neurons, respectively, while the final layer has 2 neurons to predict the class probabilities. The convolutional layers extract features from the input data and the fully connected layers classify these features into the desired classes. The input data is processed by the convolutional layers during the forward pass of the model, followed by the dropout layer to reduce overfitting, and finally passed through the fully connected layers to generate the class probabilities.

The proposed CNN architecture for classifying ion channels and non-ion channels, as well as for fine-tuning ProtBERT, ProtBERT-BFD, and MembraneBERT concurrently during training, is well-suited for this task for several reasons. First, the use of convolutional layers allows the model to extract features from the input data and learn spatial relationships between the input elements, which is particularly useful for processing sequential data such as protein sequences [57]. Additionally, the use of multiple convolutional layers with various filter sizes and numbers allows the model to learn a hierarchy of features, with each layer learning more complex features based on the simpler ones learned by the previous layers [58]. This can help the model capture more nuanced patterns in the input data and improve its ability to classify the data. The inclusion of a dropout [60] layer also helps prevent overfitting, which can improve the model's generalization to unseen data and its performance on the test set. Finally, the use of fully connected layers allows the model to classify the features extracted by the convolutional layers into the desired classes, with the three fully connected layers learning more complex relationships between the features and the classes.

2.4. Proposed Method

Our proposed method for protein classification is illustrated in Figure 2. This method involves using a variety of classifiers to classify ion channel proteins from non-ion channel membrane proteins. The classifiers include both traditional machine learning algorithms (kNN, RF, SVM, LR, FFNN) and deep learning-based approach (CNN).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

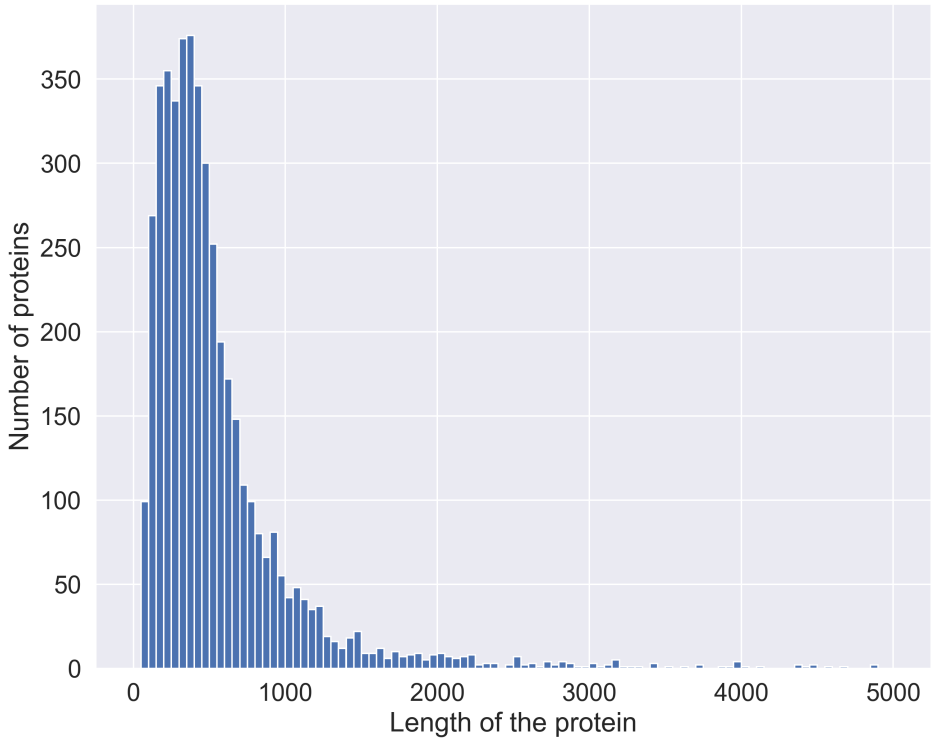

Figure 3. Distribution of protein lengths in the dataset. The distribution of protein lengths in the dataset is depicted in this histogram, with the x-axis representing the length of the proteins and the y-axis indicating the number of proteins.

For the classical machine learning classifiers, the BERT-based models (ProtBERT, ProtBERT-BFD, and MembraneBERT) are first fine-tuned using a one-layer feed-forward classifier. The representations learned by the fine-tuned BERT models are then extracted and used as input for the classical classifiers.

For the deep learning-based classifier, the BERT-based models are fine-tuned concurrently with the proposed CNN during training. This allows the BERT models to learn task-specific representations that are better suited for the protein classification task. The representations learned by the fine-tuned BERT models are then used as input for the CNN, which is trained to classify the proteins based on these representations.

During the training phase, the classifiers are trained on the training and validation sets, which are used to fine-tune the BERT-based models (ProtBERT, ProtBERT-BFD, and MembraneBERT) depending on the classifier chosen. The validation set is a subset of the training data that is used to tune the hyperparameters of a model, assess the model's generalization performance, and prevent overfitting. In our work, the validation set was randomly chosen to be 10% of the training set.

Once the classifiers and BERT-based models have been trained, they are evaluated on the test set to determine their performance in classifying the proteins. The test set consists of data that the model has not seen during training, and its performance on this set provides an estimate of the model's generalization ability to classify new, unseen data.

2.5. Grid search

In this work, we used the grid search method to find the optimal hyperparameter values for each classifier (see Table 2). The grid search method involves training and evaluating a model on a grid of hyperparameter values, using cross-validation (CV) to estimate the model's performance, in order to find the combination of hyperparameter values that results in the best performance on the data. By applying the grid search method,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

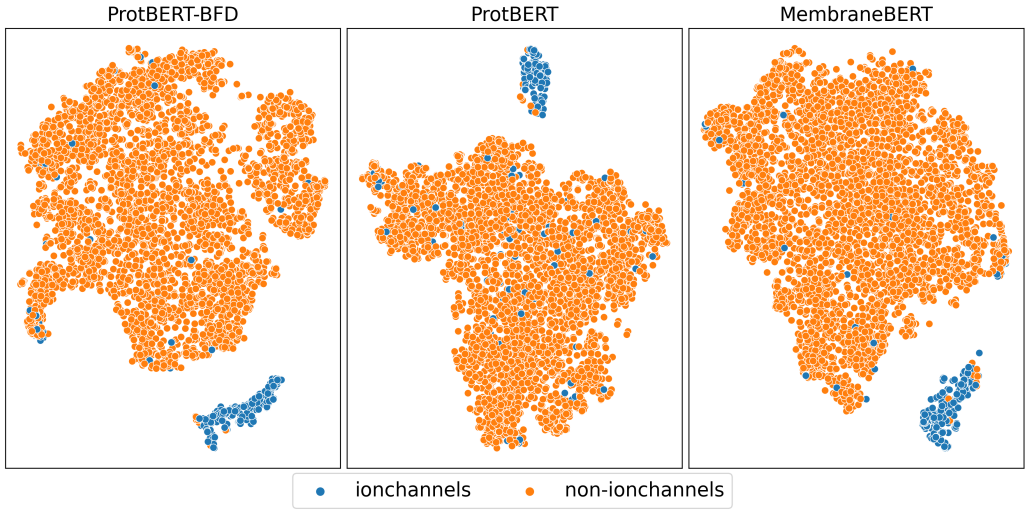

Figure 4. t-SNE plot of BERT representations. A t-SNE plot is shown in this figure, illustrating the two-dimensional representations of the sequences obtained from ProtBERT, ProtBERT-BFD, and MembraneBERT for the ion channel and non-ion channel classes in the dataset. The ion channel sequences are plotted in blue, while the non-ion channel sequences are plotted in orange.

we were able to identify the classifier and hyperparameter combination that achieved the best performance on the data.

2.6. Evaluation methods

In our study, we used 5-fold CV to evaluate the performance of all of the classifiers. CV is a technique that involves dividing the dataset into a number of folds, training the model on some of the folds, and evaluating the model on the remaining folds. This process is repeated a number of times, with different combinations of training and evaluation folds, in order to obtain a more robust estimate of the model's performance. The optimal hyperparameters for each classifier were determined based on the results of the CV, and are shown in Table 2.

2.6.1. McNemar's test

We utilize McNemar's test to evaluate the differences in performance between our proposed method, TooT-BERT-CNN-C, and the state-of-the-art model TooT-BERT-C. McNemar's test [61] is a statistical test used to compare the performance of two different classifiers on a binary classification task. It is typically used when the same set of samples has been classified by both classifiers, and the goal is to determine whether the performance of one classifier is significantly better than the other.

The test is based on the contingency table, which is a 2x2 table that compares the outcomes of two classification methods. The formula for McNemar's test is:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{12}$$

where b represents the number of cases where the first model made an incorrect prediction, while the second model made a correct prediction. c represents the number of cases where the first model made a correct prediction, while the second model made an incorrect prediction.

To interpret the results of McNemar's test, a p-value is calculated based on the chi-square statistic. If the p-value is less than a predetermined threshold (usually 0.05), then it is considered statistically significant and the null hypothesis (that there is no difference in performance between the two classifiers) is rejected. This indicates that there is a statistically significant difference in the performance of the two classifiers.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

Table 3. Performance comparison of different classifiers and representations. The table presents a comparison of the performance of different classical and deep learning classifiers and representations, as generated from ProtBERT, ProtBERT-BFD, and MembraneBERT on CV and independent test sets. The results are evaluated using various metrics, with the largest value in each column on independent test results indicated in boldface for comparison between different classifiers. The second best result in each column is indicated with an underline for further analysis and comparison.

PLM	Method	Acc(%)		Sen(%)		Spc(%)		MCC	
		CV	Ind.	CV	Ind.	CV	Ind.	CV	Ind.
ProtBERT	kNN	97.80 ± 0.23	97.69	71.61 ± 3.75	70.00	99.65 ± 0.25	99.53	0.8086 ± 0.0213	0.7972
	RF	97.30 ± 0.22	97.58	60.69 ± 3.14	63.33	99.88 ± 0.15	**100.00**	0.7557 ± 0.0234	0.7749
	SVM	95.22 ± 0.27	98.24	37.63 ± 4.23	75.00	99.29 ± 0.23	**100.00**	0.4469 ± 0.0358	0.8483
	LR	97.57 ± 0.25	97.80	67.13 ± 3.77	68.33	99.72 ± 0.25	**100.00**	0.7852 ± 0.0234	0.8068
	FFNN	97.13 ± 0.54	98.02	67.04 ± 4.23	73.33	98.69 ± 1.42	**100.00**	0.7126 ± 0.0299	0.7848
	CNN	99.09 ± 0.82	97.80	88.90 ± 3.82	66.66	99.82 ± 0.21	**100.00**	0.9235 ± 0.0728	0.8070
Average ProtBERT			97.86		69.44		99.92		0.8032
ProtBERT-BFD	kNN	98.97 ± 0.31	97.47	88.19 ± 5.12	75.00	99.73 ± 0.10	98.71	0.9137 ± 0.0275	0.7848
	RF	99.03 ± 0.45	97.47	87.97 ± 5.85	**76.67**	99.80 ± 0.12	98.82	0.9187 ± 0.0390	0.7767
	SVM	98.39 ± 0.40	97.69	79.87 ± 5.49	75.00	99.69 ± 0.13	**100.00**	0.8450 ± 0.0353	0.8016
	LR	98.96 ± 0.43	98.24	86.71 ± 5.55	**76.67**	99.82 ± 0.11	99.76	0.9123 ± 0.0375	0.8486
	FFNN	98.34 ± 0.92	97.03	82.08 ± 7.42	**76.67**	99.38 ± 0.78	**100.00**	0.8557 ± 0.0584	0.8287
	CNN	99.39 ± 0.20	**98.35**	93.38 ± 2.96	75.00	99.82 ± 0.14	**100.00**	0.9506 ± 0.0167	**0.8584**
Average ProtBERT-BFD			97.71		75.84		99.55		0.8165
MembraneBERT	kNN	99.59 ± 0.34	96.92	96.48 ± 3.03	66.67	99.81 ± 0.18	98.82	0.9665 ± 0.0278	0.7358
	RF	99.59 ± 0.33	97.03	96.32 ± 2.84	68.33	99.82 ± 0.17	98.94	0.9670 ± 0.0274	0.7495
	SVM	99.29 ± 0.32	97.03	91.86 ± 3.13	68.33	99.81 ± 0.15	**100.00**	0.9397 ± 0.0264	0.7383
	LR	99.50 ± 0.33	97.14	95.53 ± 2.88	66.67	99.78 ± 0.17	99.29	0.9590 ± 0.0275	0.7472
	FFNN	99.18 ± 0.44	97.25	91.07 ± 4.84	68.33	99.77 ± 0.20	**100.00**	0.9159 ± 0.0491	0.7383
	CNN	97.86 ± 1.57	97.91	75.55 ± 4.55	68.33	99.44 ± 1.04	**100.00**	0.8203 ± 0.1267	0.8175
Average MembraneBERT			97.21		67.78		99.51		0.7544

2.6.2. Evaluation metrics

For this paper, we used a variety of performance metrics to evaluate the effectiveness of our approach for predicting ion channels. These metrics included accuracy (Acc), sensitivity (Sen), specificity (Spc), and the Matthew's correlation coefficient (MCC).

The dataset used in this work, consists of 301 ion channels and 4,263 non ion-channels, which is an imbalanced dataset. This means that the number of samples in each class is not equal, and this can affect the performance of machine learning algorithms. In imbalanced datasets, accuracy can be misleading as it does not take into account the relative frequencies of the different classes. This can lead to the model achieving high accuracy by simply predicting the majority class all the time, even if it has poor performance on the minority class. Therefore, it is often recommended to use metrics that consider all classes, such as the MCC, which takes into account true and false positives and negatives [62].

Accuracy is a measure of the overall correct classification rate and is calculated as the number of correct predictions divided by the total number of predictions. It is expressed as a percentage and can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

Sensitivity, also known as the true positive rate, is a measure of the proportion of actual positive cases that are correctly identified as such. It is calculated using the following formula:

$$Sensitivity = \frac{TP}{TP + FN} \tag{14}$$

Specificity, also known as the true negative rate, is a measure of the proportion of actual negative cases that are correctly identified as such. It is calculated using the following formula:

$$Specificity = \frac{TN}{TN + FP} \tag{15}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

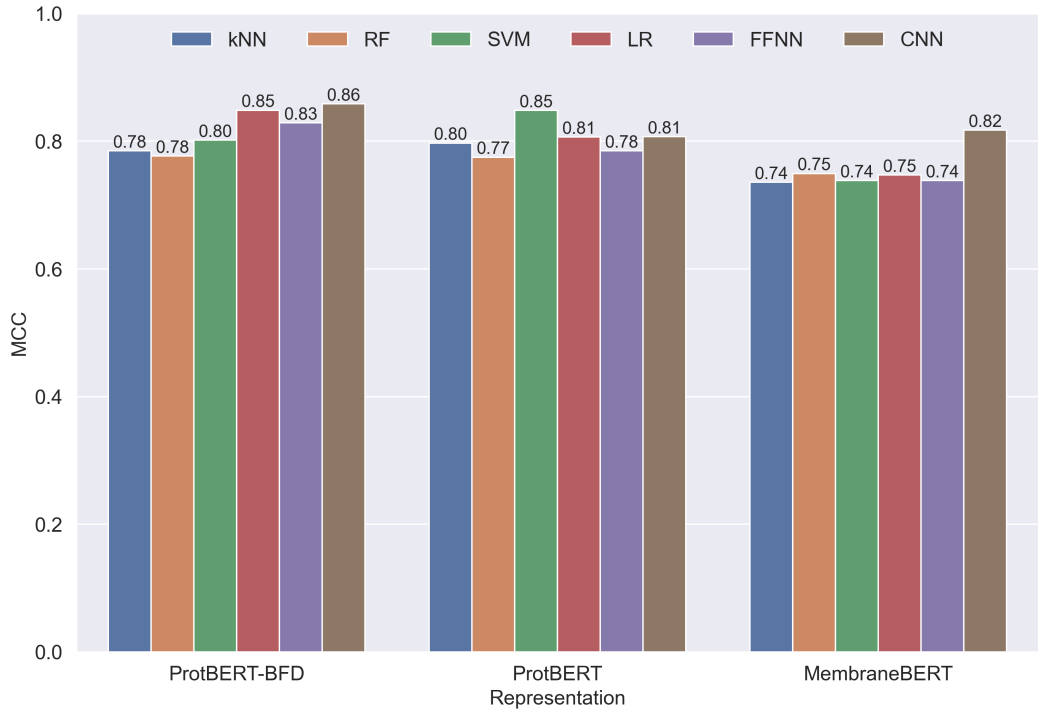

Figure 5. Performance comparison of different classifiers. The performance of different classical and deep learning classifiers and representations generated from ProtBERT, ProtBERT-BFD and MembraneBERT is compared on independent test sets using the MCC metric in this figure.

MCC is a measure of the overall accuracy of a binary classifier, taking into account both the true and false positive and negative rates. It can range from -1 (perfectly incorrect) to 1 (perfectly correct) and is calculated using the following formula:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{16}$$

where TP (True Positive) is a case where the classifier correctly predicts the positive class, TN (True Negative) is a case where the classifier correctly predicts the negative class, FP (False Positive) is a case where the classifier incorrectly predicts the positive class, and FN (False Negative) is a case where the classifier incorrectly predicts the negative class.

3. Results and Discussion

3.1. Protein sequence evaluation

One issue with using protein language models is the need to consider fixed sizes for protein sequences. This can be a concern as it may result in the loss of important structural or functional information. In our method, we truncated the protein sequences to a size of 1024 during the fine-tuning of ProtBERT, ProtBERT-BFD, and MembraneBERT due to memory and resource limitations. An analysis, shown in Figure 3, was conducted to assess the impact of this truncation on the dataset. The histogram in Figure 3 illustrates the distribution of protein sequence lengths in the dataset. As can be seen, the majority of protein sequences in the dataset are shorter than the truncation length, indicating that we are not likely to be losing a significant amount of information through this truncation during the fine-tuning of the BERT models.

In order to visualize the representations from ProtBERT, ProtBERT-BFD, and MembraneBERT for ion channels and non-ion channels, we utilize t-SNE (t-Distributed Stochastic Neighbor Embedding) [63]. t-SNE is a dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional space while preserving the relationships

Table 4. Performance comparison of different classifiers and representations. This table presents a comparison of the performance of different classical and deep learning classifiers, grouped by method, and representations generated from ProtBERT, ProtBERT-BFD, and MembraneBERT on cross-validation (CV) and independent test sets. The results are evaluated using various metrics, with the largest value in each column on independent test results indicated in boldface for comparison between different classifiers. The second-best result in each column is indicated with an underline for further analysis and comparison.

Method	Representation	Acc(%)		Sen(%)		Spc(%)		MCC	
		CV	Ind.	CV	Ind.	CV	Ind.	CV	Ind.
kNN	ProtBERT-BFD	98.97 ± 0.31	97.47	88.19 ± 5.12	75.00	99.73 ± 0.10	98.71	0.9137 ± 0.0275	0.7848
	ProtBERT	97.80 ± 0.23	97.69	71.61 ± 3.75	70.00	99.65 ± 0.25	99.53	0.8086 ± 0.0213	0.7972
	MembraneBERT	99.59 ± 0.34	96.92	96.48 ± 3.03	66.67	99.81 ± 0.18	98.82	0.9665 ± 0.0278	0.7358
Average kNN			97.36		70.56		99.02		0.7726
RF	ProtBERT-BFD	99.03 ± 0.45	97.47	87.97 ± 5.85	**76.67**	99.80 ± 0.12	98.82	0.9187 ± 0.0390	0.7767
	ProtBERT	97.30 ± 0.22	97.58	60.69 ± 3.14	63.33	99.88 ± 0.15	**100.00**	0.7557 ± 0.0234	0.7749
	MembraneBERT	99.59 ± 0.33	97.03	96.32 ± 2.84	68.33	99.82 ± 0.17	98.94	0.9670 ± 0.0274	0.7495
Average RF			97.36		69.44		99.25		0.7670
SVM	ProtBERT-BFD	98.39 ± 0.40	97.69	79.87 ± 5.49	75.00	99.69 ± 0.13	**100.00**	0.8450 ± 0.0353	0.8016
	ProtBERT	95.22 ± 0.27	98.24	37.63 ± 4.23	75.00	99.29 ± 0.23	**100.00**	0.4469 ± 0.0358	0.8483
	MembraneBERT	99.29 ± 0.32	97.03	91.86 ± 3.13	68.33	99.81 ± 0.15	**100.00**	0.9397 ± 0.0264	0.7383
Average SVM			97.65		72.78		100.00		0.7961
LR	ProtBERT-BFD	98.96 ± 0.43	98.24	86.71 ± 5.55	**76.67**	99.82 ± 0.11	99.76	0.9123 ± 0.0375	0.8486
	ProtBERT	97.57 ± 0.25	97.80	67.13 ± 3.77	68.33	99.72 ± 0.25	**100.00**	0.7852 ± 0.0234	0.8068
	MembraneBERT	99.50 ± 0.33	97.14	95.53 ± 2.88	66.67	99.78 ± 0.17	99.29	0.9590 ± 0.0275	0.7472
Average LR			97.73		70.56		99.68		0.8009
FFNN	ProtBERT-BFD	98.34 ± 0.92	97.03	82.08 ± 7.42	**76.67**	99.38 ± 0.78	**100.00**	0.8557 ± 0.0584	0.8287
	ProtBERT	97.13 ± 0.54	98.02	67.04 ± 4.23	73.33	98.69 ± 1.42	**100.00**	0.7126 ± 0.0299	0.7848
	MembraneBERT	99.18 ± 0.44	97.25	91.07 ± 4.84	68.33	99.77 ± 0.20	**100.00**	0.9159 ± 0.0491	0.7383
Average FFNN			97.43		72.78		100.00		0.7839
CNN	ProtBERT-BFD	99.39 ± 0.20	**98.35**	93.38 ± 2.96	75.00	99.82 ± 0.14	**100.00**	0.9506 ± 0.0167	**0.8584**
	ProtBERT	99.09 ± 0.82	97.80	88.90 ± 3.82	66.66	99.82 ± 0.21	**100.00**	0.9235 ± 0.0728	0.8070
	MembraneBERT	97.86 ± 1.57	97.91	75.55 ± 4.55	68.33	99.44 ± 1.04	**100.00**	0.8203 ± 0.1267	0.8175
Average CNN			98.02		70.00		100.00		0.8276

between the data points. It is useful for visualizing complex, non-linear relationships in data and is commonly used in fields such as machine learning and data visualization.

The t-SNE plot in Figure 4 demonstrates the ability of ProtBERT, ProtBERT-BFD, and MembraneBERT to capture important features of protein sequences that distinguish ion channels from non-ion channels. The plot shows that the representations produced by these models can effectively separate ion channels (depicted in blue) from non-ion channels (depicted in orange) in two dimensions, indicating that the models are able to capture fundamental differences between the two classes. These differences could include sequence composition or structural properties that are indicative of ion channel proteins.

3.2. Performance of machine learning classifiers

Table 3 presents a comparison of the performance of various classifiers using different representations from ProtBERT, ProtBERT-BFD, and MembraneBERT on the task of ion channel prediction. The classifiers included in the table are kNN, RF, SVM, LR, and a CNN. The table presents results for several evaluation metrics including accuracy, sensitivity, specificity, and MCC. The results are provided for both CV and independent test sets. The table shows the mean and standard deviation of the metric over the folds of CV, as well as the value on the independent set.

Overall, Table 3 suggests that the classifiers achieve high performance on the ion channel prediction task, with most of the mean values for the evaluation metrics being above 90%. The best results are typically achieved using the ProtBERT-BFD and MembraneBERT representations, which tend to outperform the ProtBERT representation in most cases. The SVM classifier generally performs worse than the other classifiers, with lower mean values for most of the evaluation metrics.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

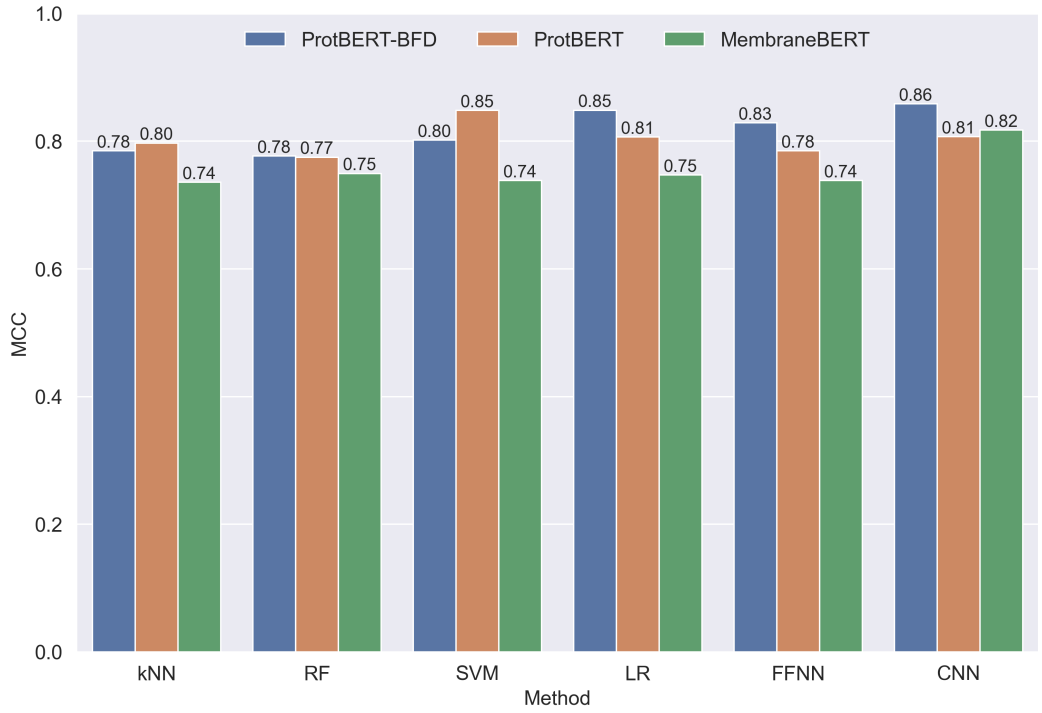

Figure 6. Comparison of performance of different classifiers. This figure shows the results of the performance comparison of different classical and deep learning classifiers, grouped by method, on independent test sets using the MCC metric. The results were generated from ProtBERT, ProtBERT-BFD, and MembraneBERT representations.

Table 3 demonstrates overfitting, as the values for the evaluation metrics on the independent set are consistently lower than the corresponding values on the cross-validation set. Also, this could potentially be due to differences in the characteristics of the cross-validation set and the independent test set, such as differences in the distribution of protein lengths or the prevalence of certain types of proteins. The cross-validation set is used to train and evaluate the performance of the classifiers and representations, while the independent test set is used to evaluate the performance of the final models chosen based on the cross-validation set. It is possible that there may be differences in the characteristics of the two datasets, which could affect the performance of the models on the independent test set. For example, the distribution of protein lengths or the prevalence of certain types of proteins may be different between the two datasets, which could potentially affect the performance of the models.

The results presented in Table 3 and Figure 5 suggest that among the classifiers considered, the CNN generally performs the best in terms of the MCC. Specifically, on the cross-validation set, the CNN achieves a mean MCC of 0.9506, and on the independent set, the CNN also achieves the highest MCC with a value of 0.8584, outperforming the other classifiers. However, it is worth noting that the other classifiers also demonstrate strong performance on the ion channel prediction task, as indicated by the high mean values for the MCC and other evaluation metrics.

For example, the kNN, RF, and LR classifiers all achieve mean MCC values above 0.9 on the cross-validation set, and the RF and LR classifiers achieve mean MCC values above 0.9 on the independent set. One reason for the superior performance of the CNN may be its ability to learn more complex and expressive features from the protein sequences. As a type of deep learning model, the CNN is able to learn hierarchical representations of the data by applying multiple layers of convolution and pooling to the input sequences [58], allowing it to capture fine-grained and nuanced features of the sequences that may be more relevant for predicting ion channels.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 January 2023 doi:10.20944/preprints202301.0221.v1

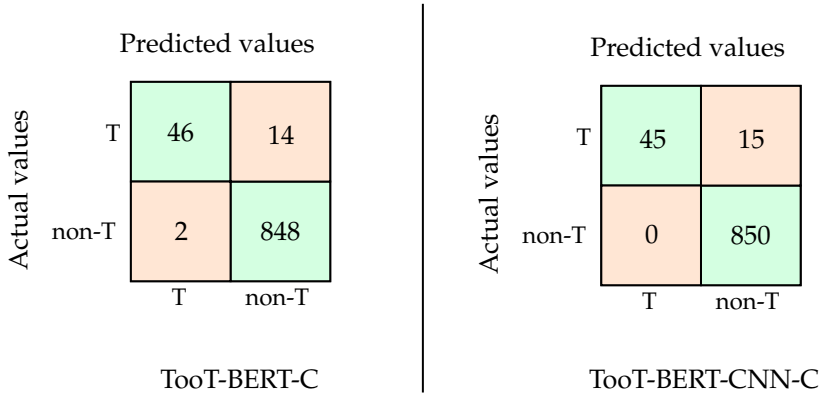

Figure 7. Confusion matrices for TooT-BERT-C and TooT-BERT-CNN-C. This figure presents confusion matrices for two approaches, TooT-BERT-C and TooT-BERT-CNN-C, used in the task of ion channel prediction.

The results presented in Table 4 are similar to those in Table 3, but in this table they are grouped by method (see Figure 6) and the average is taken for each method across the three representations. The average results in Table 4 show that the methods had high levels of accuracy, with the CNN achieving the highest score of 98.02%. In terms of sensitivity, the SVM and FFNN methods had the highest scores at 72.78%, while the CNN and SVM methods had the highest specificity scores at 100%. The MCC was highest for the CNN and LR methods, with scores of 0.8276 and 0.8009, respectively. Based on these results, the CNN method appears to have the best overall performance, followed by the SVM and FFNN methods.

3.3. Comparison to state-of-the-art approaches

Figure 7 presents the confusion matrices for the ion channel prediction task using the TooT-BERT-C and TooT-BERT-CNN-C approaches. A confusion matrix is a table that visualizes the performance of a binary classification model, with four cells representing the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model. The matrix for TooT-BERT-C indicates that the model made 46 TP, 848 TN, 2 FP, and 14 FN predictions. The matrix for TooT-BERT-CNN-C shows 45 TP, 850 TN, 0 FP, and 15 FN predictions. Both models exhibit strong performance, as indicated by the high number of TP and TN predictions. However, TooT-BERT-CNN-C appears to have slightly better performance, with a higher number of TN predictions and a lower number of FP predictions compared to TooT-BERT-C.

Table 5 compares the performance of the new approach, referred to as TooT-BERT-CNN-C, to three previous approaches for the task of ion channel prediction: DeepIon [14], MFPS_CNN [13], and TooT-BERT-C [34]. The table includes several different evaluation metrics, including accuracy, sensitivity, specificity, and MCC. The results are shown for both CV and independent sets. For each approach, the table shows the value of the metric on the independent set and the mean value over the folds of the cross-validation set.

The proposed approach demonstrates superior performance compared to the previous approaches. On the independent set, TooT-BERT-CNN-C achieves the highest values for most of the evaluation metrics, with the exception of sensitivity where MFPS_CNN performs slightly better. On the cross-validation set, TooT-BERT-CNN-C achieves the highest values for accuracy, specificity, and MCC. The strong generalization performance of TooT-BERT-CNN-C on the independent set suggests that it is able to accurately predict ion channels in unseen data.

One potential reason for the improved performance of TooT-BERT-CNN-C is the combination of BERT-based representations and a CNN classifier. BERT-based models are able to capture the context-aware structure of the sequences, while the CNN is able to learn hierarchical representations of the data.

 doi:10.20944/preprints202301.0221.v1

Table 5. Comparative performance of TooT-BERT-CNN-C with state-of-the-art. This table compares the performance of TooT-BERT-CNN-C with state-of-the-art approaches on cross-validated and independent test sets using evaluation metrics such as sensitivity, specificity, accuracy, and MCC. The maximum value in each column is highlighted in boldface.

Method	Acc(%)		Sen(%)		Spc(%)		MCC	
	Ind.	CV	Ind.	CV	Ind.	CV	Ind.	CV
DeepIon [14]	86.53	87.05	68.33	89.20	87.72	84.89	0.37	0.75
MFPS_CNN [13]	94.60	96.50	**76.70**	**95.00**	95.80	98.00	0.62	0.93
TooT-BERT-C [34]	98.24	98.96	76.67	86.71	99.76	**99.82**	0.85	0.91
TooT-BERT-CNN-C	**98.35**	**99.39**	75.00	93.38	**100.00**	**99.82**	**0.86**	**0.95**

Overall, our analysis shows that TooT-BERT-CNN-C outperforms TooT-BERT-C, as indicated by the results of a McNemar's test, which yielded a weak p-value of 0.0625.

4. Conclusions

In this study, we sought to accurately differentiate between ion channels and non-ion channels, a task of significant importance in the fields of biology and medicine. We previously demonstrated the effectiveness of protein language models for this purpose through the use of logistic regression with ProtBERT-BFD representations, resulting in the TooT-BERT-C approach. In this paper, we expanded upon this work by evaluating various classical classifiers, including k-nearest neighbors, random forest, support vector machine, and a feed-forward neural network, as well as logistic regression, and implementing a convolutional neural network for comparison. Our results showed that the new approach, TooT-BERT-CNN-C, outperformed the state-of-the-art and the previous approach, with an increase in Matthews Correlation Coefficient from 0.8486 to 0.8584 and improvement in accuracy from 98.24% to 98.35%. This demonstrates the effectiveness of our method in distinguishing between ion channels and non-ion channels, and highlights the potential of combining pre-trained language models and deep learning models for this task. In future work, we plan to investigate other pre-trained language models and deep learning models, as well as alternative representations and feature engineering techniques.

Author Contributions: Conceptualization, HG and GB; methodology, HG and GB; software, HG; validation, HG; formal analysis, HG and GB; investigation, HG; resources, HG and GB; data curation, HG; writing—original draft preparation, HG and GB; writing—review and editing, HG and GB; visualization, HG; supervision, GB; project administration, GB; All authors have read and agreed to the published version of the manuscript.

Funding: Both authors are supported by Natural Sciences and Engineering Research Council of Canada (NSERC), Genome Québec, and Genome Canada and Concordia University.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The data used in this study is available at https://tootsuite.encs.concordia.ca/datasets/TooT-BERT-C, and the code for the proposed method, TooT-BERT-CNN-C, is available at https://github.com/bioinformatics-group/toot_bert_cnn_c.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

kNN	k-Nearest Neighbors
RF	Random Forest
SVM	Support Vector Machine
LR	Logistic Regression
FFNN	Feed-Forward Neural Network
CNN	Convolutional Neural Network
MCC	Matthews Correlation Coefficient
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
t-SNE	t-distributed Stochastic Neighbor Embedding

References

1. Hille, B. *Ionic Channels of Excitable Membranes*, 3 ed.; Vol. 21, Springer, 2001.
2. Nogueira, J.J.; Corry, B. Ion Channel Permeation and Selectivity. In *The Oxford Handbook of Neuronal Ion Channels*; Bhattacharjee, A., Ed.; Oxford University Press, 2019.
3. Restrepo-Angulo, I.; De Vizcaya-Ruiz, A.; Camacho, J. Ion channels in toxicology. *Journal of Applied Toxicology* **2010**, *30*, 497–512.
4. Eisenberg, B. From structure to function in open ionic channels. *Journal of Membrane Biology* **1999**, *171*, 1–24.
5. Kulbacka, J.; Choromańska, A.; Rossowska, J.; Weżgowiec, J.; Saczko, J.; Rols, M.P. Cell Membrane Transport Mechanisms: Ion Channels and Electrical Properties of Cell Membranes. In *Transport Across Natural and Modified Biological Membranes and its Implications in Physiology and Therapy*; Kulbacka, J.; Satkauskas, S., Eds.; Advances in Anatomy, Embryology and Cell Biology, Springer International Publishing: Cham, 2017; pp. 39–58.
6. Clare, J.J. Targeting ion channels for drug discovery. *Discovery Medicine* **2010**, *9*, 253–260.
7. Picci, G.; Marchesan, S.; Caltagirone, C. Ion channels and transporters as therapeutic agents: From biomolecules to supramolecular medicinal chemistry. *Biomedicines* **2022**, *10*, 885.
8. Ashrafuzzaman, M. Artificial intelligence, machine learning and deep learning in ion channel bioinformatics. *Membranes* **2021**, *11*, 672.
9. Menke, J.; Maskri, S.; Koch, O. Computational ion channel research: From the application of artificial intelligence to molecular dynamics simulations. *Cellular Physiology and Biochemistry: International Journal of Experimental Cellular Physiology, Biochemistry, and Pharmacology* **2021**, *55*, 14–45.
10. Zhao, Y.W.; Su, Z.D.; Yang, W.; Lin, H.; Chen, W.; Tang, H. IonchanPred 2.0: A tool to predict ion channels and their types. *International Journal of Molecular Sciences* **2017**, *18*, 1838. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.
11. Gao, J.; Cui, W.; Sheng, Y.; Ruan, J.; Kurgan, L. PSIONplus: Accurate sequence-based predictor of ion channels and their types. *PLOS ONE* **2016**, *11*, e0152964. Publisher: Public Library of Science.
12. Gao, J.; Wei, H.; Cano, A.; Kurgan, L. PSIONplusm server for accurate multi-label prediction of ion channels and their types. *Biomolecules* **2020**, *10*, 876. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.
13. Nguyen, T.T.D.; Ho, Q.T.; Tarn, Y.C.; Ou, Y.Y. MFPS_CNN: Multi-filter pattern scanning from position-specific scoring matrix with convolutional neural network for efficient prediction of ion transporters. *Molecular Informatics* **2022**, p. e2100271.
14. Taju, S.W.; Ou, Y.Y. DeepIon: Deep learning approach for classifying ion transporters and ion channels from membrane proteins. *Journal of Computational Chemistry* **2019**, *40*, 1521–1529.
15. Lin, H.; Chen, W. Briefing in Application of Machine Learning Methods in Ion Channel Prediction. *The Scientific World Journal* **2015**, *2015*, e945927. Publisher: Hindawi.
16. Asgari, E.; Mofrad, M.R.K. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE* **2015**, *10*, e0141287.
17. Rao, R.M.; Liu, J.; Verkuil, R.; Meier, J.; Canny, J.; Abbeel, P.; Sercu, T.; Rives, A. MSA transformer. In Proceedings of the Proceedings of the 38th International Conference on Machine Learning. PMLR, 2021, pp. 8844–8856. ISSN: 2640-3498.
18. Rao, R.; Bhattacharya, N.; Thomas, N.; Duan, Y.; Chen, P.; Canny, J.; Abbeel, P.; Song, Y. Evaluating protein transfer learning with TAPE. In Proceedings of the Advances in Neural Information Processing Systems; Wallach, H.; Larochelle, H.; Beygelzimer, A.; Alché-Buc, F.d.; Fox, E.; Garnett, R., Eds. Curran Associates, Inc., 2019, Vol. 32.
19. Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; et al. ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2021**, pp. 1–1.
20. Heinzinger, M.; Elnaggar, A.; Wang, Y.; Dallago, C.; Nechaev, D.; Matthes, F.; Rost, B. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics* **2019**, *20*, 723.
21. Unsal, S.; Ataş, H.; Albayrak, M.; Turhan, K.; Acar, A.C.; Doğan, T. Evaluation of methods for protein representation learning: A quantitative analysis. Technical report, bioRxiv, 2020.
22. Kotsiliti, E. De novo protein design with a language model. *Nature Biotechnology* **2022**, *40*, 1433–1433.

23. Unsal, S.; Atas, H.; Albayrak, M.; Turhan, K.; Acar, A.C.; Doğan, T. Learning functional properties of proteins with language models. *Nature Machine Intelligence* **2022**, *4*, 227–245.
24. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805 [cs]* **2019**.
25. Aggarwal, D.; Hasija, Y. A review of deep learning techniques for protein function prediction **2022**.
26. Xu, M.; Zhang, Z.; Lu, J.; Zhu, Z.; Zhang, Y.; Ma, C.; Liu, R.; Tang, J. PEER: A comprehensive and multi-task benchmark for protein sequence understanding, 2022. arXiv:2206.02096 [cs].
27. Elofsson, A. Protein structure prediction until CASP15, 2022. arXiv:2212.07702 [q-bio].
28. Erdős, G.; Dosztányi, Z. Chapter 7 - Prediction of protein structure and intrinsic disorder in the era of deep learning. In *Structure and Intrinsic Disorder in Enzymology*; Gupta, M.N.; Uversky, V.N., Eds.; Foundations and Frontiers in Enzymology, Academic Press, 2023; pp. 199–224.
29. Suzek, B.E.; Wang, Y.; Huang, H.; McGarvey, P.B.; Wu, C.H.; the UniProt Consortium. UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **2015**, *31*, 926–932.
30. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583–589.
31. Ghazikhani, H.; Butler, G. TooT-BERT-M: Discriminating membrane proteins from non-membrane proteins using a BERT representation of protein primary sequences. In Proceedings of the 2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2022, pp. 1–8.
32. Alballa, M.; Butler, G. Integrative approach for detecting membrane proteins. *BMC Bioinformatics* **2020**, *21*, 575.
33. Ghazikhani, H.; Butler, G. TooT-BERT-T: A BERT Approach on Discriminating Transport Proteins from Non-transport Proteins. In Proceedings of the Practical Applications of Computational Biology and Bioinformatics, 16th International Conference (PACBB 2022); Fdez-Riverola, F.; Rocha, M.; Mohamad, M.S.; Caraiman, S.; Gil-González, A.B., Eds.; Springer International Publishing: Cham, 2023; Lecture Notes in Networks and Systems, pp. 1–11.
34. Ghazikhani, H.; Butler, G. TooT-BERT-C: A Study on Discriminating Ion Channels from Membrane Proteins based on the Primary Sequence's Contextual Representation from BERT Models. In Proceedings of the 2022 9th International Conference on Bioinformatics Research and Applications (ICBRA 2022), September 18–20, 2022, Berlin, Germany. ACM Conference Proceedings, 2022. doi: 10.1145/3569192.3569196, (submitted; accepted; in press).
35. Apweiler, R.; Bairoch, A.; Wu, C.H.; Barker, W.C.; Boeckmann, B.; Ferro, S.; Gasteiger, E.; Huang, H.; Lopez, R.; Magrane, M.; et al. UniProt: The Universal Protein knowledgebase. *Nucleic Acids Research* **2004**, *32*, D115–D119.
36. Altschul, S.F.; Gish, W.; Miller, W.; Myers, E.W.; Lipman, D.J. Basic local alignment search tool. *Journal of Molecular Biology* **1990**, *215*, 403–410.
37. Rostami, M.; He, H.; Chen, M.; Roth, D. Transfer learning via representation learning. In *Federated and Transfer Learning*; Razavi-Far, R.; Wang, B.; Taylor, M.E.; Yang, Q., Eds.; Adaptation, Learning, and Optimization, Springer International Publishing: Cham, 2023; pp. 233–257.
38. Zhang, H.; Li, G.; Li, J.; Zhang, Z.; Zhu, Y.; Jin, Z. fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively, 2022. arXiv:2211.01642 [cs].
39. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, [1706.03762].
40. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for self-supervised learning of language representations. Technical report, arXiv, 2020. arXiv:1909.11942 [cs] version: 6 type: article.
41. Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C.L.; Ma, J.; et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences* **2021**, *118*, e2016239118. Publisher: Proceedings of the National Academy of Sciences.
42. Detlefsen, N.S.; Hauberg, S.; Boomsma, W. Learning meaningful representations of protein sequences. *Nature Communications* **2022**, *13*, 1914.
43. Suzek, B.E.; Wang, Y.; Huang, H.; McGarvey, P.B.; Wu, C.H.; Consortium, U. UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **2014**, *31*, 926–932. Publisher: Oxford University Press.
44. Habibi, N.; Mousavi, S. A survey on applications of machine learning in bioinformatics and neuroscience. *Majlesi Journal of Telecommunication Devices* **2022**, *11*, 95–111. Number: 2.
45. Kramer, O. Scikit-Learn. In *Machine Learning for Evolution Strategies*; Kramer, O., Ed.; Springer International Publishing: Cham, 2016; pp. 45–53.
46. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems. Curran Associates, Inc., 2019, Vol. 32.
47. Stoltzfus, J.C. Logistic regression: A Brief Primer. *Academic Emergency Medicine* **2011**, *18*, 1099–1104.
48. Cai, Y.D.; Zhou, G.P.; Chou, K.C. Support Vector Machines for Predicting Membrane Protein Types by Using Functional Domain Composition. *Biophysical Journal* **2003**, *84*, 3257–3263.
49. Byvatov, E.; Schneider, G. Support vector machine applications in bioinformatics. *Applied Bioinformatics* **2003**, *2*, 67–77.

 doi:10.20944/preprints202301.0221.v1

50. Maltarollo, V.G.; Kronenberger, T.; Espinoza, G.Z.; Oliveira, P.R.; Honorio, K.M. Advances with support vector machines for novel drug discovery. *Expert Opinion on Drug Discovery* **2019**, *14*, 23–33.
51. Qi, Y. Random Forest for Bioinformatics. In *Ensemble Machine Learning: Methods and Applications*; Zhang, C.; Ma, Y., Eds.; Springer US: Boston, MA, 2012; pp. 307–323.
52. Jiang, L.; Cai, Z.; Wang, D.; Jiang, S. Survey of Improving k-Nearest-Neighbor for Classification. In Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), 2007, Vol. 1, pp. 679–683.
53. Bishop, C.M. Neural Networks for Pattern Recognition. In *Pattern Recognition and Machine Learning*; Oxford University Press, 1995; pp. 225–290. Chapter 5.
54. Amanatidis, D.; Vaitsi, K.; Dossis, M. Deep Neural Network Applications for Bioinformatics. In Proceedings of the 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), 2022, pp. 1–9.
55. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]* **2017**.
56. O'Shea, K.; Nash, R. An introduction to convolutional neural networks, 2015. arXiv:1511.08458 [cs].
57. Aggarwal, C.C. Convolutional Neural Networks. In *Neural Networks and Deep Learning: A Textbook*; Aggarwal, C.C., Ed.; Springer International Publishing: Cham, 2018; pp. 315–371.
58. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6.
59. Seo, S.; Oh, M.; Park, Y.; Kim, S. DeepFam: Deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics* **2018**, *34*, i254–i262.
60. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* **2014**, *15*, 1929–1958.
61. Pembury Smith, M.Q.R.; Ruxton, G.D. Effective use of the McNemar test. *Behavioral Ecology and Sociobiology* **2020**, *74*, 133.
62. Chicco, D.; Jurman, G. The advantages of the Matthews Correlation Coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* **2020**, *21*, 6.
63. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **2008**, *9*.