DEEP LEARNING FOR REMOTE HOMOLOGY

A PREPRINT

Student:
Manish Kumar Keshri
Department of Computer Science
University of Minnesota, Twin Cities
200 Union St SE, Minneapolis, MN 55455

Adviser:
Rui Kuang
Department of Computer Science
University of Minnesota, Twin Cities
200 Union St SE, Minneapolis, MN 55455

June 11, 2021

ABSTRACT

Homology is the existence of shared ancestry between a pair of proteins. Proteins with similar sequences are assumed to be homologous and usually have similar structures and functions. Any pair of proteins which are remotely related evolutionary is considered as a remote homology. Remote homology detection is the problem of classification of proteins into structural and functional classes given their amino acid sequences, particularly with low sequence similarity. Advancement in Natural Language Processing using deep learning has led to exploring its use in protein sequence classification. Similar to text which is made up of characters, protein sequences are made up of amino acids represented by different characters and arranged in a linear chain. Here, we have collected data from Structural Classification of Proteins(SCOP)[1] database. We encode the sequences into matrices using two different types of encoding format: One-hot encoding and PSSM and feed them as input to Recurrent Neural Network(RNN), Convolutional Neural Network(CNN) models.

1 Introduction

One of the way to effectively determine the function of a protein is to find out what functions are performed by homologous proteins. Therefore, protein remote homology detection is important for determination of 3D structure and function of experimentally unanalyzed proteins. Proteins are made up of amino acids, joined together in linear chains. Some proteins are just a few amino acids long, while others are made up of several thousands. These chains of amino acids fold up in complex ways which are dictated by their amino acids sequence, giving each protein a unique 3D shape which determines its function. The experimental determination of structure of proteins is a labor-intensive and time-consuming process. The number of known proteins is constantly increasing and only a small fraction of them have been experimentally analyzed in order to detect their structure and hence their function in the corresponding organism. Therefore there is the need for automated tools that can classify new proteins to structural families.

2 Background

2.1 Protein Structure Types

As I mentioned, the structure of a protein determines its activity. However, there are four distinct levels of protein structure.

2.1.1 Primary structure

Primary structure is the simplest level of a protein structure which is basically the sequence of amino acids in a linear chain. It controls all subsequent levels of protein organization. The amino acid sequence of the primary structure drives the folding and intramolecular bonding of the linear amino acid chain, which ultimately determines the protein's unique 3D shape.

2.1.2 Secondary structure

Protein secondary structure is the three dimensional form of local segments of proteins. The two most common secondary structural elements are alpha helices and beta sheets, though beta turns and omega loops occur as well. Secondary structure elements typically spontaneously form as an intermediate before the protein folds into its three dimensional tertiary structure.

2.1.3 Tertiary structure

Tertiary structure is the complete three-dimensional (3-D) structure of a protein.

2.2 Protein family tree

Proteins are hierarchically organized based on their structural and evolutionary relationships. Specifically, there are four different levels in the hierarchy from bottom to top: class, fold, superfamily, family. These levels determine the evolutionary relationships between proteins. Proteins with related structures tend to behave in similar ways, and these proteins are therefore considered a protein family.

2.2.1 Family

Proteins with high sequence similarity (e.g., >40% percent sequence identity)[2], are grouped together into families. The proteins within a particular family tend to perform similar functions within the cell and also often have long stretches of similar amino acid sequences within their primary structure.

2.2.2 Superfamily

Proteins that have lower sequence similarity (<30%) are grouped together into broader evolutionary families or superfamilies[2]. Superfamilies typically contain several protein families which show sequence similarity within each family. Proteins that belong to the same superfamily, they tend to have low sequence identities but its structure similarity has enough evidence to indicate evolutionary relatedness.

2.2.3 Fold

Proteins that belong to the same fold should have similar tertiary structure but may not be evolutionary related.

2.3 Natural Language Processing and Deep Learning

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language.

Text comprising of words and characters belong to the set of semi-structured data. Textual analysis has presented many challenges like text classification, spam detection, part of speech tagging, named entity recognition etc.

With the advancement of deep learning and availability of large datasets, methods of handling text understanding using deep learning techniques have gradually become available. There are many techniques involved in text processing like embedding, design of the model, cost function estimation etc. Embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. They are generally vector representations of a particular word which is used for distributed representations of text in an 2-dimensional space. Word2Vec[3] and Glove[4] are most popular methods of word embeddings.

2.4 Different types of encoding

2.4.1 One-hot encoding

One-hot encoding is widely used in natural language processing and information retrieve. To apply it into protein sequencing, we simply convert each amino acid character to 20 one-hot discrete vector (We only choose the most stable 20 amino acids). More specifically, given a sequence that has length L, the one-hot encoding will convert this sequence into L X 20 matrix.

As obvious, one-hot encoding is prone to the curse of dimensionality[5]. Besides, it also fails to consider the relationship between amino acids. For instance, the one hot vector presentation of "school" and "education" will be completely different. The semantic connection between two word is ignored in this encoding type.

As for remote homology detection, it might also ignore some function and structural connection between two local units. To explicitly model this connection, PSSM encoding is proposed in the following section.

2.4.2 BLOSUM62

BLOck SUbstitution Matrix is based on comparisons of Blocks of sequences derived from the Blocks database[6]. The Blocks database contains multiply aligned ungapped segments corresponding to the most highly conserved regions of proteins (local alignment versus global alignment). BLOSUM matrices are derived from blocks whose alignment corresponds to the BLOSUM-matrix number (e.g. BLOSUM 62 is derived from Blocks containing >62% identity in ungapped sequence alignment). BLOSUM62 is the default matrix for the standard protein BLAST program.

2.4.3 Position-Specific scoring matrix

A profile matrix is built for each protein sequence either from the multiple alignment of the family of closely related sequences to the given sequence or directly from an input multiple alignment.The profile helps to distinguish between conserved positions that are important for defining members of this family and non-conserved positions that are variable among the members of the family. Furthermore, it describes exactly what variation in amino acids is possible at each position by recording the probability for the occurrence of each amino acid along the multiple alignment. One such postion-specific scoring system is PSI-BLAST[19].

Given a sequence, we first run PSI-BLAST against database and then we build profiles according to the search results. So each sequence will have its corresponding profiles. Instead of using the original sequence as input, we now use its profiles representation as input. The idea is to give a probabilistic representation of the biosequence and hope the new representation may have certain better property. There are cases where we cannot build its profiles. A common solution was to use BLOSUM62 as their encoding matrix, which works well in previous literature.

2.5 Neural Networks for Protein Classification

The selection of right kind of neural network is the important for text analysis. There are several types of neural networks. However, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are mostly used in NLP. We present here these neural networks for protein sequence analysis.

2.5.1 Convolutional Neural Network (CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery[Wikipedia]. CNNs are regularized mutilayer neural networks which take advantage of the hierarchical pattern in data by assembling more complex patterns using smaller and simpler patterns and thereby decreasing the complexity of a fully connected neural network. They are also known to have shift invariance and locality[7]. The shift invariance ensures that the prediction of an image will not change when the object moves within the image. As for natural language processing, an 1d variant of CNN allows researchers to extract meaningful local pattern in the sequence data, which demonstrates its effectiveness in many tasks such as text classification, named entity recognition and machine translation.

In remote homology detection, capturing local invariance in the protein sequences is very useful information for the model. The CNN model allows to extract local features that are suitable for the remote homology detection. 1D CNN can be used for extracting local 1D patches (subsequences) from sequences and can identify local patterns within the window of convolution. And because the same transformation is applied on every patch identified by the window, a pattern learnt at one position can also be recognized at a different position, making 1D conv nets translation invariant. If we use 1D convolution, CNN can automatically find the suitable representation of local kmers. It seems to be a natural extension of traditional kmer based approaches in this area.

More specifically, given a sequence

$$W = w_1, w_2 ... w_n$$

where w_t denotes the t-th amino acid or kmer-amino acid, a convolution operator is applied to a window of h amino acids. For example, a feature c_i is generated from a window of words $x_{i:i+h1}$:

$$c_i = f(W \cdot x_{i:i+h1} + b)$$

where b is bias term and W is the filter corresponding to the 1d convolutional operator. This filter is applied to all the local continuous units of the sequence to group feature map c_1, c_2, ...c_{nh+1}. The sequences are truncated or padded to have same overall length which helps to feed in the activations from CNN model to feed the forward neural network. This feature map is then passed to pooling layer for max pooling or mean pooling. The idea of pooling layer is to capture the most important feature within the feature map and enlarge the context size of each unit.

2.5.2 Wildcard kernel filters in CNN

In a CNN model, regular kernel filters cannot replicate mutations in the protein sequences. To overcome this, we used wildcard kernels[26]. In the implementation, we change the filters in the input layer of the network. Since all kernel filters are one-dimensional, we select a position within the kernel filter dimension and make all the weights at that position as zero and these zero weights are never updated in the course of training the model. This zeroing of weights helps to discard the effect of the amino acid present at that position. These positions are randomly selected across all filters and as such mutations in the protein sequences is replicated.

Instead of randomly selecting different positions across all filters in the first layer of a model, we can keep one position fixed across all filters. Ensemble models with different fixed positions would be another way of wildcard kernel implementation. In this way, one model can learn to detect mutation at a particular position of the kernel dimension and several models together can learn to detect mutations at different positions.

2.5.3 Long Short-Term Memory

Long Short-Term Memory(LSTM) is special form of Recurrent Neural Network(RNN) widely used in the field of natural language processing. LSTM has the ability to capture long term dependency and higher order relationship, which makes it suitable for remote homology detection. Specially, a common LSTM unit is composed of a input gate i_k, a forget gate f_k, an output gate o_k and a memory cell c_k. This gated control were developed to deal with the exploding and vanishing gradient problems that can be encountered with training RNNs with long sequences.

Given a protein sequence, the sequence input is first converted into a matrix. Similar to the procedure we did for CNN, this input matrix can either comes from one-hot encoding, PSSM or BLOSUM62. The matrix is then fed into the LSTM unit. Specifically, Given a sequence

$$W = w_1, w_2 ... w_n$$

where w_t denotes the t-th amino acid, we find its corresponding representation through one-hot encoding, PSSM or BLOSUM62 matrix and then feed it into LSTM unit.

In RNN, weights of hidden layer denoted as w_h and weights of output layer denoted as w_y, weights of recurrent computation denoted as w_r, hidden representation denoted as h and output denoted as y, RNN Network can be formulated as

$$h_t = (W_h W + W_r h_{t1})$$

$$y = f(W_y h_t)$$

The sequences are truncated or padded to have same overall length which helps to feed in the activations from LSTM model to the feed forward neural network.

3 Related Work

Over the past decades, various methods have been proposed for the task of homology detection. Dynamic programming algorithms, such as the Needleman-Wunsch[8] and Smith-Waterman[9] algorithms calculates the score of the local optimal pairwise alignment of two sequences. These algorithms calculate the similarity score of the given sequence with all the database sequences. The given and database sequences are classified as homologous if the score exceeds some predefined threshold. Since, they make all possible pairwise comparisons to all of the database sequences, they are computationally expensive.

Later, machine learning based classification models were proposed, which treat protein remote homology protein detection as a superfamily level classification task. These methods take the advantages of machine learning algorithms by using both positive and negative samples to train a classifier [10, 11]. Support Vector Machine have been most effective in this field[12,13]. Kernel tricks are employed to measure the similarity between protein pairs and several kernels have been proposed, such as profile kernel[18], mismatch kernel [14], motif kernel [15], LA kernel [16], SW-PSSM [13], SVM-Pairwise [17], etc.

Deep learning based techniques have also been proposed for improving the discriminative power compared with other machine learning methods. Recurrent Neural Network (RNN) with LSTMs and GRUs have been the most successful deep learning techniques. LSTM networks have also been used to generate representation of proteins [20]. ProDeC-BLSTM[21] based on Bidirectional Long Short-Term Memory with attention have been proposed for remote homology detection.

4 Experiments

4.1 Datasets

The Structural Classification of Proteins (SCOP) database provides a detailed and comprehensive description of the relationships of all known proteins structures. A widely used benchmark dataset has been used to evaluate the performance of various methods [15], which was constructed based on the SCOP database [22] by Hochreiter [20]. This dataset can be accessed from `http://www.bioinf.jku.at/software/LSTM_protein/`.

The SCOP database [22] classifies the protein sequences into a hierarchy structure, whose levels from top to bottom are class, fold, superfamily, and family. 4019 proteins sequences are extracted from SCOP database, whose identities are lower than 95%, and they are divided into 102 families and 52 superfamilies. For each family, there are at least 10 positive samples. For the 102 families in the database, the training and testing datasets are defined as:

$$S_{\text{train}}(k) = S_{\text{train}}{}^{+}(k) \cup E_{\text{train}}{}^{+}(k) \cup S_{\text{train}}{}^{-}(k)$$

$$S_{\text{test}}(k) = S_{\text{test}}{}^{+}(k) \cup S_{\text{test}}{}^{-}(k)$$

where $S_{test}{}^{+}$(k) represents the k^{th} positive testing dataset with proteins in k^{th} family, and $S_{train}{}^{+}$(k) represents the k^{th} positive training dataset containing proteins in the same superfamily and not in the k^{th} family. $E_{train}{}^{+}$(k) denotes the extended positive training dataset for k th training dataset. The added training samples are extracted from Uniref50 [23] by using PSI-BLAST [19] with default parameters except that the e-value was set as 10.0. For all of the superfamilies except which k^{th} family belongs to, select one family in each of the superfamilies respectively, to form the k^{th} negative testing dataset $S_{test}{}^{-}$(k) and the rest of proteins in these superfamilies are included in the negative training dataset $S_{train}{}^{-}$(k). The average number of samples of all the 102 training datasets is 9077.

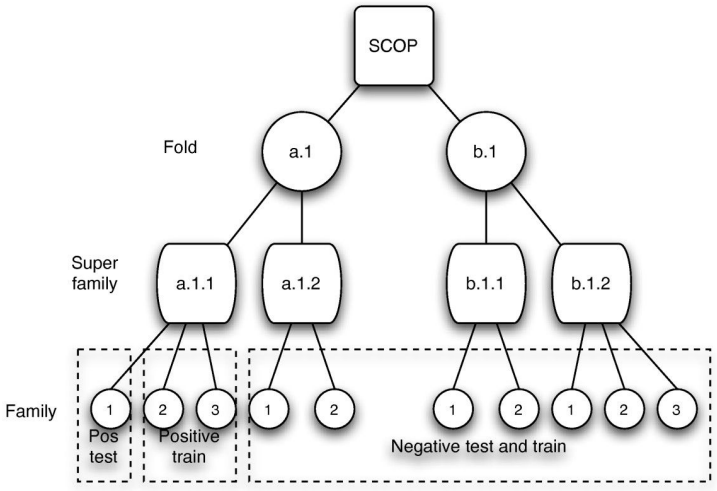

Figure 1: The figure shows how the SCOP database is divided into training and test sets. For each classifier tested on the superfamily benchmark, the sequences of the SCOP database are divided into positive and negative training and test sets. One SCOP family is used as a positive test set. The negative test set is made from one random family from each of the other superfamilies. The positive training set is made from the superfamily of the classifier, excluding the positive test set family. The negative training set is made from the other superfamilies, excluding the negative test sets. *Image Scource: References [15]*

4.2 Sequence Encoding

The input to the classification model is a protein sequence converted into a matrix of real values. A protein sequence X of length n is represented by a sequence of characters X = a1, a2,...,an. such that each character corresponds to one of the 20 standard amino acids. The matrix can be formed by using different types of encoding like one-hot, BLOSUM62 and PSSM. For PSSM matrix generation, we used PSI-BLAST with default parameters except that the e-value was set as 0.01 and number of iterations was set to 5.

4.3 Neural network architecture

Key to our algorithm for protein classification is its learning methodology, which is based on Convolutional Neural Networks. Given a set of positive training sequences S_{train}^{+} and a set of negative training sequences S_{train}^{-}, the model learns a classification function. To evaluate the model performance, we record the validation error during each epoch on the set of positive test sequences S_{test}^{+} and a set of negative test sequences S_{test}^{-}.

The overall framework is the model presented below. Here, CNN layers progressively learn higher level features of the sequences. Each Convolutional layer consists 40 filters to extract patterns from the sequences followed by a non-linear transformation of RelU. The layers operate on 3-D chunks of data, where the first two dimensions are (generally) the height and width of an encoded matrix, and the third dimension is a number of such matrices stacked over one another, which is also called the number of channels in the encoded matrix volume. We reduce the channels to 2-D matrix in each layer by selecting the maximum values across the stacked channels. The first row of these matrices across all layers are concatenated into one and fed into the fully connected layers. The two fully connected layers have 200 hidden units in each of them which transforms the input into a vector of two, one for each class. The softmax layer transforms these output values into the probability values for each class. For each for the protein family, positive and negative training sequences are given and the model trains on the sequences. This architecture was proposed by Hoada [5], and we used it with default hyperparameters with SGD as the optimizer and filter size as 6. Once the model is trained, a new sequence X is predicted to be positive or negative class.

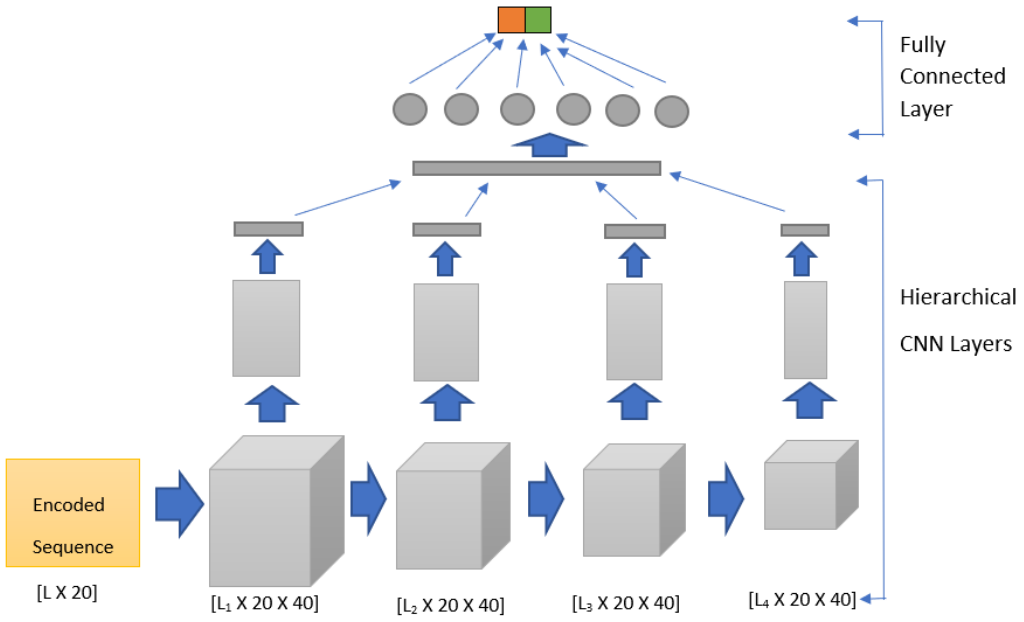

Figure 2: The structure of the used neural network. The input is the encoded matrix of the protein sequence. Next, layers of convolution + RelU is applied for extracting the sequence patterns. Hierarchical CNN layers pool the output of each CNN layer by extracting the maximum across the channels. The first row of the pooled output is fed into a fully connected layer. A softmax layer at the end transforms the final output into probability values for each class.

Encoding type	ROC	ROC50
One-hot	0.926	0.615
BLOSUM62	0.956	0.782
PSSM	0.971	0.825

Table 1: Performance comparison of the model across different types of encoding

4.4 Metrics

Since the dataset is imbalanced, we couldn't use accuracy as the measure of the performance. Following previous work [12], we use ROC and ROC50 to evaluate the performance of the model. ROC is also known as the area under the curve which is built by plotting the true positive rate against false positive rate under different threshold. ROC 50 only counts up to the first 50 false positives.

We train the model for each family. In total, 102 models are trained and validated for each family. Each model takes around 10 minutes to train on NVIDIA 1080TI GPU. The average ROC and ROC50 values are used for to evaluate the overall performance of the model.

4.5 Results for different encoding

We tried a number of experiments to determine best encoding. One-hot and BLOSUM62 were reasonable in their performance but PSSM profile encoding gave the best results in this data. There are cases where we cannot build its profiles. A common solution was to use BLOSUM62 as their encoding matrix, which works well in the previous literature as well. One common characteristic across all encodings is the width of the encoded matrix which is 20. Table 1 presents the comparison of performance of the model on different encodings.

4.6 Results on sparse matrix with PSSM embedding

To introduce sparsity in the PSSM embedded matrix, we kept different thresholds to select values in the matrix. The absolute value of values in the matrix was compared against the thresholds, and if the absolute value was less than the threshold, it was replaced with zero. Introducing sparsity will reduce the noise in the encoding as well as reduce the computation time with sparse matrix multiplication. The values in the PSSM profile matrix can take integer values between -16 to +14, and when these values were plotted as a histogram, they seem to follow a normal distribution. We kept different thresholds to obtain different percentages of sparsity. Finally, Table 2 and Figure 3 to show sparsity vs performance of the model.

Sparsity percentage	ROC	ROC50
0	0.971	0.825
13	0.971	0.825
37	0.974	0.825
57	0.974	0.829
73	0.967	0.788
84	0.951	0.732
91	0.936	0.676
96	0.926	0.614

Table 2: Model performance in terms of ROC and ROC50 for varying percentages of sparsity in the embeddded matrix.

4.7 Results on use of Wildcard Kernel in the CNN layers

We tried to use different types of kernels instead of regular kernels in CNN layers. One such kernel is wildcard kernel which replicates the mutations in protein sequences. We tried both kinds of wildcard kernel implementations as decribed above in section 2.5.2. However in the ensemble model implementation, we tried to evaluate mutations at position 1, 2, 3, and 4 of size 6 one-dimensional kernel filter. The performance achieved on the use of wildcard kernels with one-hot encoding are summarized below in Table 3.

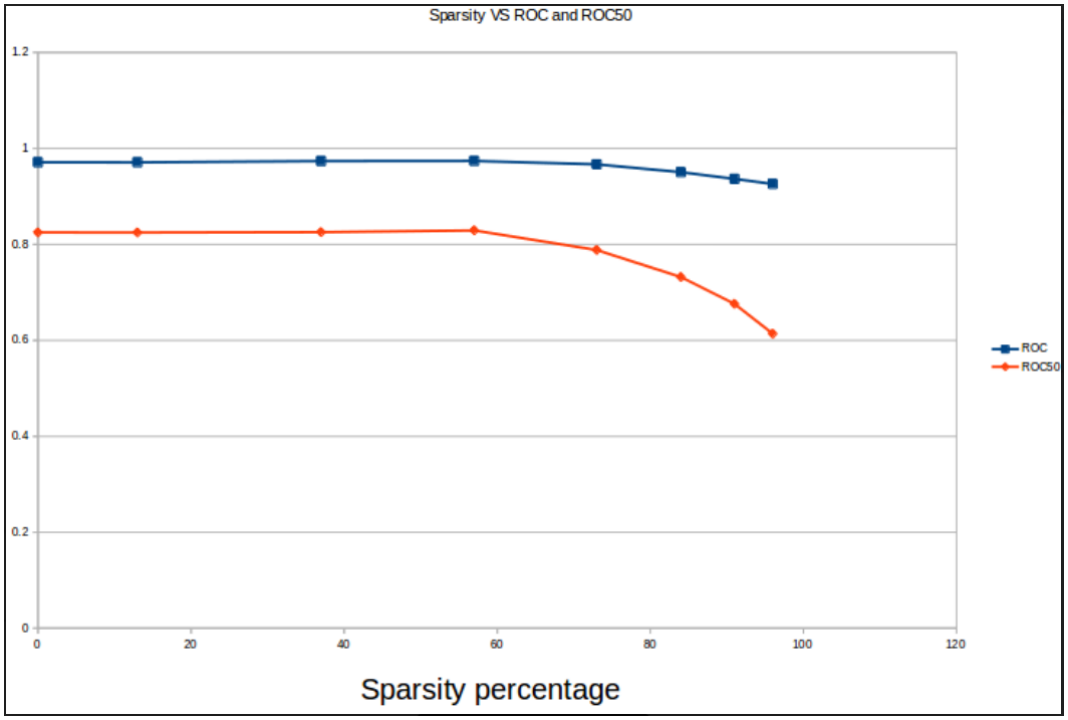

Figure 3: Model performance vs Percentage sparsity in the embedded matrix

Type of wildcard kernel implemenation	ROC	ROC50
Single model	0.881	0.485
Ensemble 1,2,3,4	0.916	0.593

Table 3: Model performance in terms of ROC and ROC50 for different wildcard kernel implementations in CNN

5 Discussion and Future Work

From the previous results, we have seen that we can use different kinds of embedding to achieve significant boost in performances. PSSM embedding generated by PSI-BLAST gives the highest performance. However, we can use machine learning models to learn the embedding in a semi-supervised manner, by taking advantage of the unlabeled data.

ELMo[24] gained its language understanding from being trained to predict the next word in a sequence of words - a task called Language Modeling. This is convenient because we have vast amounts of text data that such a model can learn from without needing labels. One of the disadvantage of ELMo[24] that it can't take advantage of both left and right contexts simultaneously which is eliminated using BERT[25]. This type of language modelling can be used to build embedding for protein sequence as well as we have vast amounts of unlabeled data.

While Word2Vec[3] is a predictive model that predicts context given word, Glove[4] learns by constructing a co-occurrence matrix (words X context) that basically count how frequently a word appears in a context. This matrix is factorized to achieve a lower-dimension representation. Similarly, we can build embedding for kmers in protein sequences.

Gappy kernels can find similar sequences that do not match exactly but with some gaps which replicates mutations and reading errors. We can try to use this kernel instead of regular kernels in CNN layers. Though it will be computationally expensive, but we can expect some improvement in the performance.

6 Conclusion

We presented use of different types of encoding for protein sequences and neural network architecture that can get competitive results in remote homology detection.

Though increasing the sparsity in PSSM embedding slows down the convergence and increases the number of epochs to get the best performance. But at the same time, increased sparsity calls for sparse matrix computations which is relatively computationally cheaper than dense matrix computations. By sparse computation implementation, we can actually decrease the time to converge for models with sparse input.

References

[1] Loredana Lo Conte, Bart Ailey, Tim JP Hubbard, Steven E Brenner, Alexey G Murzin, and Cyrus Chothia. Scop: a structural classification of proteins database. *Nucleic acids research*, 28(1):257–259, 2000.

[2] Christine A. Orengo and Janet M. Thornton. Protein families and their evolution-a structural perspective. *Annual Review of Biochemistry*,74:867-900, 2005.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean Distributed Representations of Words and Phrases and their Compositionality. *Neural Information Processing Systems*,arXiv:1310.4546, 2013.

[4] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*,D14-1162, 2014.

[5] Hoada Chu. Deep Representation Learning for Remote Homology. *University of Minnesota, Master's Project*,2019.

[6] Henikoff S, Henikoff JG. Amino acid substitution matrices from protein blocks. *Comparative Study Research Support, U.S. Gov't, P.H.S.*,89(22):10915-9, 1992.

[7] Yann LeCun, L eon Bottou, Yoshua Bengio, and Patrick Haffner Gradient-based learning applied to document recognition. *Proceedings of the IEEE*,86(11):2278–2324, 1998.

[8] Needleman, S. B. and Wunsch, C. D.. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*,48, 443-453, 1970.

[9] Smith, T. F. and Waterman, M.. Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195-197, 1981

[10] Wei L, Liao M, Gao X, Zou Q.. Enhanced Protein Fold Prediction Method through a Novel Feature Extraction Technique. *IEEE Trans Nanobiosci*, 14(6):649–659, 2015.

[11] Zhao X, Zou Q, Liu B, Liu X.. Exploratory predicting protein folding model with random forest and hybrid features. *Curr Proteomics*, 11(4):289–299, 2014.

[12] Asa Ben-Hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19(suppl 1):i26–i33, 2003.

[13] Huzefa Rangwala and George Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005.

[14] Leslie CS, Eskin E, Cohen A, Weston J, Noble WS. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[15] Håndstad T, Hestnes AJ, Sætrom P. Motif kernel generated by genetic programming improves remote homology and fold detection. *BMC Bioinform.*, 8(1):23, 2007.

[16] Saigo H, Vert JP, Ueda N, Akutsu T. Protein homology detection using string alignment kernels. *Bioinformatics.*, 20(11):1682–1689, 2204.

[17] Liao L, Noble WS. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *J Comput Biol.*, 10(6):857–868, 2003.

[18] Rui Kuang, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of bioinformatics and computational biology*, 3(03):527–550, 2005

[19] Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389-402, 1997.

[20] Hochreiter S, Heusel M, Obermayer K. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

[21] Shumin Li, Junjie Chen, Bin Liu. Protein remote homology detection based on bidirectional long short-term memory. *BMC Bioinformatics*, 18(1):86, 2017.

[22] Murzin AG, Brenner SE, Hubbard T, Chothia C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Mol Biol.*, 247(4):536-40, 1995.

[23] Suzek BE, Huang H, McGarvey P, Mazumder R, Wu CH. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[24] Peters, Matthew E. and Neumann, Mark and Iyyer, Mohit and Gardner, Matt and Clark, Christopher and Lee, Kenton and Zettlemoyer, Luke. Deep contextualized word representations. *Proc. of NAACL*, arXiv:1802.05365, 2017.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, 2018.

[26] Christina Leslie, Rui Kuang. Fast String Kernels using Inexact Matching for Protein Sequences. *Journal of Machine Learning Research 5*, 1435–1455, 2004.