Classification of Macromolecules Based on Amino Acid Sequences Using Deep Learning

Sarwar Khan
Department of Computer Science
National Chengchi University
Taipei, Taiwan
sarwar@iis.sinica.edu.tw

Imad Ali
Department of Computer Science
University of Swat
Khyber Pakhtunkhwa, Pakistan
imadali@uswat.edu.pk

Faisal Ghaffar
System Design Engineering Department
University of Waterloo
Waterloo, Canada
faisal.ghaffar@uwaterloo.ca

Qazi Mazhar-ul-Haq
National Taipei University
of Technology
Taipei, Taiwan
qazi@ntut.edu.tw

Received: 31 July 2022 | Revised: 26 August 2022 | Accepted: 28 August 2022

Abstract-The classification of amino acids and their sequence analysis plays a vital role in life sciences and is a challenging task. Deep learning models have well-established frameworks for solving a broad spectrum of complex learning problems compared to traditional machine learning techniques. This article uses and compares state-of-the-art deep learning models like Convolution Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) to solve macromolecule classification problems using amino acid sequences. The CNN extracts features from amino acid sequences, which are treated as vectors with the use of word embedding. These vectors are fed to the above-mentioned models to train robust classifiers. The results show that word2vec as embedding combined with VGG-16 performs better than LSTM and GRU. The proposed approach gets an error rate of 1.5%.

Keywords- CNN; LSTM; amino acid; macromolecules

I. INTRODUCTION

The last decade has witnessed the great success of deep learning as it has brought revolutionary advances in many application domains including computer vision, natural language processing, and signal processing. The key idea behind deep learning is the consideration of feature learning and classification in the same network architecture, using back-propagation to update model parameters and learn discriminative feature representations. Notably, many novel deep learning methods have been proposed, which improve classification performance significantly [1-3]. Studying deoxyribonucleic acid (DNA) in life sciences is essential for understanding organisms. Current sequencing technologies make it possible to read DNA sequences at a lower cost. DNA databases are increasing daily, and the power of modern computing is required. Classifying DNA sequences is a critical and essential task. Four major classes of organic macromolecules are always found in all life forms on Earth: carbohydrates, lipids (or fats), proteins, and nucleic acids. The significant macromolecule classes are similar in that they are large polymers assembled from small repeating monomer subunits. Proteins are large, complex molecules that play many critical roles in the body. They are made up of hundreds or thousands of smaller units called amino acids, which are attached in long chains. Twenty different types of amino acids can be combined to make a protein. The sequence of amino acids determines each protein's unique 3-dimensional structure and specific function. The interaction of protein with protein and protein with DNA/RNA (ribonucleic acid) plays a pivotal role in protein function. Experimental detection of residues in protein-protein interaction surfaces must come from determining the structure of protein-protein, protein-DNA, and protein-RNA complexes. However, experimental determination of such complexes lags far behind the number of known protein sequences. Hence, there is a need to develop reliable computational methods for identifying protein-protein, protein-RNA, and protein-DNA interface residues. Identifying macromolecules and detecting specific amino acid residues that contribute to the strength of interactions is a fundamental problem with broad applications ranging from rational drug design to the analysis of metabolic and signal transduction networks.

This article aims to utilize deep learning models to classify the macro molecule types based on the amino acid sequence and residue count. We used state-of-the-art deep learning models like CNN [4], LSTM [5], GRU) [6]. We used word embedding to represent the amino acid sequences as vectors. The CNN extracts features from amino acid sequences, which are treated as vectors. These vectors are then fed to the models mentioned above to train robust classifiers. Our results show that word2vec embedding combined with VGG-16 performs better than LSTM and GRU.

Corresponding authors: Sarwar Khan

II. LITERATURE REVIEW

The revolution in machine learning, mainly deep learning [7-10], allowed the study and extraction of complex patterns from data and made the machine models more robust. For instance, authors in [7] proposed deep learning techniques to perform brand and logo recognition tasks using multiple modern CNN models. In [8], a CNN model was applied to handwritten character recognition and was compared to a standard handwritten digit recognition task. It was found that CNNs efficiently deal with the variability of 2D shapes and outperform the other techniques. A variable size replicated CNN in [9] gathers relevant input information and eliminates irrelevant variabilities. A CNN model was proposed in [10] for the study and investigation of the accuracy and efficiency of new image datasets via transfer learning. Authors in [11] targeted learning as an informative feature representation of protein sequence as the input of neural network models to obtain the final predicting output of the belonging protein family. Authors in [12] proposed a framework with a deep 1D CNN, which is robust in both fold recognition and the study of sequence-structure relationships to classify protein sequences. Authors in [13] developed a framework with a CNN that used the idea of translation to convert DNA sequences to word sequences for final classification. Likewise, recurrent neural networks have shown promising results in many machine learning tasks. For example, an LSTM-based end-to-end approach was proposed in [14] for sequence learning, where the model makes nominal assumptions on the sequence structure. Similarly, authors in [15] described that recurrent neural networks could perform better on a challenging machine translation task. In the current article, we aim to evaluate the deep leaning techniques CNN, LSTM, and GRU for macromolecule classification. It is well established that these techniques work well on sequence-based tasks, even with long-term dependencies.

III. MATERIALS AND METHODS

A. Dataset

We used the protein data bank dataset [21]. A DNA molecule consists of two long polynucleotide chains composed of 4 types of nucleotide subunits, called a DNA chain. These may be adenine (A), cytosine (C), guanine (G), or thymine (T). The dataset contains two files with a different number of entries. One file has 467304 entries with 5 columns (Table I). The other file contains protein meta-data, i.e. resolution, extraction method, experimental technique, etc. with 141401 entries and 14 columns. We merged the two files based on structure IDs. The pre-processing step is to drop all the entries with NaN value or if a label or sequence is missing. After removing the missing values, the sequence is checked for the removal of tags and numbers. Once the dataset is cleaned, the sequence is divided into tri-gram, each sequence now combining three character strings. For example, CGC GAA TTC GCG. The final block may not have all three, and we added 0 to make it an equal slice (padding). The final output contains two columns, one for sequence and one for the label. As discussed above, there are 432474 rows in the processed data with 4 labels. These are all unbalanced data, and this issue will be discussed below.

TABLE I. TYPES OF MACROMOLECULE PROTEINS

Type	Label	Data structure	Entries
Structure ID	structureid	Object	140250
Chain ID	chainid	Object	2837
Sequence	protein sequence	Object	104813
Residue count	No. of residues ATCG's	Integer	4737
Macromolecule type	Macromolecule type	Object	14

B. Biological Structures in the Dataset

The central dogma of life if that DNA makes RNA, RNA makes amino acids, and amino acids make proteins. DNA and RNA are made of four types of nucleotides (A, T, C, G). The nucleotide sequence is the combination of these nucleotides in a row. Three nucleotides combine to form a codon, building a block of amino acids. The amino acids then are combined to form proteins. To make a protein, at least 20 amino acids are necessary. Consider the following real example. ATT is a codon, which consists of three nucleotides. This codon represents amino acid (isoleucine) represented by the letter "I." TTT is another codon, i.e. another amino acid (phenylalanine) and is characterized by the letter "F." These "IF"s are combined with others to make proteins. The letters in the codon represent nucleotides, while the letters in the protein sequence represent amino acids. At least 20 amino-acids must be combined to make a functional protein. The maximum number depends on when a stop codon is met. Such a codon can be met after 20 or after 500 amino acids. The amino acid sequence determines the type of proteins.

IV. THE PROPOSED METHOD

Figure 1 shows the block diagram of the proposed system. This system can be broadly divided into 3 tasks. The first 2 are dataset processing and embedding. We have different choices in embedding, but word2vec [16], Fast-Text [17], and GloVe [18] are well-known embedding techniques in Natural Language Processing (NLP). These embedding techniques have been tested in different bioinformatics tasks with promising results. Another famous embedding method is one-hot vector. We used word2vec in this study. The final task is the CNN, in which the number of network layers must be determined. We also need to find hyper-parameters along with the model's size, height, and width. The output layer is a SoftMax layer used to classify the sequence. Classifying macromolecule types using sequences can be seen as a sequence classification problem. This is analogous to the sentence classification task in NLP. Thus, we apply a skip-gram analysis from NLP research to model our problem. There are various sequence models in the deep learning domain. We used LSTM, GRU, and 1D convolution for our problem. Recurrent neural networks, such as LSTM, are specifically designed to support input data sequences. Figure 2 shows the layout of the model. CNNs can learn the complex dynamics within the temporal ordering of input sequences and use internal memory to remember or use information across long input sequences. As it is crucial to any deep learning task, we applied data prepossessing before feeding them to our model. Prepossessing includes handling missing values and downsampling the dominant class to balance data distribution.

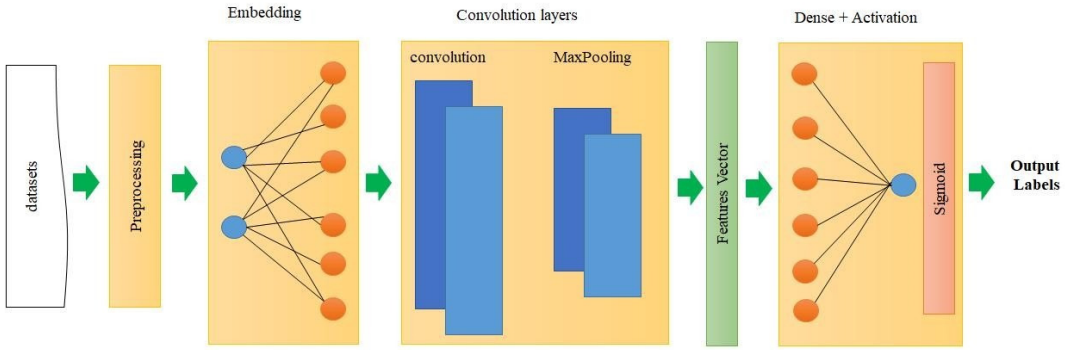

Fig. 1. The blog diagram of the proposed method.

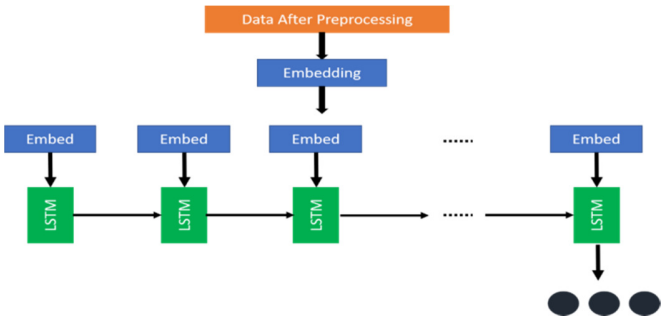

Fig. 2. LSTM model using embedding at the input.

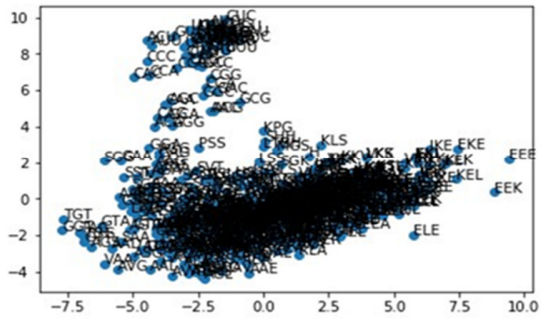

Fig. 3. word2vec model relation between sequences.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 128, 50)	1150
conv1d_1 (Conv1D)	(None, 128, 64)	19264
max_pooling1d_1 (MaxPooling1	(None, 64, 64)	0
conv1d_2 (Conv1D)	(None, 64, 128)	24704
max_pooling1d_2 (MaxPooling1	(None, 32, 128)	0
conv1d_3 (Conv1D)	(None, 32, 128)	49280
max_pooling1d_3 (MaxPooling1	(None, 16, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 1024)	2098176
dropout_1 (Dropout)	(None, 1024)	0
batch_normalization_1 (Batch	(None, 1024)	4096
dense_2 (Dense)	(None, 512)	524800
dropout_2 (Dropout)	(None, 512)	0
batch_normalization_2 (Batch	(None, 512)	2048
dense_3 (Dense)	(None, 4)	2052

Total params: 2,725,570
Trainable params: 2,722,498
Non-trainable params: 3,072

Fig. 4. CNN model parameters with dimensions.

A. *Word Embedding*

We used word2vec as the embedding technique in this work [17]. We need embedding techniques because deep learning or machine learning models only deal with real numbers. Embedding not only converts these texts or sequences into numbers but also produces relationships between them. We have two algorithms for word2vec, skip-gram and Common Bag Of Words (CBOW). After using both these algorithms in the 3 models, we find that skip-gram suits better in our case. We used the tri-gram data to generate our word2vec model. Figure 3 shows the relationship between the sequences. We used different output dimensions, i.e. 100, 150, and 300. We find that the 300-dimension output has better performance.

B. *Convolution Neural Network*

The CNN is the most famous deep learning method. We used the network architecture of VGG [19, 20].

We are using Convolution1D as the data dimension is 1D. This network has 4 convolution layers, each layer followed by a max-pooling layer. The network also includes batch normalization and dropout to prevent the model from over-fitting. Two dense layers follow the final max-pooling layer. The hyper-parameters which give the best results were set as: learning rate: 0.001, batch size: 512, loss function: cross-entropy, optimizer: Adam, number of epochs: 20, dropout rate: 0.5, activation: ReLU. The final layer is SoftMax. Figure 4 shows the model architecture along with the model parameters.

V. EXPERIMENTS AND RESULTS

We evaluated our models on a protein data bank [21], a database for the three-dimensional structural data of large biological molecules, such as proteins and nucleic acids. Performance comparison between different models will only make sense if we keep the same pre-processing and embedding. Our experiments show that word2vec with output dimension of

300 performs better, and we will keep this setting unless mentioned otherwise.

A. CNN Model Results

Figure 5 shows the validation and training loss over the 50 epochs with an embedding dimension of 50. We used an early stopping algorithm to save and use our best model. Training loss is 0.028, while validation loss is 0.034.

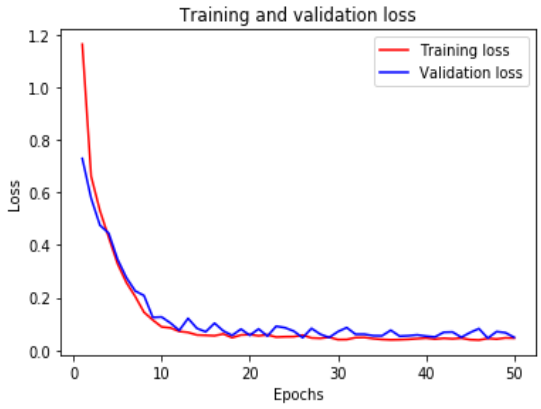

Fig. 5. Training and validation loss.

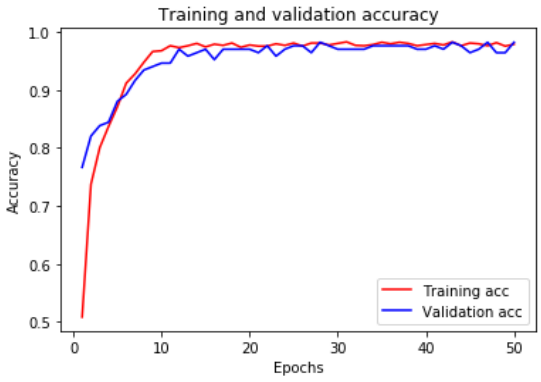

Fig. 6. Model training and validation accuracy.

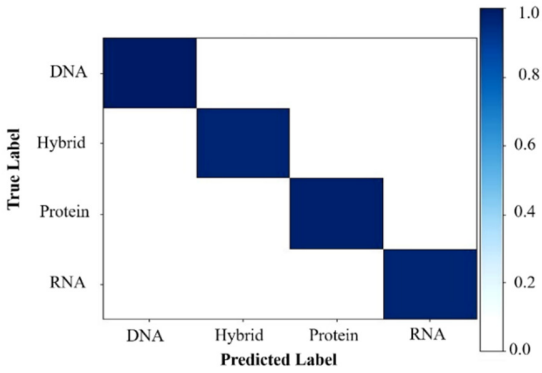

Fig. 7. Confusion matrix for all classes.

Figure 6 shows the accuracy curve for the same setting, where the final training accuracy is 99.2% and the test accuracy is 98.8%. The micro-average and macro-average are almost the same. To further dig out the details, we drew the confusion matrix of all four classes, as shown in Figure 7, where the accuracy of each class is nearly the same, as we can see from the diagonal color. Figure 8 shows the CNN model's Precision, Recall, and F1-Score. The model has precision of 0.98, 0.97, 0.98, and 1.00 for classes of DNA, Hybrid, RNA, and Protein respectively. Recall and F1-Score can also be easily observed in Figure 8.

	precision	recall	f1-score	support
DNA	0.98	1.00	0.99	49
Hybrid	0.97	0.97	0.97	35
Protein	0.98	0.98	0.98	47
RNA	1.00	0.97	0.99	36
micro avg	0.98	0.98	0.98	167
macro avg	0.98	0.98	0.98	167
weighted avg	0.98	0.98	0.98	167

Fig. 8. Precision, Recall, and F1-Score.

B. Additional Simulations

One hundred estimators initialized Random Forest (RF) [22], showing 96.83% test accuracy for more than 90,000 test sequences. We tested the RF for balanced and unbalanced data, and the results were almost identical. Table II shows each class's Precision, Recall, and F1-Score. The results are not good enough compared to CNN, but RF is much faster and less expensive than CNNs. We are reporting these results to show that traditional machine learning algorithms also work better for some visual question answering times.

TABLE II. RANDOM FOREST RESULTS

Label	Precision	Recall	F1-Score
Protein	0.96	0.99	0.97
DNA	0.90	0.80	0.85
RNA	0.93	0.69	0.79
Hybrid	0.94	0.88	0.91
Macro-avg.	0.93	.84	0.88
Weighted-avg	0.96	0.96	0.96

TABLE III. PERFORMANCE OF DIFFERENT NETWORKS

Model	T. Acc.	T. Loss	Val-Acc.	Val-Loss
CNN	98.19%	0.0486	97.74%	0.0819
GRU	90.79%	0.2691	89.70%	0.2715
LSTM	95.12%	0.3982	95.14%	0.1962
CNN-GRU	94.85%	0.1509	92.74%	0.1962
RF	95.17%	0.4019	94.87%	0.4906
VGG16	99.11%	0.0288	98.10%	0.0297

The next step is to compare different state-of-the-art algorithms like LSTM, RF, and GRU with the proposed CNN. We used TensorFlow embedding in this case with 50-dimension vectors. In this case, the LSTM and GRU have the same network structure and use the same number of cells, i.e. 512. Table III shows the comparison of accuracy and loss across the different networks. The Table shows that VGG-16 [19] performs better with word2vec as embedding. For example, the training accuracy (T. Acc.) and the validation accuracy (Val. Acc.) for the VGG-16 are 99.11 and 98.10

respectively. Moreover, the training loss (T. Loss) and the validation loss (Val. Loss) for the VGG-16 are 0.0288 and 0.0297 respectively. The 7-layer CNN also performs better compared to LSTM and GRU. All CNNs are of one dimension. The other models' results can be easily read in Table III.

VI. CONCLUSION

In this article, we used NLP techniques to classify the protein sequences and very good results based on accuracy and precision were achieved. The CNN and CNN-FRU models have better performance than the other models. Although we didn't train the models for long epochs, we still obtained high accuracy with CNN. Moreover, the data are highly non-normalized. These models perform much better when the data are normalized. Future work includes classifying the data with more models.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, May 2016, pp. 770-778, https://doi.org/10.1109/CVPR.2016.90.

[2] C.-L. Liu, Hsaio W.-H., and Tu Y.-C., "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788-4797, Aug. 2018, https://doi.org/10.1109/TIE.2018.2864702.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826, https://doi.org/10.1109/CVPR.2016.308.

[4] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The handbook of brain theory and neural networks*, Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, https://doi.org/10.1162/neco.1997.9.8.1735.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." arXiv, Dec. 11, 2014, https://doi.org/10.48550/arXiv.1412.3555.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, Feb. 2017, https://doi.org/10.1145/3065386.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Aug. 1998, https://doi.org/10.1109/5.726791.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, https://doi.org/10.1038/nature14539.

[10] M. Hussain, J. J. Bird, and D. R. Faria, "A Study on CNN Transfer Learning for Image Classification," in *Advances in Computational Intelligence Systems*, 2019, pp. 191–202, https://doi.org/10.1007/978-3-319-97982-3_16.

[11] T. K. Lee and T. Nguyen, "Protein Family Classification with Neural Networks," [Online]. Available: https://cs224d.stanford.edu/reports/Lee Nguyen.pdf.

[12] J. Hou, B. Adhikari, and J. Cheng, "DeepSF: deep convolutional neural network for mapping protein sequences to folds," *Bioinformatics*, vol. 34, no. 8, pp. 1295–1303, Apr. 2018, https://doi.org/10.1093/bioinformatics/btx780.

[13] N. G. Nguyen *et al.*, "DNA Sequence Classification by Convolutional Neural Network," *Journal of Biomedical Science and Engineering*, vol. 9, no. 5, pp. 280–286, Apr. 2016, https://doi.org/10.4236/jbise.2016.95021.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks." arXiv, Dec. 14, 2014, https://doi.org/10.48550/arXiv.1409.3215.

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate." arXiv, May 19, 2016, https://doi.org/10.48550/arXiv.1409.0473.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, Red Hook, NY, USA, Sep. 2013, pp. 3111–3119.

[17] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: Compressing text classification models." arXiv, Dec. 12, 2016, https://doi.org/10.48550/arXiv.1612.03651.

[18] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Jul. 2014, pp. 1532–1543, https://doi.org/10.3115/v1/D14-1162.

[19] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv, Apr. 10, 2015, https://doi.org/10.48550/arXiv.1409.1556.

[20] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot Ensembles: Train 1, get M for free." arXiv, Mar. 31, 2017, https://doi.org/10.48550/arXiv.1704.00109.

[21] R. P. D. Bank, "RCSB PDB: Homepage," *Protein Data Bank*. https://www.rcsb.org/.

[22] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, https://doi.org/10.1023/A:1010933404324.