# Analysis of Artificial Intelligence Methods for Predicting Protein Function Based on Primary Structure

**Dhruva Cheethirala**

*Received July 28, 2024*
*Accepted September 23, 2024*
*Electronic access October 15, 2024*

The *grand challenge* of biology for the last 50 years has been to find a method to reliably predict the way proteins fold. Mapping the folded (or tertiary) structure of proteins based on their primary structure allows us to understand their function, a critical task. While artificial intelligence tools to reliably predict the tertiary structure exist, predicting the function of the protein from its primary structure is not as well explored. In this paper, we explore multiple artificial intelligence approaches to predicting protein function, including a deep learning approach and using a Long Short Term Memory (LSTM) approach. We used a CountVectorizer approach to process the sequence data for the deep learning model. We assign each amino acid a number when processing the data for the LSTM. Our deep learning model provided a maximum accuracy of 0.6091 and the LSTM model provided a maximum accuracy of 0.7415. Our results show us that there is potential for stronger results when predicting protein function, especially if we further explore other model types.

**Keywords:** Proteins, Protein folding, Machine learning, Artificial Intelligence, Deep learning, Recurrent neural networks, Long short term memory, Computational Biology, Bioinformatics.

## Introduction

For the last 50 years, accurate prediction of how amino acid chains fold into proteins has been a *grand challenge* of biology [1]. This is because modeling the folded, or tertiary, structure of proteins offers valuable insights into the function of proteins [2]. In particular, the tertiary structure of proteins is essential knowledge for the drug development industry, as the tertiary structure informs researchers of the function of proteins that are present in antibiotics [3]. Numerous traditional methods have therefore been developed for experimentally determining the tertiary structure, including X-Ray Crystallography, Nuclear Magnetic Resonance (NMR), Cryogenic Electron Microscopy, and Dual Polarisation Interferometry [3]. However, each of these methods has its drawbacks, primarily that they are expensive and time-intensive processes [4].

Contrarily, the primary structure, a chain of amino acids that make up the protein, is much simpler to find through experimental methods such as mass spectrometry. While the primary structure determines the tertiary structure [5], accurately predicting how this chain of amino acids folds into a 3D structure remains a major problem in modern biology. As such, artificial intelligence has been consistently applied to attempt to solve this problem. In 2020, Google's research subsidiary DeepMind introduced AlphaFold. AlphaFold is a novel deep learning model that takes the primary structure of a protein as an input, and outputs the tertiary structure with high levels of accuracy [6]. AlphaFold utilizes attention-based architecture first introduced by Google in 2017, through the paper "Attention Is All You Need" [7], which enables its high accuracy by being able to analyze long sequences of data with less loss of essential information.

AlphaFold2 (the second generation of the model) performed incredibly in the 14th Critical Assessment of protein Structure Prediction (CASP14), receiving a median GDT (Global Distance Test) score of 92.4 [8]. CASP is considered the gold standard competition for computational biology methods that predict the tertiary structure of proteins. John Mault, a co-founder of CASP, considered AlphaFold2's GDT score to be comparable to experimental methods without the tedious and expensive lab analysis required [9]. In 2022, DeepMind announced that AlphaFold had predicted the structures to nearly all proteins known to science: over 214 million structures [10,11]. DeepMind claims that its partners are already using AlphaFold to solve real-world problems including breaking down single-use plastics and finding new drugs to treat liver cancer [12].

Following the reconstruction of the tertiary structure of a protein, artificial intelligence has also been applied in regards to predicting protein function from structure. DeepFRI, a Graph Convolutional Network, predicts protein function using the tertiary structure as an input. As a pre-trained, task-agnostic, language model, DeepFRI is able to reliably predict function, even when the input is a structure that is computationally inferred. At the time of its release in 2021, DeepFRI was outperforming many state-of-the-art models [13].

However, the exciting advancements of combining methods such as AlphaFold and DeepFRI raise questions on how else AI can be applied to this *grand challenge* of biology. The ultimate goal of predicting the tertiary structure is to understand the function of the protein. This fact encourages us to challenge whether or not we can skip the intermediate step of the tertiary structure to directly and reliably predict the function of a protein using the primary structure as an input without first determining the tertiary structure. While it is clear that function prediction directly from the tertiary structure will likely be more accurate, it is computationally costlier and less accessible. Thus, it is critical to explore the capabilities of predicting protein function from primary structure.

Predicting protein function directly from primary structure, skipping the intermediate step of finding the tertiary structure, is not as well explored as predicting tertiary structure from primary structure. There has been some work using transformer/attention-based architecture to predict protein function from primary structure. For example, ReLSO, a transformer-based model, can predict "fitness" for a protein sequence, which represents its general functionality/foldability [14]. This model doesn't, however, classify proteins into specific types of functions. In this paper, we therefore examine different model architectures in an attempt to predict the function of the protein resulting from an amino acid chain without predicting the tertiary structure. Specifically, we study two different approaches that consider the primary structure of a protein, the sequence of amino acids. The first approach studied, a standard Neural Network considers the protein input as a collection of amino acids without taking the order of them into account, as the Bag of Words representation of the protein sequences is based on amino acid frequency, and not order. The second method, based on a Long-Short Term Model, does take the order into account. Our paper thus aims to answer the question; to what extent is the primary structure of a protein sufficient to determine its function? As our result will show, taking the sequence of the amino acids into account is important for accurate protein function prediction.

Our paper is organized as follows. First, we will introduce the data and different machine learning models used. Following this, we will present the accuracy each of our methods was able to achieve on the used datasets. Lastly, we will discuss the implications of these results and consider possible limitations of our work.

## Methodology

In this section we will provide an overview of the two different machine learning methods evaluated for predicting protein function using only the primary structures as well as the data used to evaluate the models.

### Data

The data we use for this paper was taken from Kaggle. The dataset, named Structural Protein Sequences, contains the primary structure of numerous proteins and a classification of their function as a protein [15]. The primary structures were taken from the Protein Data Bank, where accurate experimental methods such as X-ray Crystallography, NMR techniques, and cryo-electron microscopy were used to find the sequences [16]. There was no metadata in the dataset, besides a protein ID that is irrelevant for this paper. After filtering out null values, the dataset has 346,321 amino acid chain and classification pairs. There are 4468 different classifications, but many of these contain only one or just a few corresponding amino acid chains. We filter out any classifications that don't have at least 100 entries, as they are nonoptimal for model training. This is to prevent having a significant class imbalance, as functions with just a few corresponding proteins would be ignored by the model. Given the size of the dataset, 100 is a reasonable amount of proteins to require for each classification to be represented by the model. Additionally, we specifically remove any classifications that aren't directly describing a type of protein function. We do this by only using classifications ending in the word 'PROTEIN', which tells us that they are labeling a specific protein that carries out a unique function. For this dataset, classifications ending in protein specifically describe a protein with a function. No relevant classifications were ignored. This leaves us with 35 classifications. These are: ['CALCIUM-BINDING PROTEIN', 'VIRAL PROTEIN', 'GLYCOPROTEIN', 'RIBOSOMAL PROTEIN', 'LIPID BINDING PROTEIN', 'DNA BINDING PROTEIN', 'MEMBRANE PROTEIN', 'CONTRACTILE PROTEIN', 'STRUCTURAL PROTEIN', 'RNA BINDING PROTEIN', 'FLAVOPROTEIN', 'ANTIVIRAL PROTEIN', 'SIGNALING PROTEIN,' 'BIOTIN-BINDING PROTEIN', 'PEPTIDE BINDING PROTEIN', 'SUGAR BINDING PROTEIN', 'METAL BINDING PROTEIN', 'TRANSPORT PROTEIN', 'PLANT PROTEIN', 'LUMINESCENT PROTEIN', 'MOTOR PROTEIN', 'DE NOVO PROTEIN', 'ANTIMICROBIAL PROTEIN', 'ANTITUMOR PROTEIN', 'VIRUS/VIRAL PROTEIN', 'LIGAND BINDING PROTEIN', 'FLUORESCENT PROTEIN', 'BIOSYNTHETIC PROTEIN', 'NUCLEAR PROTEIN', 'CIRCADIAN CLOCK PROTEIN', 'BIOTIN BINDING PROTEIN', 'IMMUNE SYSTEM/VIRAL PROTEIN', 'MEMBRANE PROTEIN, TRANSPORT PROTEIN', 'HYDROLASE/TRANSPORT PROTEIN', 'ACETYLCHOLINE-BINDING PROTEIN']

In total, we have 56396 records within these 35 classifications. The data is stored in multiple ways depending on the type of model. We primarily use PyTorch tensors to store and process the data.

Some limitations of the dataset are that we filter out everything but 35 distinct classifications, a potential point of bias.

This means practically, we could only use our model on a primary structure if we know that it is within one of these classifications. A more useful approach would be to have data that covers many different types of proteins in order to have a more flexible model. This is something that can be expanded on with access to more broad data in the future.

### Methods

We utilize two primary architectures to create our models. First, a simple deep learning model utilizing linear layers to process the chain as a whole, using count-based vectors to process the inputs. Second, a Long Short-Term Memory (LSTM) model that applies Recurrent Neural Networks (RNNs) to recursively process each acid in the chain. This way, we can examine the differences in accuracy from processing each acid individually and processing the chain as a whole. Both models were implemented using the PyTorch framework [17] and were run using a T4 GPU on Google Colab. All code used for the experiments is available online.

### Deep Neural Network

Our first model is a standard Deep Neural Network (DNN) with fully connected layers [18–20]. We process our input, which is initially a string with a character representing each acid in the chain, using sklearn's CountVectorizer. This Bag of Words representation provides us with a count-based representation of the string in the form of a vector. We input this vector into a PyTorch model, with varying numbers of linear layers of varying sizes which we discuss further below. We use an optimizer with a varying learning rate. Additionally, we use a DataLoader to process the large amounts of input data in chunks of 64 entries. The Adam optimizer is used for this model.

One of the main hyperparameters for deep neural networks is the number of layers used and the number of nodes for each layer. In the final iteration of our deep neural network, we have eight layers total. Our input layer is of size 24, reflecting the dimensions of our input matrix. Our next 7 layers are of sizes 100, 200, 250, 150, 125, 75, and 50 in that order. Our final output layer is of size 35, to match the 35 different classifications our model can output. Each linear layer is followed by a ReLU layer, except the output layer. After many variations of the model, the iteration with these arguments showed the best results.

We generally run the model for 250 epochs, and decrease the learning rate after 50 epochs of a higher learning rate. We save the loss and accuracy for testing and training data after each epoch. We switched to the lower learning rate to take advantage of less overfitting over more epochs while still getting the initial growth of a high learning rate over the first 50 epochs. This strategy proved to show higher accuracy and lower loss.

The model runs for 250 epochs. For the first 50 epochs, the model trains at a higher learning rate of 0.0005. After 50 epochs, there is a scheduled decrease in learning rate to 0.0001 for the next 200 epochs. This strategy allows the model to make larger optimizations in the first 50 epochs, while reducing the impact of overfitting over the next 200. By taking smaller steps when overfitting generally occurs, scheduling a learning rate decrease helps mitigate its impact [21]. This strategy proved to show higher accuracy and lower loss.

### Long-Short Term Memory Model

Our second model is a more complex Long-Short Term Memory (LSTM) approach [20,22,23]. In order to process our input, each chain must be stored as a list, and each character must occupy one index in the list. This is because we need to convert each acid into an integer from 1-26 in order to process the data, and using a float or similar datatype to store the number would lead to the value being processed as "infinity" due to the restrictions of the float datatype and the large length of amino acid chains. Due to computing constraints, we only process the chains with 32 acids or less. While there is a concern of this leading to potential data loss, this wasn't observed in practice. If the chain is less than 32 in length, we add 0s to the end of its list until it is length 32 for smoother processing. Because the padding is a different token than the rest of the amino acid chain, the model learned to ignore these parts of the data. We store all these lists in a tensor, and when we feed this data into our model, we use a data loader with varying batch size to process the data.

In the final iteration of our second model, we have an embedding dimension of 32, a hidden dimension of 128, and 7 hidden layers. Additionally, our data loader uses batch sizes of 64 and the optimizer has a learning rate of 0.0001 (after switching from 0.0005 after 50 epochs). We run the model for 2000 epochs.

## Results

In the following we will discuss the results of our experiments with the two different deep learning architectures. For each model we will discuss the final accuracy and present the training curves.

### Deep Neural Network

Using a deep neural network for predicting the function of a protein purely on the set of amino acids the protein is made from, we find moderate results. As visualized in Fig. 2, after 250 epochs we find high final accuracy on the training data and lower accuracies for testing data. This phenomenon is called extreme overfitting and is a common issue for this type of architecture. In our Discussion we will discuss different approaches for overcoming this. Interestingly, our testing accuracy actually decreases shortly after switching to the lower learning rate of

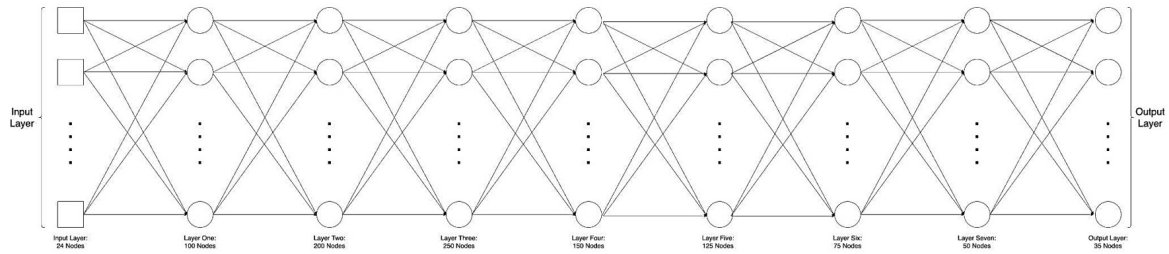


**Fig. 1** Diagram shows the structure of the deep learning model. Each layer is shown with the correct number of nodes written below.

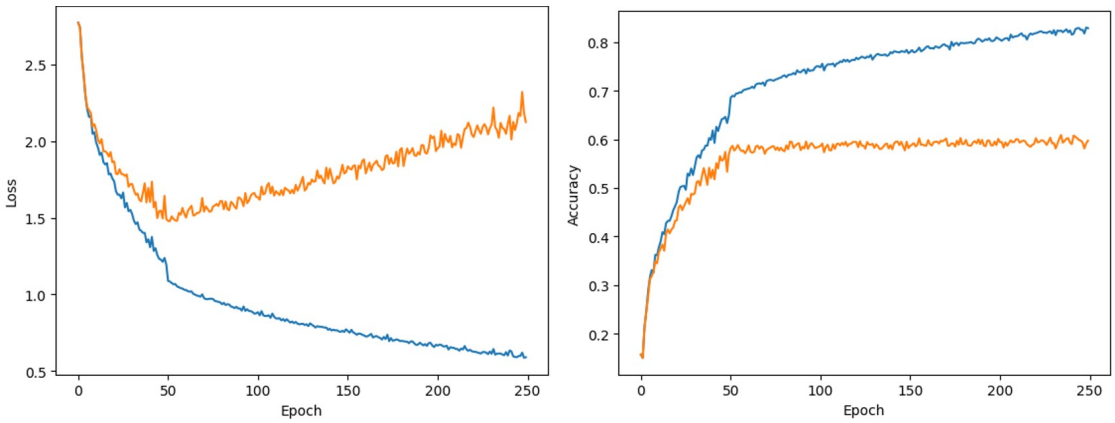


**Fig. 2** Plots of loss (left) and accuracy (right) vs. the number of epochs of training of our DNN. In both plots the blue line represents the training data while the orange represents the test data.

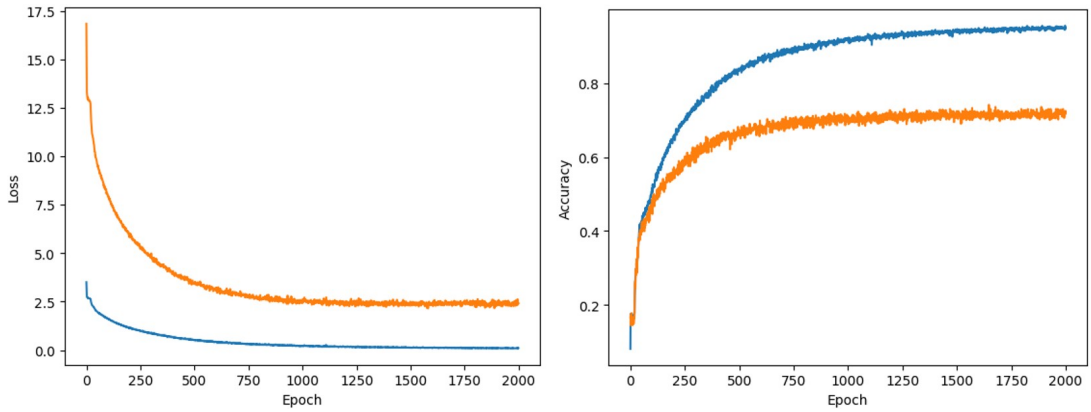


**Fig. 3** Plots of loss (left) and accuracy (right) vs. the number of epochs of training of our LSTM. In both plots the blue line represents the training data while the orange represents the test data.

0.0001 from 0.0005. Regardless, our model performs decently well and provides us with a maximum accuracy of 0.6091.

### Long-Short Term Memory

When including the sequenced data in addition to the set of amino acids, the second approach, a Long-Short Term Memory model, shows stronger results. After 2000 epochs, our model has a better maximum test accuracy. Additionally, while we once again see higher training than testing accuracies, indicating overfitting the overfitting doesn't appear as significant as in the previous model. By using larger datasets and dropout techniques, we can mitigate the overfitting. These techniques are explained in more detail in the Discussion. The maximum test accuracy is 0.7415.

## Conclusion

In this work we have studied the difference in performance between two different machine learning models for predicting protein function without taking into account the tertiary structure. Specifically, we have compared standard fully connected Deep Neural Networks with Long-Short Term Memory models. These two approaches primarily differ by how they take the amino acid sequence into account.

Notably we found that Long-Short Term Memory models significantly outperform Deep Neural Networks in terms of accuracy, with a test accuracy of 0.7415 (LSTM) compared to 0.6091 (Deep Neural Network). This result is in line with what we would expect as Long-Short Term Memory models retain more information than Deep Neural Networks, and have information about the order of the amino acid sequence. Since protein sequences are sequential forms of data, LSTMs are geared towards solving these problems as they can process sequential data recursively and utilize useful information for their predictions from previous parts of the sequence. As such, for future research into predicting protein function without considering the tertiary structure we recommend the use of recurrent neural architectures such as the LSTM model.

### Limitations and Future Work

Due to computational restrictions a number of limitations were placed on the study that we believe would be important to lift in future work. Most notably, we looked at a smaller dataset and only utilized 35 classifications for functions. For more practical models, we would need to use a much larger dataset with many more classifications to truly be able to understand the intricacies of different chains and how that impacts their function. Additionally, only using 35 classifications means numerous rarer types of proteins are disregarded in favor of more common protein types. Moreover, many more epochs and smaller learning rates for our model might have provided stronger accuracies/results.

Additionally, certain amino acids have inherent similarities[24]. This could skew our results and lead to inconsistencies when our models handle these similar acids. Examining how our models approach these cases could allow us to improve the models' performance.

Overfitting was observed in our model, with test and training accuracy and loss diverging eventually, as seen in Figures 2 and 3. We can mitigate the impact of overfitting with access to larger datasets. Another potential method for future approaches is to use a dropout technique, where random neurons in the neural network are deactivated in training to reduce dependence on specific neurons[25].

Furthermore, we believe that our results can be further improved by considering different architectures of the models studied here (eg. more layers and nodes) as well as using different architectures all together. For example, the attention-based models as mentioned in the introduction could possibly provide an improvement over the LSTM architecture. A more thorough examination of other models could provide more insight into the feasibility of predicting protein function from primary structure.

## References


1 Google DeepMind, *AlphaFold: A Solution to a 50-year-old Grand Challenge in Biology*, 2020, https://deepmind.google/discover/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology, Accessed: 2024-07-26.

2 A. Davis, *Health Tech Zone*, 2019.

3 Creative Biolabs, *Tertiary Structure Analysis*, 2024, https://www.creative-biolabs.com/drug-discovery/therapeutics/tertiary-structure-analysis.htm, Accessed: 2024-07-26.

4 SITNFlash, *Science in the News*, 2021.

5 I. Rehman, *Biochemistry, Tertiary Protein Structure*, 2022, https://www.ncbi.nlm.nih.gov/books/NBK470269, Accessed: 2024-07-26.

6 J. Jumper and et al., *Nature*, 2021, **596**, 583–589.

7 A. Vaswani and et al., *arXiv*, 2019, **30**, 5998–6008.

8 A. P. S. Database, *AlphaFold Protein Structure Database*, https://alphafold.ebi.ac.uk/faq, Accessed: 2024-07-26.

9 A. Al-Janabi, *BioTechniques*, 2022, **72**, 73–76.

10 S. Goldman, *VentureBeat*, 2023.

11 M. Varadi and et al., *Nucleic Acids Research*, 2023, **52**, D368–D375.

12 Google DeepMind, *AlphaFold*, 2022, https://deepmind.google/technologies/alphafold, Accessed: 2024-07-26.

13 V. Gligorijević and et al., *Nature Communications*, 2021, **12**, year.

14 E. Castro and et al., *Nature Machine Intelligence*, 2022, **4**, 840–851.

15 Kaggle, *Structural Protein Sequences*, 2018, https://www.kaggle.com/datasets/shahir/protein-data-set, Accessed: 2024-07-26.

16 K. F. Aoki-Kinoshita, *Advances in Glycomics*, Elsevier eBooks, 2023, pp. 516–524.

17 A. Paszke and et al., *arXiv.org*, 2019.

18 *Colab Research Google*, https://colab.research.google.com/drive/1NVsegTF4lw2W7MgC0hJFP-UPoeN61dX4?usp=sharing, Accessed: 2024-07-26.

19 A. Tam, *Machine Learning Mastery*, 2023.

20 Danofer, *Predicting Protein Classification*, 2018, https://www.kaggle.com/code/danofer/predicting-protein-classification, Accessed: 2024-07-26.

21 K. Li, *neptune.ai*, 2023.

22 *Colab Research Google*, https://colab.research.google.com/drive/1ZnTH67IhQHfeiWMEvD6Eo5CPY4r4wlct?usp=sharing, Accessed: 2024-09-07.

23 A. Tam, *Machine Learning Mastery*, 2023.

24 J. Stephenson and S. J. Freeland, *Journal of Molecular Evolution*, 2013, **77**, 159–169.

25 A. Jain, *Medium*, 2024.