Deep Learning for Protein Secondary Structure Prediction: An RNN Approach

Ana CarolinaMorais[1] and Fernanda Fernandes[1]

Department of Informatics Engineering, University of Coimbra
`anamorais@student.dei.uc.pt, mrfernandes@student.dei.uc.pt`

Abstract. Prediction of protein secondary structure is a fundamental problem in computational biology with important applications in drug design, disease understanding, and bioinformatics. This paper explores the use of deep learning techniques, specifically Recurrent Neural Networks (RNNs), to predict the secondary structure of proteins based on their amino acid sequences. We propose an approach that leverages the sequential nature of protein sequences and applies one-hot encoding for amino acids, followed by padding and label encoding. The model is trained on a dataset of protein sequences with the corresponding secondary structure labels, and its performance is evaluated through various metrics such as precision and confusion matrices. The results show that the RNN model is capable of learning complex patterns in protein sequences and making accurate predictions of secondary structures. This work demonstrates the potential of deep learning methods in advancing the field of protein structure prediction.

Keywords: Protein Secondary Structure · Deep Learning · Recurrent Neural Networks · Protein Sequence Prediction · Machine Learning.

1 Introduction

Predicting protein secondary structure from amino acid sequence is a fundamental problem in computational biology and bioinformatics. The secondary structure of proteins refers to the local conformations formed within the polypeptide chain due to hydrogen bonds between the atoms of the backbone [Ismi et al., 2022]. These local structures primarily include α, β, and loops. Understanding the secondary structure is a crucial step toward deciphering the overall folding patterns of proteins, which in turn influence their function and properties [Szelogowski, 2023].

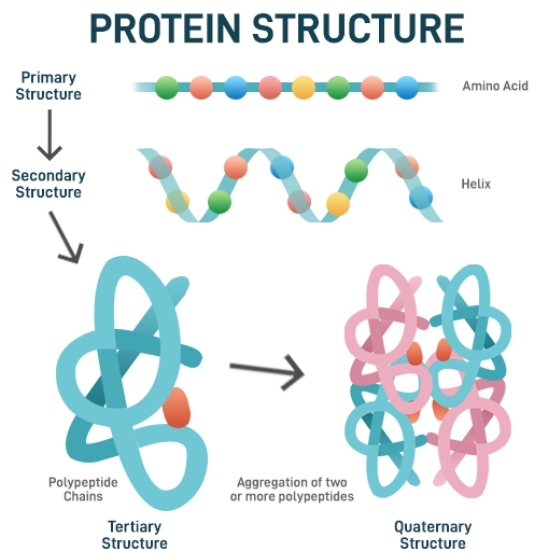

Fig. 1. Overview of protein structure: primary, secondary, tertiary, and quaternary levels.[2]

As illustrated in Figure 1, proteins are organized into four structural levels. This study focuses exclusively on the **secondary structure**, which serves as the foundation for higher-order folding and biological function. The accurate prediction of these structures enables deeper insights into protein behavior and interactions.

Since the inception of Protein Secondary Structure Prediction (PSSP) in the 1960s, five generations of methods have progressively advanced the field. Early approaches relied on statistical properties of amino acids, while subsequent generations incorporated evolutionary data, feature engineering, and, most recently, **deep learning** techniques. Architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Graph Neural Networks (GNNs) have transformed the landscape of PSSP, achieving unprecedented accuracy, surpassing 80% for three-state classification tasks [Ismi et al., 2022].

Despite these advances, challenges persist, particularly in capturing long-range dependencies within amino acid sequences and handling complex structural variations. Moreover, breakthroughs such as AlphaFold2 have revolutionized the field of structural biology by predicting full 3D protein structures [Jumper et al., 2021]. However, specialized tools are still necessary for high-precision secondary structure prediction, which remains a critical task for applications like drug design and protein engineering.

[2] https://www.geeksforgeeks.org/protein-structure-primary-secondary-tertiary-quaternary/

This article explores a deep learning approach using Recurrent Neural Networks for predicting protein secondary structure from sequence data. We discuss the challenges, detail the model architecture, and compare our results with existing state-of-the-art methods, aiming to advance the accuracy and efficiency of protein secondary structure prediction.

2 Related Work

Protein secondary structure prediction has been an active field of study for decades. Early approaches, such as those of [Chou and Fasman, 1974], relied on statistical methods to predict protein conformations based on amino acid propensity. The introduction of neural networks in the 1980s, as demonstrated by [Holley and Karplus, 1989], marked a significant improvement. They applied a simple feed-forward neural network to predict secondary structures, achieving an accuracy of 63% for helix, strand, and coil prediction. This work highlighted the potential of neural networks, though accuracy was limited by computational resources and data availability at the time.

Subsequent advancements incorporated sequence profiles generated through PSI-BLAST and position-specific scoring matrices (PSSM). Methods such as [Rost and Sander, 1993] model improved accuracy significantly by integrating evolutionary information. Modern methods leverage deep learning architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs). For instance, [Li and Yu, 2016] introduced cascaded CNN-RNN models, achieving notable improvements. Similarly, [Lyu et al., 2021] proposed the MLPRNN model, a reductive deep-learning architecture combining bidirectional gated recurrent units (BGRUs) and multi-layer perceptrons (MLPs). Their method achieved 83.32% Q3 accuracy and 70.59% Q8 accuracy on benchmark datasets.

Our proposed work builds upon these foundations by further exploring the use of recurrent neural networks (RNNs) to predict secondary protein structures. Unlike traditional methods that depend heavily on hand-crafted features such as PSSM or HMM profiles, our approach emphasizes one-hot encoding and sequence padding to process raw amino acid sequences. By evaluating the RNN model's ability to learn sequential patterns, we achieve comparable performance with modern state-of-the-art techniques. Furthermore, our model's evaluation includes metrics such as precision and confusion matrices, ensuring a robust performance assessment.

3 Materials

3.1 Dataset

In this study, we used datasets[3] for protein secondary structure prediction, including the training and test sets. The files were loaded using the following paths in the code:

[3] https://www.kaggle.com/datasets/tamzidhasan/protein-secondary-structure-casp12-cb513-ts115/data

```
train_path = "Dataset\\training_secondary_structure_train.csv"
test_path = "Dataset\\test_secondary_structure_cb513.csv"
```

The **training dataset** serves as a primary input for model development, enabling the model to learn patterns within protein sequences and their corresponding secondary structures. The **test dataset** is used to evaluate the model's generalization capabilities on unseen data, ensuring that the trained model is able to predict secondary structures accurately.

By utilizing distinct training and testing datasets, we mitigate the possibility of overfitting and validate the model's capability to perform in real-world scenarios. These datasets hold significant importance in evaluating the robustness and accuracy of our deep learning approach for the prediction of secondary structures.

3.2 Dataset Description

The datasets contain sequences of amino acids annotated with their corresponding secondary structure classes:

- **seq**: The primary amino acid sequence of a peptide.
- **sst3**: The three-state (Q3) secondary structure classification.
- **sst8**: The eight-state (Q8) secondary structure classification.

Three-State (SST3) Classification: For this study, we used the three-state classification, which simplifies the secondary structure into three broad categories, making it a good starting point for model development:

- **H**: Helix
- **E**: Strand
- **C**: Coil

Eight-State (SST8) Classification: For a more detailed secondary structure classification, the SST8 states provide additional granularity:

- **H**: α-helix
- **B**: β-bridge
- **E**: β-strand
- **G**: 3-helix
- **I**: π-helix
- **T**: Turn
- **S**: Bend
- **C**: Coil

These datasets serve as the foundation for training and testing our deep learning model, which aims to predict protein secondary structures directly from sequence data.

4 Methods

4.1 Preprocessing

Data preprocessing prepares raw protein sequence data for model training. It includes several key steps:

1. **Data Loading and Filtering**: Datasets are loaded from CSV files, and sequences containing non-standard amino acids are filtered out to ensure consistency.

2. **Feature Normalization**: Sequences are normalized to a uniform range of [0, 1], ensuring equal contribution from all features.

3. **One-Hot Encoding of Sequences**: Sequences are converted into one-hot encoded vectors, representing each amino acid as a binary vector.

4. **Label Encoding:** Labels representing secondary structure types (e.g., `sst3` or `sst8`) are encoded as integers for machine learning compatibility.

5. **Padding Sequences:** Sequences are padded to the length of the longest sequence in the dataset, ensuring uniformity for batch processing.

6. **Final Data Preparation:** The sequences and labels are processed, encoded, and padded, ensuring a clean and consistent dataset for model training.

These steps ensure that the dataset is structured and ready for machine learning, with consistent inputs for training.

4.2 Model Architecture

In this study, we employed deep learning techniques to predict protein secondary structures from amino acid sequences. Specifically, we utilized three types of Recurrent Neural Networks: standard RNNs, Long Short-Term Memory networks (LSTM), and Gated Recurrent Units (GRU). Each of these models is well-suited for handling sequential data such as protein sequences, where the order of amino acids plays a critical role in determining the secondary structure.

4.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed for processing sequential data, where each output depends on previous inputs. They maintain an internal state that captures information from prior time steps, making them ideal for tasks such as protein sequence analysis, where the order of amino acids is crucial. RNNs are particularly effective for predicting protein secondary structures, as they capture dependencies within sequences. However, RNNs face challenges such as vanishing gradients, which can hinder learning long-term dependencies. Advanced architectures like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address these issues, offering better performance for sequence-based tasks [IBM, 2024].

4.4 Types of Recurrent Neural Networks

Long Short-Term Memory LSTMs [IBM, 2024] improve on standard RNNs by introducing a more complex unit structure. An LSTM cell consists of three gates: the input gate, the forget gate, and the output gate. These gates regulate the flow of information within the cell and allow it to remember information for long periods, making LSTMs particularly well-suited for tasks that require capturing long-term dependencies. The forget gate decides which information should be discarded, while the input gate controls the amount of new information to store. The output gate determines the final output of the cell.

Gated Recurrent Units GRUs [IBM, 2024] are a simplified version of LSTMs, with fewer parameters, making them computationally more efficient. A GRU combines the forget and input gates into a single update gate, which simplifies the model while still allowing it to capture long-term dependencies. The GRU's reset gate controls how much of the previous memory to forget, and the update gate decides how much of the new memory to retain. This reduced complexity often makes GRUs faster to train, while still achieving comparable performance to LSTMs in many applications.

Comparison of LSTM and GRU While both LSTMs and GRUs are effective in handling long-term dependencies, the choice between them depends on the specific task and dataset. LSTMs tend to perform better in tasks where more complex memory mechanisms are needed, but GRUs are often preferred in tasks where computational efficiency is critical, as they are simpler and require fewer parameters. The performance differences between the two are often marginal, and in some cases, GRUs have been shown to outperform LSTMs in terms of training time and accuracy.

In this work, several types of RNN architectures were tested, including the traditional RNN, LSTM, and GRU. The experiments were designed to evaluate the performance of these models in predicting protein secondary structures.

4.5 Metrics

The performance of the RNN model is evaluated using several important metrics that assess both the accuracy and the class-specific performance of the model. The following metrics are computed:

Accuracy Accuracy is a fundamental metric that measures the percentage of correct predictions out of all predictions. It is calculated as:

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

In this case, the accuracy is computed for both the training and test sets.

F1 Score The F1 score is the harmonic mean of precision and recall, and is particularly useful when dealing with imbalanced classes. It provides a single measure of model performance, balancing the trade-off between precision and recall. The weighted F1 score, which accounts for the class distribution, is calculated as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where precision is the proportion of positive predictions that are actually correct, and recall is the proportion of actual positives that are correctly identified.

Confusion Matrix The confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. From this, sensitivity (recall) and specificity can be derived. The confusion matrix for multiple classes provides insight into which classes are often confused by the model.

Sensitivity Sensitivity, also known as recall or true positive rate, is the proportion of actual positive cases that are correctly identified by the model. It is calculated as:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

This metric is important when the goal is to identify as many positive cases as possible.

Specificity Specificity, measures the proportion of actual negative cases that are correctly identified. It is calculated as:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity is important in ensuring that the model does not falsely classify negative instances as positive.

4.6 Experimental Setup

The experiments were designed to evaluate the performance of three types of Recurrent Neural Networks (RNNs): standard RNNs, Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRU). These architectures were chosen due to their ability to capture long-range dependencies in sequential data, which is particularly useful for protein sequence analysis. Each RNN model was trained with varying hyperparameters to explore the effect of different configurations on the model's performance.

The following hyperparameters were considered for experimentation:

- **Learning rate** (*lr*): The learning rate determines the size of the steps the model takes when adjusting its weights during training. This hyperparameter controls how fast the model converges.
- **Hidden size** (*hidden_ size*): The size of the hidden state vector in the RNN. A larger hidden size allows the model to capture more complex patterns in the data.
- **Dropout rate** (*dropout_ rate*): Dropout is applied during training to prevent overfitting by randomly setting a fraction of the input units to zero at each update.
- **Number of layers** (*num_ layers*): The number of stacked RNN layers. Increasing the number of layers can help the model capture more intricate dependencies in the sequence.
- **Batch size** (*batch_ size*): The number of samples processed before the model's weights are updated. A larger batch size can lead to more stable training, while a smaller batch size may increase model generalization.
- **RNN type** (*rnn_ type*): The type of RNN architecture to be used in the model. Options include standard RNNs, LSTMs, and GRUs, each with different strengths in handling sequence data.

A random search strategy was employed to test various combinations of these hyperparameters. For each combination, the model was trained for 20 epochs, and the performance was evaluated on both training and test datasets. The performance metrics used to evaluate the models include accuracy, F1 score, confusion matrix, sensitivity, and specificity.

The results from these experiments are presented in the next section, where we discuss the final selection of the model and its performance on the test dataset.

5 Results

In this section, we present the results obtained during the training and evaluation of Recurrent Neural Networks (RNNs) applied to protein secondary structure prediction. The experiments were conducted with various hyperparameter configurations, exploring the impact of parameters such as hidden size, number of recurrent layers, dropout, learning rate, and batch size.

We evaluated three types of RNN architectures: standard RNNs, Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM) networks. While all models were tested, the best-performing results were consistently achieved with LSTM networks, likely due to their superior ability to capture long-range dependencies and mitigate the vanishing gradient problem. The LSTM's memory cell structure allowed for more effective learning across lengthy amino acid sequences, which proved critical for improving prediction accuracy. Optimization was performed through Random Search, enabling efficient exploration of the hyperparameter space.

The results are summarized in Table 1, sorted from best to worst performance based on accuracy.

The best performance was achieved in Trial 3, which reached an accuracy of 60.56%. This configuration used 3 recurrent layers, with 512 units in the hidden layer (hidden size), a dropout of 0.2, a learning rate of 0.00407, and a batch size of 16. In comparison, Trial 0 showed similar performance (60.49%) but with a configuration of 2 layers, a hidden size of 384, and a dropout of 0.4. Both models demonstrate the importance of deep architectures with moderate regularization to capture patterns present in amino acid sequences.

On the other hand, the worst performance was recorded in Trial 4, with an accuracy of 57.11%.

Table 1. Test Results with Different Hyperparameters

Trial	Value	Hidden Size	Num Layers	Dropout	Learning Rate	Batch Size
3	**60.56**	**512**	**3**	**0.2**	**0.00407**	**16**
0	60.49	384	2	0.4	0.00122	16
2	60.47	256	2	0.3	0.00964	16
6	60.29	448	2	0.2	0.00079	32
5	59.88	384	2	0.4	0.00149	32
10	59.81	512	3	0.1	0.00317	32
8	58.49	256	1	0.1	0.00044	16
9	58.26	64	1	0.5	0.00182	64
1	58.17	128	3	0.4	0.00031	16
7	57.45	256	3	0.2	0.00044	32
4	57.11	384	1	0.2	0.00021	32

6 Discussion

The results obtained demonstrate that the configuration of hyperparameters has a significant impact on the performance of Recurrent Neural Networks applied to protein secondary structure prediction. Trial 3, which used 3 recurrent layers, 512 hidden units, and a dropout of 0.2, achieved the best result, suggesting that deeper architectures allow the model to capture long-range dependencies present in amino acid sequences. Additionally, the moderate dropout value (0.2) contributed to avoiding overfitting, improving the model's generalization capabilities.

Another critical factor was the batch size. The best results (Trials 3, 0, and 2) were obtained with small batch sizes (16). This indicates that smaller batches favor more frequent parameter updates, allowing the model to adjust more precisely during training, particularly in recurrent networks.

The learning rate also showed a direct influence on performance. Models with learning rates between 0.001 and 0.004 achieved the best results, ensuring stable and efficient learning. Very low rates, such as in Trial 4 (0.00021), limited the model's convergence.

On the other hand, shallow architectures with a single recurrent layer (e.g., Trial 4 and Trial 8) demonstrated inferior performance, even with adequate regularization. This suggests that a single layer is insufficient to capture the hierarchical and complex relationships between amino acids.

It was also observed that excessive dropout values (e.g., 0.5 in Trial 9) negatively impacted performance, possibly by eliminating important units during training. Conversely, moderate dropout values between 0.2 and 0.4 were more effective at preventing overfitting without compromising the model's learning capacity.

The results reinforce that the ideal combination of depth, moderate regularization, and adjusted learning rates is essential for successful protein secondary structure prediction. In particular: Deeper models (3 layers) exhibit greater learning capacity, larger hidden sizes (512) enable more robust representations, moderate dropout (0.2–0.4) balances learning and regularization, small batch sizes (16) favor more precise parameter updates during training.

These findings demonstrate the effectiveness of Recurrent Neural Networks (RNNs) in capturing complex sequential patterns in proteins and confirm the importance of a systematic approach to hyperparameter optimization.

7 Conclusion

The results of this study demonstrate the potential of deep learning models, particularly Recurrent Neural Networks (RNNs), to capture the sequential and complex dependencies inherent in protein secondary structures. Our findings emphasize the importance of deep architectures, balanced regularization, and carefully tuned hyperparameters, such as dropout and learning rate, in enhancing predictive accuracy. Notably, smaller batch sizes and multiple recurrent layers proved particularly effective in improving performance.

While the achieved accuracy of 60.56% is promising, there remains room for further improvement. Future directions include integrating evolutionary information via sequence profiles like PSSMs, leveraging hybrid architectures that combine CNNs and RNNs, and exploring advanced regularization techniques such as adaptive dropout and weight decay. Additionally, modern attention-based models like Transformers hold significant potential for capturing intricate sequence patterns and dependencies.

Expanding the dataset's size and diversity will also be crucial for enhancing model generalization, ensuring robust performance on unseen sequences. These advancements could pave the way for state-of-the-art solutions in protein structure prediction, with far-reaching applications in drug discovery, protein engineering, and bioinformatics, ultimately advancing research in computational biology.

Acknowledgments

We would like to acknowledge the assistance of several tools and resources that contributed to the development of this paper. ChatGPT was utilized to help refine and correct the writing throughout the manuscript. Additionally, GitHub Copilot provided valuable support in the execution of the code. Furthermore, We also wish to express our gratitude to the authors of [Reddy, 2020] for sharing the RNN code that was instrumental to our project.

References

[Chou and Fasman, 1974] Chou, P. Y. and Fasman, G. D. (1974). Prediction of protein conformation. *Biochemistry*, 13(2):222–245. PMID: 4358940.

[Holley and Karplus, 1989] Holley, L. H. and Karplus, M. (1989). Protein secondary structure prediction with a neural network. *Proceedings of the National Academy of Sciences*, 86(1):152–156.

[IBM, 2024] IBM (2024). Recurrent neural networks.

[Ismi et al., 2022] Ismi, D. P., Pulungan, R., and Afiahayati (2022). Deep learning for protein secondary structure prediction: Pre and post-AlphaFold. *Comput. Struct. Biotechnol. J.*, 20:6271–6286.

[Jumper et al., 2021] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

[Li and Yu, 2016] Li, Z. and Yu, Y. (2016). Protein secondary structure prediction using cascaded convolutional and recurrent neural networks.

[Lyu et al., 2021] Lyu, Z., Wang, Z., Luo, F., Shuai, J., and Huang, Y. (2021). Protein secondary structure prediction with a reductive deep learning method. *Frontiers in Bioengineering and Biotechnology*, 9.

[Reddy, 2020] Reddy, Y. (2020). Rnn python code in keras and pytorch.

[Rost and Sander, 1993] Rost, B. and Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232(2):584–599.

[Szelogowski, 2023] Szelogowski, D. (2023). Deep learning for protein structure prediction: Advancements in structural bioinformatics. *bioRxiv*, pages 2023–04.