# Secondary protein structure prediction using Deep Learning and Hidden Markov Models

Michele Copetti

June 2024

## Contents

# 1 Introduction

Proteins are essential macromolecules that play a critical role in virtually all biological processes. The function of a protein is inherently linked to its three-dimensional structure, which is determined by its amino acid sequence. Understanding protein structure is fundamental to fields such as biochemistry, molecular biology, and medicine. The secondary structure of proteins, which includes alpha helices and beta sheets, serves as the building blocks for the tertiary structure, influencing how the protein folds and interacts with other molecules. Accurate prediction of protein secondary structure is therefore a crucial step in determining protein function and designing novel therapeutics.

Despite significant advances in experimental techniques like X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and cryo-electron microscopy, determining protein structures experimentally remains time-consuming, expensive, and technically challenging. As a result, there is a substantial interest in developing computational methods to predict protein structures from amino acid sequences. Computational prediction not only accelerates the pace of discovery but also makes structural insights accessible for proteins that are difficult to study experimentally [5].

The aim of this project is to develop and evaluate computational tools for predicting the secondary structure of proteins. By leveraging machine learning algorithms and large datasets of known protein structures, the goal is to create a model that can accurately predict the formation of alpha helices, beta sheets, and other secondary structural elements from primary amino acid sequences. This involves analyzing large datasets of protein sequences and their corresponding structures, training predictive models, and validating their performance.

The Project is divided into two parts, a Deep Learning (DL) and a probabilistic part and the results are then compared against each other. In the DL part, the goal is to take advantage of the sequential nature of amino acid sequences and therefore use a recurrent neural network, specifically a long short-term memory (LSTM) network [1].
In the probabilistic part a Hidden Markov Model (HMM) is used to model the problem and a modified version of the Expectation-Maximization (EM) algorithm, the Baum-Welch algorithm, is used to determine the optimal parameters and labels of the model.

# 2 Methods

## 2.1 Overview

The project employs a hybrid approach combining deep learning and probabilistic models to predict the secondary structure of proteins from their primary amino acid sequences. Specifically, a Long Short-Term Memory (LSTM) network is used for capturing long-range dependencies within protein sequences, while a Hidden Markov Model (HMM) is used for probabilistic sequence analysis. The HMM is optimized using the Baum-Welch algorithm, enhancing its accuracy and robustness. This section details the mathematical concepts, algorithms, and data processing involved in the model.

## 2.2 Preprocessing

Before we can develop and train our LSTM and our HMM, we need to preprocess our data (The data can be found here) [6]
For the deep learning part, the dataset was parsed to contain only one label per sequence and the sequences were made equally long to a length of 7 (the most common length for alpha helices and beta sheets). The primary amino acid sequence was one-hot-encoded, converted to a tensor and the labels were converted to categorical variables (0,1 or 2 for 'Other (C)', 'Helix (H)' or 'Sheet (E)'). Also, because of limited computational ressources, sequences were capped to 40'000. The sequences were divided into a 60-20-20 split for training, validation and testing.
For the HMM part preprocessing was almost the same but instead of one-hot-encoding the primary sequences, they were converted to categorical variables from 0 to 19 and converted to arrays rather than tensors. Also, the sequences were divided into an 80-20 split for training and testing.

## 2.3 Long short-term memory (LSTM) networks

LSTM networks are a type of recurrent neural network (RNN) designed to model sequential data[1]. They are particularly well-suited for handling long-range dependencies due to their unique architecture and are designed to overcome the limitations of traditional RNNs by avoiding issues like vanishing or exploding gradients, which includes memory cells and gates (input, output, and forget gates) that control the flow of information (See Figure 1).

### 2.3.1 Components of an LSTM Cell [4]

1. **Cell State ($C_t$)**: This represents the long-term memory of the network.

2. **Hidden State ($H_t$)**: This represents the short-term memory and is used for output at each time step.

3. **Input Gate ($I_t$)**: Controls how much of the new input should be added to the cell state.

4. **Forget Gate ($F_t$)**: Controls how much of the previous cell state should be forgotten.

5. **Output Gate ($O_t$)**: Controls how much of the cell state should be exposed to the hidden state.

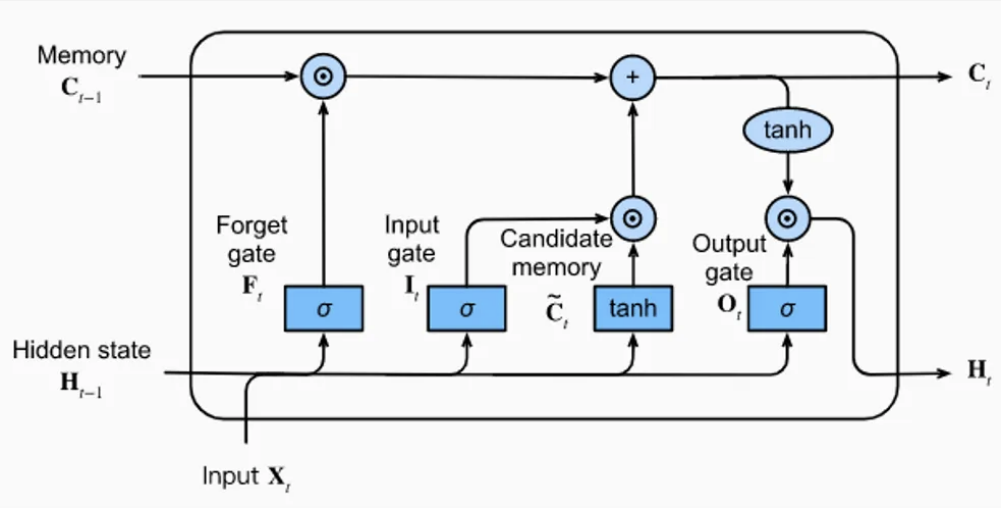


Figure 1: Architecture of an LSTM cell including the Cell State, Hidden state, Input-, Output- and Forget-Gate

### 2.3.2 Detailed LSTM Equations [4]

1. **Forget Gate**: The forget gate determines which information from the previous cell state should be discarded. It is computed using the previous hidden state ($H_{t-1}$) and the current input ($X_t$), passed through a sigmoid activation function.

$$f_t = \sigma(W_f \cdot [H_{t-1}, X_t] + b_f)$$

Where:

- $\sigma$ is the sigmoid function.
- $W_f$ are the weights for the forget gate.

- $b_f$ is the bias for the forget gate.

2. **Input Gate**: The input gate decides which values from the current input should be used to update the cell state. It has two parts: a sigmoid layer and a tanh layer to create new candidate values ($\tilde{C}_t$) that could be added to the cell state.

$$I_t = \sigma(W_i \cdot [H_{t-1}, X_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [H_{t-1}, X_t] + b_C)$$

Where:

- $W_i$ and $W_C$ are the weights for the input gate and candidate cell state, respectively.
- $b_i$ and $b_C$ are the biases.

3. **Cell State Update**: The cell state is updated by combining the old cell state ($C_{t-1}$) and the new candidate cell state ($\tilde{C}_t$), modulated by the forget and input gates.

$$C_t = f_t \cdot C_{t-1} + I_t \cdot \tilde{C}_t$$

4. **Output Gate**: The output gate determines the next hidden state based on the updated cell state. The hidden state is a filtered version of the cell state, modulated by a tanh layer and the output gate.

$$O_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o)$$

$$H_t = O_t \cdot \tanh(C_t)$$

Where:

- $W_o$ is the weight for the output gate.
- $b_o$ is the bias.

### 2.3.3 Model Instantiation and training

The specific model used in this project is instantiated as follows:

```
Model(input_dim=20, hidden_dim=140, layer_dim=6, output_dim=3)
```

*Input Dimension*: This parameter represents the dimensionality of the input features. In this case, each amino acid in a protein sequence is encoded as a one-hot vector of length 20, corresponding to the 20 standard amino acids.

*Hidden Dimension*: This parameter specifies the number of features in the hidden

state of the LSTM cells. A hidden dimension of 140 means that each LSTM cell produces an output vector of length 140 at each time step. This allows the model to capture complex patterns and dependencies within the input sequences.

*Layer Dimension* This parameter denotes the number of LSTM layers stacked in the network. Having multiple layers enables the model to learn hierarchical features from the input data, with each successive layer capturing increasingly abstract representations of the sequence information.

*Output Dimension*: This parameter corresponds to the number of output classes, representing the three types of secondary structures being predicted: Helices, Sheets and Other. The final fully connected layer maps the learned features to these three classes, providing the probability of each amino acid belonging to each secondary structure type.

**Input Handling**: The model takes as input sequences of amino acids, where each amino acid is encoded as a 20-dimensional one-hot vector. This encoding allows the model to process the sequence data effectively.

**Feature Learning**: The LSTM layers process these sequences to capture temporal dependencies. Each LSTM cell outputs a hidden state of 140 dimensions, and with 6 stacked layers, the model can learn complex and hierarchical features from the sequence data.

**Prediction**: The final fully connected layer converts the learned features into predictions for the secondary structure. The output is a vector of length 3, with each element representing the likelihood of the sequence being a Helix, Sheet, or Other.

***Training:*** As Loss function, the *CrossEntropyLoss* was chosen, which is the most common surrogate Loss function for multinomial classification problems. As optimizer, Adam was chosen with a learning rate of 10e-4, which is a first-order gradient-based optimization algorithm. The Batch-size was chosen to be 100 and the model was trained for 175 epochs.

## 2.4 Hidden Markov Model and Baum-Welch Algorithm

### 2.4.1 Hidden Markov Model (HMM)

A Hidden Markov Model (HMM) is a statistical model used to describe systems that transition between a set of hidden states over time [2],[8].

An HMM is defined by the following components:

- **States ($S$)**: The hidden states in our model

$$S = \{s_1, s_2, ...s_n\}$$

- **Observations ($V$)**: The set of observed symbols

$$V = \{v_1, v_2, ...v_n\}$$

- **Transition Probabilities ($A$)**: The probabilities of transitioning from one hidden state to another:

$$A = \{a_{ij}\} = P(s_{t+1} = s_j \mid s_t = s_i)$$

- **Emission Probabilities ($B$)**

$$B = \{b_j(k)\} = P(O_t = v_k \mid s_t = s_j)$$

- **Initial State Probabilities ($\pi$)**: The probabilities of being in each hidden state at the initial time step:

$$\pi = \{\pi_i\} = P(s_1 = s_i)$$

### 2.4.2 Baum-Welch Algorithm

The Baum-Welch algorithm is an Expectation-Maximization (EM) algorithm used to estimate the unknown parameters of an HMM [3], [7]. It iteratively adjusts the transition and emission probabilities to maximize the likelihood of the observed data. The algorithm involves two main steps:

- **Expectation Step (E-step)**: Calculate the expected number of times each transition and each emission occurs.
- **Maximization Step (M-step)**: Update the model parameters to maximize the likelihood of the observed data given these expected counts.

### 2.4.3 HMM and Baum-Welch for Protein Secondary Structure Prediction

In the context of protein secondary structure prediction, the hidden states of an HMM represent different secondary structure elements: Helices (H), Sheets (E), or Other (C). The observed states (emissions) are the amino acids in the protein sequence.
Normally, there exist multiple hidden states, and the goal in secondary structure prediction is to infer the label for each position of an amino acid sequence. However, in my approach, I used the HMMs differently since my goal is to predict only one label per sequence and not a label for every position of each sequence. During training, a separate HMM is trained for each class using the sequences labeled with that class. Each HMM has three hidden states, but the parameters are adjusted such that one hidden state becomes dominant for sequences of a particular class. During testing, I compute the likelihood of a new sequence under each trained HMM and assign the label corresponding to the HMM with the highest likelihood.

***Adjusted HMM Parameters:***

For each class $N \in \{H, E, O\}$, the HMM parameters are:

- **States ($S_N$)**: The set of states for the HMM corresponding to class $N$. Each HMM has three hidden states:

$$S_N = \{s_{N1}, s_{N2}, s_{N3}\}$$

- **Observations ($V$)**: The set of observed symbols (amino acids).

- **Transition Probabilities ($A_N$)**: The probabilities of transitioning from one hidden state to another:

$$A_N = \{a_{ij}\} = P(s_{t+1} = s_{Nj} \mid s_t = s_{Ni})$$

- **Emission Probabilities ($B_N$)**: The probabilities of observing each amino acid given the hidden state $s_{Ni}$:

$$B_N = \{b_j(k)\} = P(O_t = v_k \mid s_t = s_{Nj})$$

- **Initial State Probabilities ($\pi_N$)**: The probabilities of being in each hidden state at the initial time step:

$$\pi_N = \{\pi_{Ni}\} = P(s_1 = s_{Ni})$$

***Adjusted Baum-Welch Algorithm:***

**Initialization** The transition probabilities $A_N$, emission probabilities $B_N$ and initial probabilities $\pi_N$ are initialized randomly.

**Expectation Step (E-step)**

1. **Forward Algorithm**: Calculates the probability of observing a sequence up to time $t$ given state $s_{Ni}$ at time $t$:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = s_{Ni} \mid \lambda)$$

   - **Initialization**:

$$\alpha_1(i) = \pi_{Ni} \cdot b_{s_{Ni}}(O_1) \quad \text{for } i = 1, 2, 3$$

   - **Recursion**: For $t = 2$ to $T$:

$$\alpha_t(j) = \left( \sum_{i=1}^{3} \alpha_{t-1}(i) \cdot a_{ij} \right) \cdot b_{s_{Nj}}(O_t)$$

2. **Backward Algorithm**: Computes the probability of the ending partial observation sequence from time $t+1$ to the end, given state $s_{Ni}$ at time $t$:

   - **Initialization**:

$$\beta_T(i) = 1 \quad \text{for } i = 1, 2, 3$$

   - **Recursion**: For $t = T - 1$ to 1:

$$\beta_t(i) = \sum_{j=1}^{3} a_{ij} \cdot b_{s_{Nj}}(O_{t+1}) \cdot \beta_{t+1}(j)$$

3. **Compute** $\xi_t(i, j)$

   The probability of being in state $s_{Ni}$ at time $t$ and state $s_{Nj}$ at time $t+1$:

$$\xi_t(i, j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_{s_{Nj}}(O_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_t(i) \cdot a_{ij} \cdot b_{s_{Nj}}(O_{t+1}) \cdot \beta_{t+1}(j)}$$

4. **Compute** $\gamma_t(i)$

   The probability of being in state $s_{Ni}$ at time $t$:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{3} \alpha_t(i) \cdot \beta_t(i)}$$

**Maximization Step (M-step)** Update the initial state probabilites $\pi_N$ transition probabilities $a_{ij}$ and emission probabilities $b_{s_{Ni}}(v_k)$:

$$\pi_i = \gamma_1(i)$$
$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$
$$b_{s_{Ni}}(v_k) = \frac{\sum_{t=1}^{T} \gamma_t(i) \cdot \delta(O_t = v_k)}{\sum_{t=1}^{T} \gamma_t(i)}$$

***Getting the Likelihood of a new sequence:***

Given an HMM $\lambda_N = (A_N, B_N, \pi_N)$ and an observed sequence $O = (O_1, O_2, \ldots, O_T)$, the forward algorithm computes the likelihood of the sequence under the model.

**Initialization**

$$\alpha_1(i) = \pi_{Ni} \cdot b_{s_{Ni}}(O_1) \quad \text{for } i = 1, 2, 3$$

**Recursion** For $t = 2$ to $T$,

$$\alpha_t(j) = \left( \sum_{i=1}^{3} \alpha_{t-1}(i) \cdot a_{ij} \right) \cdot b_{s_{Nj}}(O_t)$$

**Termination** The likelihood of the observed sequence $O$ under the model $\lambda_N$ is given by:

$$P(O \mid \lambda_N) = \sum_{i=1}^{3} \alpha_T(i)$$

***Classification Step:***

For a given sequence $O$, compute the likelihood under each HMM corresponding to each class:

$$P(O \mid \lambda_H), \quad P(O \mid \lambda_E), \quad P(O \mid \lambda_O)$$

Assign the label corresponding to the HMM with the highest likelihood:

$$\hat{N} = \arg \max_{N \in \{H,E,O\}} P(O \mid \lambda_N)$$

# 3 Results

This section presents the performance results of the Deep Learning (DL) and Hidden Markov Model (HMM) approaches for protein secondary structure prediction. The performance metrics considered are Precision, Recall, F1-Score, and Accuracy. Additionally, confusion matrices and loss/accuracy curves are provided to visualize the performance and training dynamics.

## 3.1 Performance Metrics

Table 1 and 2 summarizes the performance metrics for the DL and HMM approaches. The DL approach achieves a precision, recall, F1-Score, and accuracy of

| Metric | Value |
|---|---|
| Precision | 0.88 |
| Recall | 0.88 |
| F1-Score | 0.88 |
| Accuracy | 0.88 |

Table 1: Metrics for the DL part

| Metric | Value |
|---|---|
| Precision | 0.78 |
| Recall | 0.76 |
| F1-Score | 0.76 |
| Accuracy | 0.76 |

Table 2: Metrics for the HMM part

0.88, indicating a high level of performance in predicting the secondary structure of proteins. In comparison, the HMM approach achieves a precision of 0.78, recall of 0.76, F1-Score of 0.76, and accuracy of 0.76, which, while lower than the DL approach, still demonstrates good performance.

## 3.2 Confusion Matrices

Figures 2 and 3 show the normalized confusion matrices for the DL and HMM approaches, respectively. The confusion matrices provide a detailed view of the true versus predicted labels, allowing for a deeper understanding of the performance for each class. The confusion matrix for the DL approach (Figure 2) shows that the model performs well across all classes, with the highest accuracy in predicting the 'C' (Other) class. The 'H' (Helix) and 'E' (Sheet) classes also show strong performance, although the model struggles the most with predicting the class 'H' correctly.

In contrast, the confusion matrix for the HMM approach (Figure 3) indicates a slightly lower performance, particularly for the 'E' (Sheet) class, which shows more misclassifications into the 'Other' and 'Helix' classes. The 'Other' class has the highest accuracy, similar to the DL approach, but the overall performance is slightly lower. The HMM approach also struggles the most with predicting the 'H' class correctly, only achieving a score of 0.72 for this class.

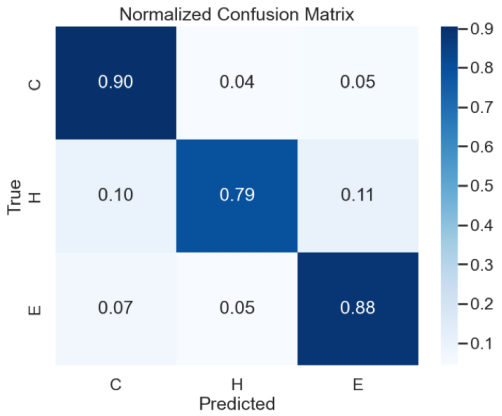


Figure 2: Normalized Confusion Matrix for the DL part

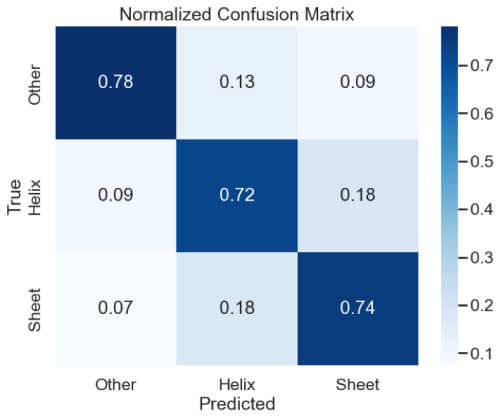


Figure 3: Normalized Confusion Matrix for the HMM part

## 3.3 Training Dynamics

Figure 4 shows the training and validation loss over epochs for the DL approach, as well as the validation accuracy over epochs. The training loss decreases steadily

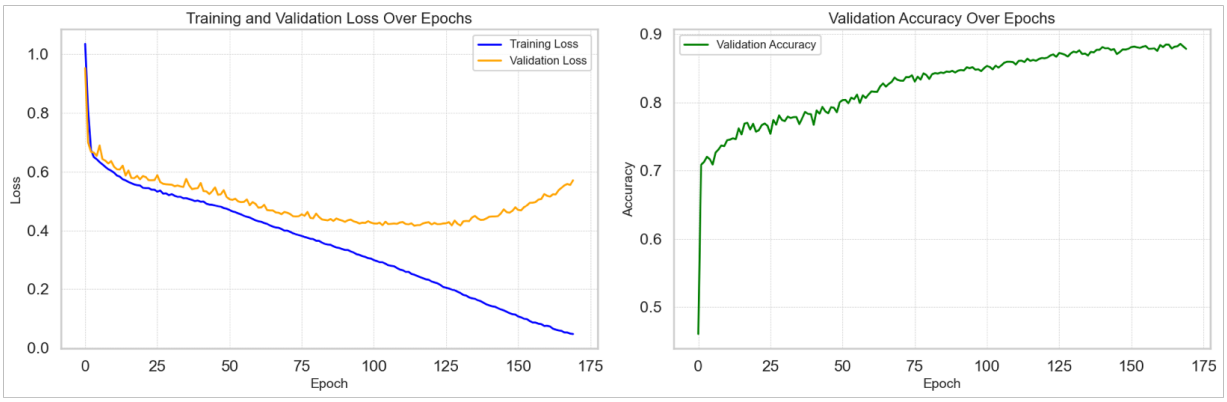


Figure 4: Training Loss, Validation Loss and Validation Accuracy Over Epochs

over time, indicating that the model is learning effectively. The validation loss, however, shows an increase after about 100 epochs, suggesting potential overfitting. Despite this, the validation accuracy continues to improve and is higher after 175 epochs than it was at 100 epochs. This indicates that the model is able to generalize well on the test data, even if the validation loss increases slightly.

# 4 Discussion

The results indicate that the Deep Learning (DL) approach outperforms the Hidden Markov Model (HMM) approach in predicting protein secondary structure. The higher precision, recall, F1-Score, and accuracy metrics for the DL model suggest that it is more effective in capturing the patterns in the protein sequences. The confusion matrices further illustrate this, with the DL model showing fewer misclassifications across all classes.
However, the HMM approach still demonstrates good performance, particularly given its simplicity compared to the DL approach. The HMM model's performance may be improved with further tuning of parameters and incorporation of additional features.
While the DL architecture showed superior predictive power, it is essentially a black-box model. This lack of interpretability can be a significant drawback, especially in biological research where understanding the underlying mechanisms is crucial. Additionally, the DL model requires significantly more computational resources, both in terms of hardware and training time, which can be a limiting factor for some applications.
In contrast, the EM/Baum-Welch algorithm combined with HMMs remains a powerful tool. These models are robust and can handle missing data and unknown parameter distributions effectively. Moreover, HMMs are computationally efficient and "easily" understandable, making them a valuable choice for many practical applications. The interpretability of HMMs allows researchers to gain insights into the underlying biological processes, which is often more important than achieving the highest predictive accuracy. Also, HMMs are basically only semi-supervised, even working with missing labels and could theoretically be expanded to be fully unsupervised, whick makes them important for tasks where labelling is difficult.
The training dynamics indicate that the DL model may be overfitting the training data after a certain number of epochs. However, the continuous improvement in validation accuracy suggests that the model retains its generalization capabilities on the test data. This is confirmed when looking at the test scores in Table 1. This highlights the importance of monitoring both loss and accuracy during training to better understand the model's performance.

# 5 Conclusion

In summary, the DL approach provides a higher level of accuracy in predicting protein secondary structures compared to the HMM approach. Despite this, the HMM together with the Baum-Welch algorithm, which iteratively refines the transition and emission probabilities, remains a valuable model due to its interpretability and

potential for further optimization. The choice between these methods should consider both the specific application requirements and the available computational resources.

## 5.1 Outlook

Future research directions could include extending the current methodologies to more complex problems, such as tertiary structure prediction and protein-protein interaction prediction. These tasks represent the next step in understanding protein function and behavior. Additionally, hybrid approaches that combine the strengths of DL and HMMs could be explored to leverage both the high predictive power of DL and the interpretability and robustness of HMMs.
Furthermore, developing more interpretable DL models or incorporating interpretability mechanisms within DL architectures could bridge the gap between predictive power and understanding. Another potential direction is to optimize DL models for better computational efficiency, making them more accessible for broader applications.

## 5.2 Self-assessment

Reflecting on this project, several limitations and challenges were encountered. One limitation is the black-box nature of the DL model, which, despite its high accuracy, provides little insight into the underlying biological processes. Additionally, the high computational resources required for training the DL model posed a significant challenge.
The HMM approach, while more interpretable and computationally efficient, did not achieve the same level of accuracy as the DL model. However, its robustness and ability to handle missing data and unknown parameter distributions are significant strengths.
Future studies could focus on improving the interpretability of DL models and optimizing them for computational efficiency. Additionally, further tuning of the HMM parameters and exploring hybrid models could enhance predictive performance while maintaining interpretability. Overall, this project provides valuable insights into the strengths and weaknesses of both DL and HMM approaches, and helping my future studies in Data Science and Bioinformatics.

# References


[1] Sepp Hochreiter and Jürgen Schmidhuber. *Long short-term memory*. Vol. 9. 8. MIT Press, 1997, pp. 1735–1780.

[2] Sean R Eddy. "What is a hidden Markov model?" In: *Nature Biotechnology* 22.10 (2004), pp. 1315–1316.

[3] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[4] Christopher Olah. *Understanding LSTM Networks*. Accessed: 2024-06-13. 2015. URL: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[5] John Jumper, Richard Evans, Alexander Pritzel, et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[6] Alf Random. *Protein Secondary Structure Dataset*. Accessed: 2024-06-11. 2024. URL: `https://www.kaggle.com/datasets/alfrandom/protein-secondary-structure`.

[7] Wikipedia. *Baum-Welch algorithm*. Accessed: 2024-06-13. 2024. URL: `https://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm`.

[8] Wikipedia. *Hidden Markov model*. Accessed: 2024-06-13. 2024. URL: `https://en.wikipedia.org/wiki/Hidden_Markov_model`.


# Appendix

The entire code can be found here