Chapter 1

INTRODUCTION

The amino acid sequence of a protein largely determines its 3D structure, but predicting this structure from sequence is very difficult. Protein structure is crucial for understanding its function and designing drugs or enzymes that can modulate it. One of the steps towards protein structure prediction is to infer its secondary structure, which consists of local folding patterns such as alpha helices and beta sheets. These patterns are stabilized by hydrogen bonds between the backbone atoms of the protein. Various methods have been developed to predict secondary structure from sequence, such as information theory-based methods (e.g., GOR method), machine learning methods (e.g., JPred4), and deep learning methods (e.g., AlphaFold2 and RoseTTAfold). The accuracy of secondary structure prediction affects the quality of the final 3D structure prediction.

Proteins have four levels of structure that describe how they are folded and arranged. These are the primary, secondary, tertiary, and quaternary structure.

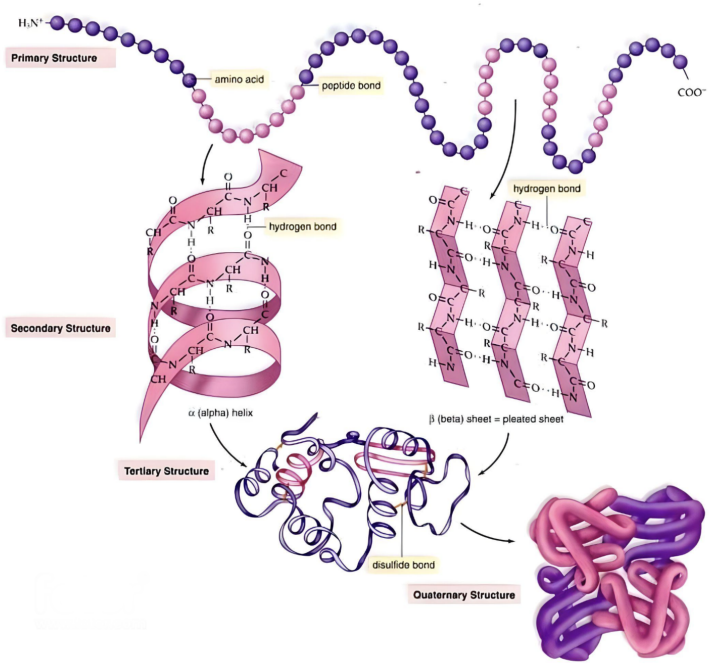


Figure 1.1: This figure illustrates the four levels of protein structure. The primary structure is the order of amino acids in the polypeptide chain. The secondary structure is the shape of local regions of the chain, such as alpha helices and beta sheets. The tertiary structure is the overall 3D configuration of the entire chain. The quaternary structure is the arrangement of multiple chains in a protein complex.Figure adapted from [1]

Primary structure is the level of protein structure that refers to the order of amino acids that make up a polypeptide chain.

Secondary structure is the local folding of a polypeptide chain due to backbone interactions. The backbone is the part of the chain that does not include the R groups. The two main types of secondary structure are the α helix and the β pleated sheet.

Tertiary structure is the 3D shape of a polypeptide. It depends mainly on interactions between the R groups of the amino acids in the protein. Predicting protein tertiary structure from only its amino acid sequence is a very challenging problem, but using the simpler secondary structure definitions is becomes more tractable.

My project goal was to use Deep Learning Techniques(Convolutional Neural Networks) to predict protein secondary structure (SS) from primary structure.

Chapter 2

Protein Structures and Protein Data

The primary structure of proteins is the sequence of amino acids in their polypeptide chain. The human body has 20 natural amino acids, each with a one-letter code: A for Alanine, C for Cysteine, D for Aspartic Acid, and so on. X is sometimes used for an unknown or any amino acid. (See Figure 2.1)

Protein Profiles (See figure 2.2) are a better way to represent the primary structure than just the amino acid sequence. They capture evolutionary relationships and protein family features. They are made by turning multiple sequence alignments into position-specific scoring systems (PSSMs). Each amino acid in the alignment gets a score based on how often it appears at that position [2].

Protein polypeptide chains vary in length from a few to hundreds of amino acids. The average chain in the dataset has 208 amino acids. Any amino acid can be at any position in the chain, so a 4-amino acid chain has 204 different combinations.

Secondary structure is the shape of local segments of amino acids in a protein. For example, alpha-helix is a coil and beta-strand is a zig-zag. Secondary structure shows the chemical properties of the protein and helps predict its tertiary structure. There are two ways to predict secondary structure: 3-state and 8-state.

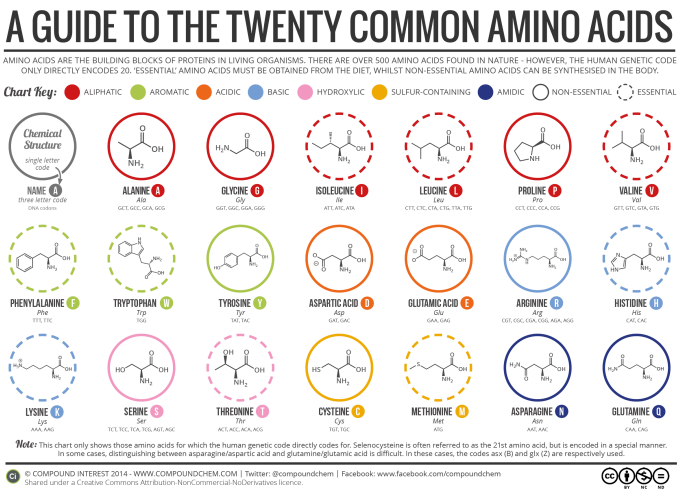


Figure 2.1: A table of the 20 amino acids in the human body and their properties.

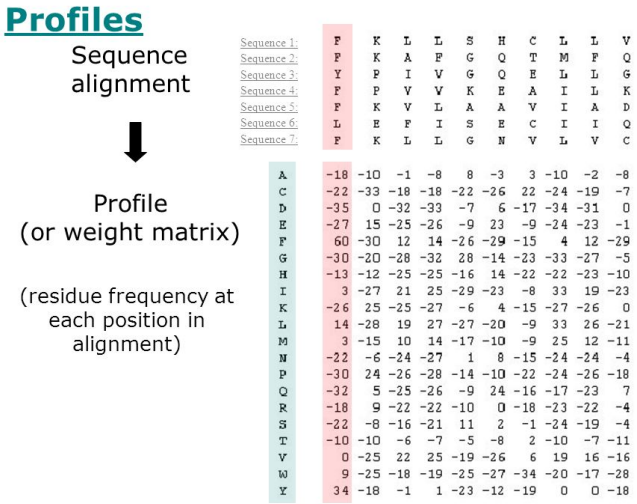


Figure 2.2: Protein Profiles are made by turning multiple sequence alignments into position-specific scoring systems (PSSMs). Each amino acid in the alignment gets a score based on how often it appears at that position [3].

For 3-state prediction the goal is to classify each amino acid into either

- alpha-helix(H)
- beta-strand(E)
- coil region(C)

The letters for secondary structures H,E,C are different from those for amino acids.

8-state prediction splits alpha-helix into three states: H, G, and I. It splits beta-strand into E and B. It splits coil region into S, T, and L [3]. These are the 8 states:

- E = extended strand, participates in β ladder
- B = residue in isolated β-bridge
- H = α-helix
- G = 3-helix (3-10 helix)
- I = 5-helix (π-helix)
- T = hydrogen bonded turn
- S = bend
- L = loop (any other type)

2.1 Dataset

Here, we utilize five different datasets, namely, CullPdb 6133, CullPdb 6133 filtered and CB513 Among these datasets CullPdb 6133, and CullPdb 6133 filtered for training. Furthermore, CB513 and 272 protein sequence of CullPdb 6133 for testing.

CullPdb 6133

CullPdb 6133 [4] dataset is a non-homologous protein dataset that is provided by PISCES CullPDB with the familiar secondary structure for protein. This dataset contains a total of 6128 protein sequences, in which 5600 ([0:5600]) protein samples are considered as the training set, 272 protein samples [5605:5877] for testing, and 256 proteins samples ([5877,6133]) regarded as the validation set. Moreover, CullPdb 6133 (non-filtered) dataset has 57 features, such as amino acid residues (features [0:22)), N- and C- terminals (features [31,33)), relative and absolute solvent accessibility ([33,35)), and features of sequence profiles (features [35:57)). We

used secondary structure notation (features [22:31)) for labeling. This CullPdb dataset is publicly obtainable from [5].

CullPdb 6133 filtered

The CB513 dataset and the CullPdb 6133 dataset have some overlap. To avoid this, the CullPdb 6133 filtered version removes sequences that share more than 25% similarity with CB513. This version has 5534 protein sequences. We use 5234 of them for training and 300 for validation, chosen randomly.

CB513

CB513 dataset contains 514 protein sequences, and it is widely utilized to compare the performance of protein secondary structure prediction. Here, we use this dataset for evaluating our testing result after training with CullPdb 6133 filtered. This dataset can be accessed from [2, 52].
Note that one of the sequences in CB513 is longer than 700 amino acids, and it is splited to two overlapping sequences and these are the last two samples (i.e. there are 514 rows instead of 513).

To use the features from CullPdb 6133 and CullPdb 6133, we reshaped these datasets from 6133 proteins x 39900 features and 5534 proteins x 39900 features into 6133 proteins x 700 amino acids x 57 features and 5534 proteins x 700 amino acids x 57 features respectively. Here, 700 indicate the peptide chain, and 57 signify the features for each amino acid. We use amino acid residues and sequence profiles (total of 42 features) for training input. Besides, the output is a protein secondary structure labeling.

2.2 Dataset Summary

In our dataset folder we have following 3 numpy files

cb513+profile_split.npy
cullpdb+profile_6133_filtered.npy

cullpdb+profile_6133.npy

cullpdb+profile_6133_filtered.npy is the one with training/validation/test set division, after filtering for redundancy with CB513. this is used for evaluation on CB513.

cb513+profile_split.npy is the CB513 including protein features. Note that one of the sequences in CB513 is longer than 700 amino acids, and it is splited to two overlapping sequences and these are the last two samples (i.e. there are 514 rows instead of 513).

It is currently in numpy format as a (N protein x k features) matrix. You can reshape it to (N protein x 700 amino acids x 57 features) first.

The 57 features are:

- $[2, 3)$: amino acid residues, with the order of 'A', 'C', 'E', 'D', 'G', 'F', 'I', 'H', 'K', 'M', 'L', 'N', 'Q', 'P', 'S', 'R', 'T', 'W', 'V', 'Y', 'X', 'NoSeq'
- $[22, 31)$: Secondary structure labels, with the sequence of 'L', 'B', 'E', 'G', 'I', 'H', 'S', 'T', 'NoSeq'
- $[31, 33)$: N- and C- terminals;
- $[33, 35)$: relative and absolute solvent accessibility, used only for training. (absolute accessibility is thresholded at 15; relative accessibility is normalized by the largest accessibility value in a protein and thresholded at 0.15; original solvent accessibility is computed by DSSP)
- $[35, 57)$: sequence profile. Note the order of amino acid residues is ACDEFGHIKLMNPQRSTVWXY and it is different from the order for amino acid residues

The last features of amino acids and secondary structures indicate the end of the protein sequence. The features in [22,31) and [33,35) are not used for testing.

As [5] recommended, the dataset (with 6133 proteins) was randomly split into training (5600), validation (256) and testing (272) subsets.

2.3 Implementation

The project used Keras as the framework and Tensorflow as the backend. It tried two main methods:

1. The CNN took the entire protein sequence (primary structure) as an input, and produced an output of size 700 x 9, which was the predicted secondary structure sequence.

2. The CNN took small segments of the protein sequence as inputs, and moved along the sequence. For each segment, it predicted the secondary structure (8 classes) at the middle position of the segment.

2.4 Whole protein sequence prediction

The initial model had 3 main 1D Convolutional Layers with Dropout regularization that took the entire protein sequence as input. The input had 700 amino acids (with some padding) and 9 channels for the 8 possible states of each amino acid and the padding state. (refer Fig 2.3)

▸ This simple **model** consists of 3 main 1D Convolutional Layers:

```
LR = 0.0005
drop_out = 0.3
batch_dim = 64

loss = 'categorical_crossentropy'

# We fix the window size to 11 because the average length of an alpha helix is around
eleven residues
# and that of a beta strand is around six.
# See references [6].
m = Sequential()
m.add(Conv1D(128, 11, padding='same', activation='relu', input_shape=(dataset.sequence_len,
dataset.amino_acid_residues)))
m.add(Dropout(drop_out))
m.add(Conv1D(64, 11, padding='same', activation='relu'))
m.add(Dropout(drop_out))
m.add(Conv1D(dataset.num_classes, 11, padding='same', activation='softmax'))
opt = optimizers.Adam(lr=LR)
m.compile(optimizer=opt, loss=loss, metrics=['accuracy', 'mae'])
```

Figure 2.3: This figure illustrates the CNN model for Whole sequence protein prediction

This prototype had a small number of parameters (125.512 trainable parameters). However, this approach had a major issue: the padding for shorter sequences

affected the loss, which was computed on the whole output sequence (using "categorical_crossentropy" loss from tensorwlow).

This meant that a custom loss had to be created to account for the outputs from the padding region, which had a different shape for each example. This approach was soon discarded.

2.5 Sliding Window CNN

The model had 3 main 1D Convolutional Layers with DropOut and Batch Normalization, followed by 3 Fully connected Layers.

The window size was chosen to be larger than 11, because an alpha helix has an average length of about eleven residues and a beta strand has about six (See references [6]). Several even sizes from 11 to 23 were tried, and 17 gave the best results (balancing performance and training time). (refer Fig 2.4)

The model had 232.552 parameters, of which 231.912 were trainable. It was trained on 946494 examples and validated on 120704 examples (windows).

This **model** implementation:

```
cnn_width = 17

LR = 0.0009 # maybe after some (10-15) epochs reduce it to 0.0008-0.0007
drop_out = 0.38
batch_dim = 64

loss = 'categorical_crossentropy'

m = Sequential()
m.add(Conv1D(128, 5, padding='same', activation='relu', input_shape=(cnn_width,
dataset.amino_acid_residues)))
m.add(BatchNormalization())
m.add(Dropout(drop_out))
m.add(Conv1D(128, 3, padding='same', activation='relu'))
m.add(BatchNormalization())
m.add(Dropout(drop_out))
m.add(Conv1D(64, 3, padding='same', activation='relu'))
m.add(BatchNormalization())
m.add(Dropout(drop_out))
m.add(Flatten())
m.add(Dense(128, activation='relu'))
m.add(Dense(32, activation='relu'))
m.add(Dense(dataset.num_classes, activation = 'softmax'))
opt = optimizers.Adam(lr=LR)
m.compile(optimizer=opt, loss=loss, metrics=['accuracy', 'mae'])
```

Figure 2.4: This figure illustrates the CNN model for Sliding Window CNN

Chapter 3

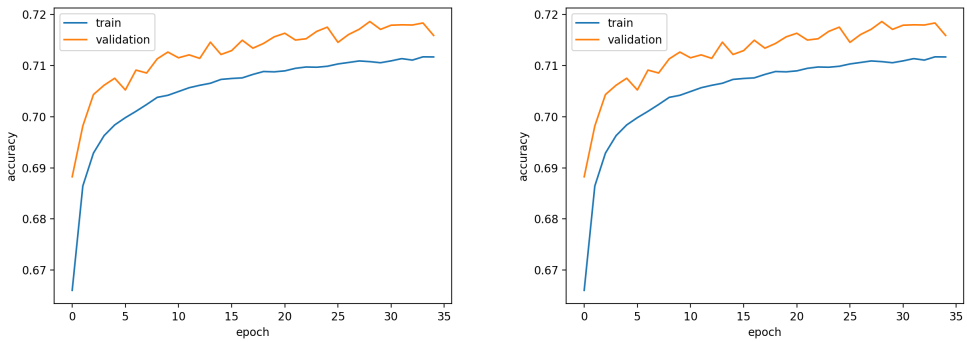
Result

Sliding Window CNN

The model was trained on the CullPDB dataset with the split described in section 2.1 for 35 epochs (taking about 6 hours on CPU). Figure 3.1 shows the learning curves for this method. The model achieved an accuracy of 0.721522 (Q8 Accuracy) on the test set, which is similar to the results of [5] and [6] with different methods. The model was also trained on the filtered dataset: CullPDB6133+filtered from [4] and tested on the public benchmark CB513. The accuracy was 0.6833 (Q8 Accuracy), again close to [5] and [6].

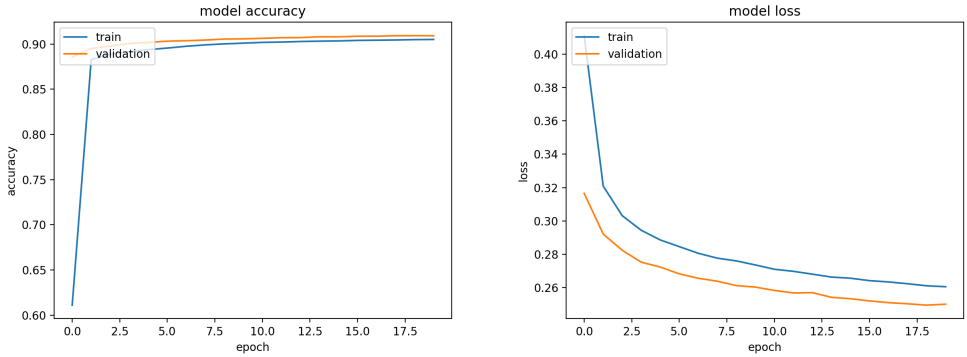
Whole protein sequence prediction

The model was trained on the CullPDB dataset with the split described in section 2.1 for only 20 epochs (taking about 25 minutes on CPU). Figure 3.2 shows the learning curves for this method. The model did not consider the padding in the loss calculation, so the values were skewed. The model achieved an accuracy of 0.6966 (Q8 Accuracy) on the test set, which was close to the results of the Window CNN with much less time. The model also achieved an accuracy of 0.6557 (Q8 Accuracy) when trained on the filtered dataset and tested on the CB513 dataset.



(a) Window CNN Accuracy (Q8 Accuracy) (b) Window CNN Loss

Figure 3.1: Learning curves: (a) Accuracy at each epoch for the Window CNN approach, (b) Loss value at each epoch for the Window CNN approach.



(a) Whole protein CNN Accuracy (b) Whole protein CNN Loss

Figure 3.2: Learning curves: (a) Accuracy at each epoch for the Whole protein approach, (b) Loss value at each epoch for the Whole protein approach.

Chapter 4

Conclusion & Future Scope

In this study our model achieved the results with high efficiency. The best model had an accuracy of 0.721522 (Q8 Accuracy) on the test set as described in section 2.1 and 0.6833 (Q8 Accuracy) on the public benchmark CB513.

Protein secondary structure prediction is a challenging and important task in bioinformatics. Deep learning techniques have shown promising results in this field, achieving high accuracy and efficiency. In recent years, many researchers have proposed advanced neural network models that can predict the secondary structure of proteins with more than 70% accuracy in Q3 and Q8 metrics. These models use various features and architectures to capture the complex patterns and dependencies of amino acid sequences. The future scope of protein secondary structure prediction using deep learning techniques is very vast, as there are still many open problems and opportunities for improvement. Some possible directions for future research include incorporating more biological knowledge, exploring different representations and inputs, enhancing model interpretability and robustness, and applying the techniques to other related tasks.

The code developed for this project is available on GitHub at:https://github.com/umakantmukhiya Secondary-Structure-using-CNN.git

Bibliography

[1] S. Mader, Biology. McGraw Hill, 2010.

[2] EMBL−EB (The home for big data in biology). What are profiles? https://bit.ly/2Uj1RVC

[3] Wikipedia. Protein secondary structure — Wikipedia, The Free Encyclopedia. https://bit.ly/2WRirsJ

[4] Wang G, Jr DR. Pisces: a protein sequence culling server. Bioinformatics. 2003;19(12):1589−91.

[5] Zhou J, Troyanskaya O. Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. In: Proceedings of the 31st international conference on machine learning (ICML-14); 2014. p. 745–53.

[6] Wang, Sheng and Peng, Jian and Ma, Jianzhu and Xu, Jinbo. Protein secondary structure prediction using deep convolutional neural fields. In Scientific reports, 2016 6th volume, Nature Publishing Group, pages 18962.

[7] https://www.biorxiv.org/content/biorxiv/early/2019/12/12/871921.full.pdf

[8] https://github.com/taneishi/CB513_dataset