Final Year Project Report

Large Scale Prediction of Contact Partners in Proteins by Deep Neural Networks and Computational Analysis of Coevolution

Jonathan Sweeney

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Gianluca Pollastri

UCD School of Computer Science

University College Dublin

April 6, 2018

Project Specification

General Information

Proteins are the basis of life and over the last several years we have learned a great deal of information about them through genome sequencing projects and other massive-scale experiments. However the most important aspect of proteins (the three-dimensional structure they assume once they are synthesised) remains elusive and the experimental techniques to reveal it have not scaled up as quickly as the techniques that tell us about other aspects of proteins, such as their basic components, presence or absence in a sample, etc. Functional characterisation of proteins is also progressing slowly, but the detection of functional sites that play a significant role in proteins is highly valuable.

Many different types of interactions allow proteins to assume their native shapes. Whatever these interactions may be, they can all be represented as being contacts. If a protein were a sequence of beads on a tangled necklace, if one observes the necklace any two beads can be described as being in contact (they touch each other) or not (they are apart), regardless of whether the ones that are in contact are glued to each other or just happen to be next to each other because of the overall shape of the tangled necklace. Quite interestingly, if we know which pairs of beads touch each other, the shape of the necklace usually follows. Knowing the set of contacts yields the structure.

The aim of this project will be to predict the propensity of pairs of amino acids in proteins to be in contact. While the problem in its broadest form is very challenging, the student will look at simplified versions of it, e.g. only contacts between amino acids (beads) that are a few positions apart in the protein/necklace will be considered, significantly reducing the search space of the problem. Evolutionary information in the form of multiple similar proteins will also be considered. In essence, the student will be given a snapshot of co-evolution to work with (many similarly tangled necklaces) and will be able to extract information and patterns from all of them together rather than just one at a time. The presence or absence of contacts will be predicted by powerful Deep Neural Networks using Deep Learning algorithms.

The student will be provided with C++ software implementing artificial neural networks of any desired depth. Raw datasets of proteins of known structure will also be provided, but these will need to be extensively analysed and parsed by the student. After this, the student will have to run a number of large scale tests using networks of different depths and different training algorithms on the data.

Core: Process and analyse provided protein data sets, get the neural network code to work, run at least one full training.

Advanced: Run a comprehensive set of tests exploring different network sizes and architectures.

Build a full system and compare it to the state of the art.

Reading:

https://en.wikipedia.org/wiki/Protein_structure_prediction

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4894841/

Abstract

The goal of this project was to develop a predictive system to determine the likelihood of certain amino-acids to be in contact within a protein chain, improving and informing researchers' understanding of the protein's overall structure. This information is vital to many areas of research and development, including the treatment of several degenerative diseases.

This system was achieved by training a deep-learning neural network using different architectures and configurations to use more easily-obtained knowledge of the protein to make a prediction of the structures within the chain. The final system design performed with an average prediction accuracy above 83% on unseen test data, comparable with the level of performance of current research. This system can now be made available for ongoing use in the research community.

Acknowledgments

I'd like to thank my supervisor, Dr. Gianluca Pollastri, for all his help and guidance throughout the project.

Table of Contents

Chapter 1: **Introduction**

This project aims to improve the determination of the 3d structures of proteins by building a system to predict the likelihood of certain amino-acids within the protein chain being in close contact with one another. Research techniques, both experimental and computational, to determine most aspects of proteins have far outstripped our ability to find their structure. This inability severely limits the potential of efficiently developing treatments and drugs for many protein-folding diseases, such as Alzheimers and Parkinsons.[1] This project addresses this problem through a predictive system which uses the more easily-obtained knowledge of the protein to improve our understanding of its shape.

The problem of determining protein folds has been an ongoing challenge from as early as the 1970s,[2] and though methods have improved considerably since then, progress still lags far behind the analysation of other protein attributes. Of the more than 100,000,000 unlabelled protein sequences known to the research community, less than 0.1% have been structurally solved for effective use in science, engineering, and medicine. All of these areas, and many others, would benefit from improved prediction of protein structures.

In recent years, deep-learning neural networks and other machine learning methodologies have provided huge advancement in protein folding[3]. Though prediction of functional meta-structures within protein chains has also proven difficult, determining secondary and tertiary structures has proven to be a viable stepping-stone to unlocking the full shape of a protein. While this problem is still immensely complicated, this project will focus on examining and exploiting the relationship between the easier-to-find primary sequence of amino acids and the occurrences of $\beta - sheet$ structures, which can in turn be used to greatly simplify the path to the overall protein shape.[4]

The project was divided into several stages, each of which followed on to the next.

- Parsing of raw datasets of fully known protein descriptions, and extraction of the relevant information.
- Selection and building of different training datasets from the extracted attributes using different parameters and balances of classifications.
- Running of the neural network using different inputs and architectures while testing and evaluating the performance of each configuration and choice of training set.
- Determination of the most effective setup of the neural network and training sets through tests and comparisons between previous results.
- Comparison of final configuration performance to current predicative systems.

The structure of this report will be to first detail some of the background research carried out on currently-existing beta-sheet prediction systems, followed by the theory required to complete the project. Next it will detail the project approach to analysing and parsing the raw data, before giving an overview of the implementation and training process carried out and the challenges encountered. Finally it will outline the evaluations performed, compare the results to the current state of the art, and give some applications and extensions that could improve its use in the future.

Chapter 2: **Background Research**

2.1 Similar Existing Research and Systems

The first step in proceeding with this project was to look into similar predictive systems and research already available. This section details the relevant aspects of those systems and and any crossover or diversions they may have with this project.

- *HyperChem Homology Modelling Professional*

  HyperChem[5] is a commercially offered algorithm that makes use of observed frequencies of amino acid residues to make predictions on the occurrences of secondary structure. While its algorithmic approach is a good example of a non-neural net prediction system, the fact that it is commercial and relatively old (2002) means there is not much crossover between it and the system built during this project.

- *APSSP*

  The Advanced Protein Secondary Structure Prediction Server[6] is a neural network implementation of a similar secondary structure prediction problem to the one tackled by this project. Its makes use of a nearest neighbour algorithm within the decision process.

- *SPIDER2*

  Sequence-based Prediction of Local and Nonlocal Structural Features for Proteins[7] is considered one of the most accurate prediction models that implements a Deep Learning Neural Network. As it looks for structure beyond the secondary structures such as local backbone angles and accessible surface area, it is mostly out of scope of this project. However, it's ability to make comparatively comprehensive predictions using the same input as this project shows the potential of DNN to make major advancements in protein folding.

- *PaleAle 4.0*

  Porter PaleAle 4.0[8] is the UCD-developed implementation of DNN secondary structure prediction system. While it has a large crossover in purpose with the aim of this project, many of the methods it uses (Expanded amino-acid alphabet, double filtering of results, consideration of full sequences) are well beyond the scope of this project. However, its use of bidirectional neural networks and public training set gives an indication of the possible extensions and further work that could be done on the final system produced in this project. Porter is just one of a series of prediction services for different aspects of proteins' structural and behavioural information made available on http://distillf.ucd.ie.

2.2 Theory

This section explains the molecular and technical theory that was required for this project. The first step when researching the theory behind this project was to establish the current scientific methods of protein modelling and understand the relationships within a protein chain that would form the basis of the project's approach.

The motivation behind using predictive systems for protein modelling in the first place is due to the enormous cost of experimentally synthesising protein chains in a laboratory setting in order to examine their shape and properties. While there have been recent advancements in this area, modern techniques still often rely on crystallography-based approaches, requiring a very high level of purity and isolation of the desired protein molecules.[9]

Even once the structure has been crystallised, long chains of polypeptides (often consisting of hundreds of amino acids) still must be formed through protein-protein interactions within a medium such as yeast or other host organisms before their structures and interactions can be analysed using X-ray crystallography.[10]
While this method often yields detailed and highly accurate results, X-ray crystallography can have difficulty identifying some types of errors in the crystallisation process that result in differences between the synthesised protein and the naturally-occurring target of the researcher.[11] These mistakes can have disastrous effects when incorrect information is used in the manufacture of designer drugs and pharmaceuticals.

Due to these limitations and the high cost (in both time and materials) of chemical synthesis, as well as recent improvements in machine learning techniques, the computational predictive approach to protein modelling has come to the forefront as the most viable path to effectively identify the enormous corpus of unlabelled sequences.
Additionally, as several methods have been developed that incorporate knowledge of underlying substructures within a protein chain into the physical examination process described above, the inexpensive generation of reliable structural data to serve as an intermediate stepping stone between the unlabelled sequences and a full model of the protein is extremely valuable.[12] [13]

2.2.1 Structures Within Proteins

Proteins consist of a series of molecules called amino acids joined one after another in a chain of arbitrary length by peptide bonds. As there exist just 20 amino acids from which all proteins are built, much of a protein's properties and behaviours are determined by the linear sequence of amino acids in its chain, called its primary structure.

When a protein is synthesised, the interaction between the various chemistries of neighbouring amino acids causes bends and folds along the chain, forming the complex 3D structure that dictates how the protein will interact with other molecules. As it does so, parts of disparate amino acids from different sections of the chain interact with one another through mechanisms such as hydrogen bonding and weak Van der Waals forces, forming secondary substructures such as α-helices and β-sheets. These secondary structures further influence the complex arrangement of the protein's final shape as they constrain the possible combinations of folds along the chain.[14]

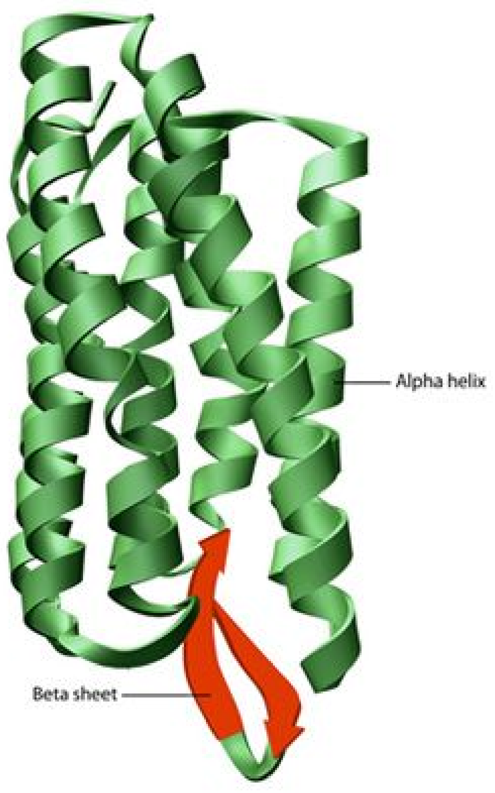


Figure 2.1: Example of a secondary structure within a fully-folded protein.[15]

In the above Figure 2.1 a large part of the chain has settled in α-helices formations, represented by a series of loops. The pair of directed arrows towards the bottom indicates the occurrence of two β-strands in proximity to each other, forming a β-sheet.

2.2.2 Beta Sheet Secondary Structure

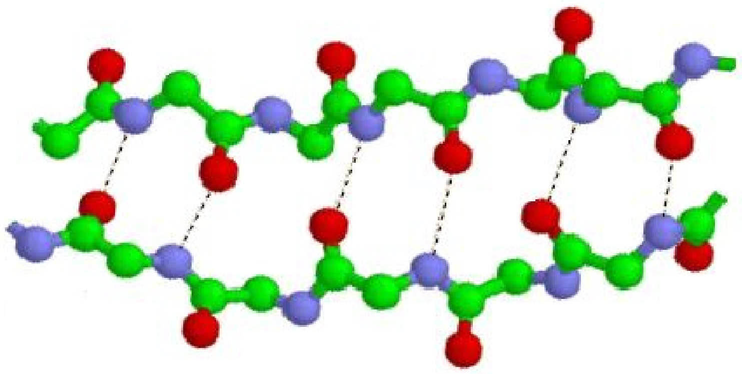


Figure 2.2: Diagram showing contact partners present in β-strand structures.[16]

As the scope of the project focusses on the prediction of amino-acid contact partners in different β-strands within given protein chains, it is important to recognise their characteristics.

As a protein chain folds over on itself as described in the previous section, the hydrogen bonding that occurs between separate amino acids can cause sections of the chain called β-strands to "line up" with one another such that the molecules within a strand interact strongly with their opposing "contact partner" molecules in different strands. This action forms the β-sheet structures found within the shape of protein chains. As shown in Figure 2.2, each amino acid within a β-strand will interact most strongly with the amino acid directly opposite it across the divide that the shape of the protein has created.

β-sheet structures are comprised of two or more β-strands in the chain running parallel or antiparallel to one another.

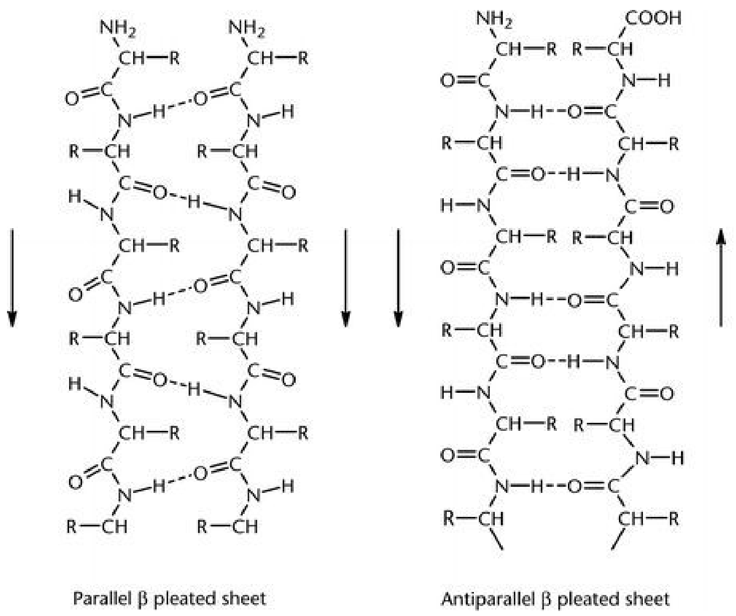


Figure 2.3: Diagram showing the different formations of parallel vs antiparallel β-sheets.[17]

2.2.3 Finding overall structure

The interactions between each amino-acid in the β-sheet substructure in part dictates the shape of the overall structure of the protein. The identification of secondary structures is a stepping-stone to fully modelling the folded chain.[18] While it is rare that a protein can be fully solved from the knowledge of its primary and secondary structures, understanding the intra-chain molecular interactions constrains the arrangement to certain shapes and greatly reduces the number of possible combinations of positions of the protein. This information can then be combined with other knowledge of the protein or with other analytical processes to achieve a solved protein model.[19]

2.2.4 Sliding Context Window Approach

The goal of this project was to build a predictive system capable of identifying pairs of amino acids with a high probability of being in contact within a protein chain. As the information given as input to the system could not be more descriptive than the protein's primary structure, any extra information given to the neural network apart from the contact partners themselves had to also come from the primary structure alone.

As there are only 20 types of amino acids found in protein chains, the neural network could not effectively learn to recognise features and patterns within its inputs if given only the amino acids in contact with one another. To reduce the massive amounts of false-positive and false-negative associations made by the system, the primary structure of protein contact partners had to be examined in a wider context, taking into account the chain neighbours on either side of the amino acids in question.

In order to accomplish this, a sliding context window of a given length needed to be implemented in any parsing script. The sliding window would pass over the primary protein structure until it identified a contact partner in its central position. The information describing the amino acids inside the window on either side of the contact partner would then be captured along with the central amino acid.

The addition of this extra information would give the network the opportunity to learn from the more varied patterns and sequences now present in its input.

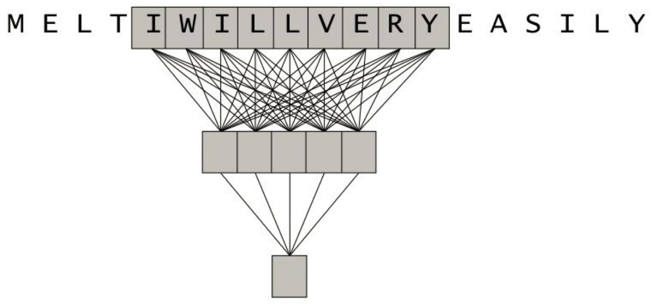


Figure 2.4: Figure portraying the sliding-context window model.[20]

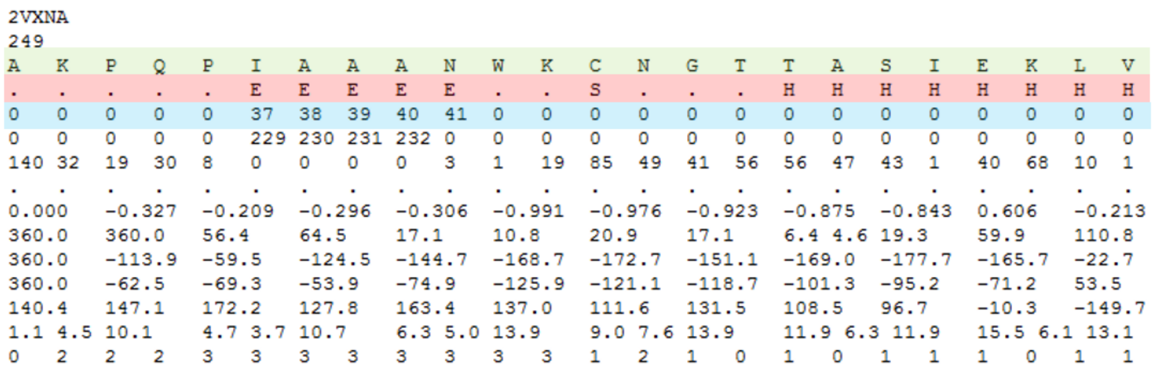
This approach has been implemented in research systems on protein structure prediction from as early as 1988, when the sliding window architecture of a re-purposed text-to-speech neural network was used to predict the occurrences of secondary structures, much like this project sought to do.[21] The concept behind this approach is still used in protein prediction and research today.[22]

Chapter 3: **Project Approach and Design**

This chapter details the work done to formally establish the approach that lead to the final predictive system. First it describes the examination of the raw datasets of solved, fully labelled proteins. Then it formally explains the input model which would be used to train and test the feed-forward neural network, detailing the process that would be used to parse the data. Next it goes through the textual encoding process required to prepare the parsed data for input in the neural network. Finally it gives an short overview of the generation of statistics files used to better understand the distribution of the relevant structures within the data.

3.1 Examining Raw Datasets

The first step in understanding the work required to run full trainings of the neural network was to analyse the sets of protein data provided. All 16,000 proteins in the combined corpus of the provided data files were fully labelled with primary, secondary, and tertiary structural data. This information is far too complex and detailed to be of any use in a machine learning approach, so it was necessary to fully understand the value and relevance of each trait before beginning to parse the datasets.



```
2VXNA
249
A   K   P   Q   P   I   A   A   A   N   W   K   C   N   G   T   T   A   S   I   E   K   L   V
.   .   .   .   .   E   E   E   E   E   .   .   S   .   .   .   H   H   H   H   H   H   H   H
0   0   0   0   0   37  38  39  40  41  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   229 230 231 232 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
140 32  19  30  8   0   0   0   0   3   1   19  85  49  41  56  56  47  43  1   40  68  10  1
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
0.000   -0.327  -0.209  -0.296  -0.306  -0.991  -0.976  -0.923  -0.875  -0.843  0.606   -0.213
360.0   360.0   56.4    64.5    17.1    10.8    20.9    17.1    6.4 4.6 19.3    59.9    110.8
360.0   -113.9  -59.5   -124.5  -144.7  -168.7  -172.7  -151.1  -169.0  -177.7  -165.7  -22.7
360.0   -62.5   -69.3   -53.9   -74.9   -125.9  -121.1  -118.7  -101.3  -95.2   -71.2   53.5
140.4   147.1   172.2   127.8   163.4   137.0   111.6   131.5   108.5   96.7    -10.3   -149.7
1.1 4.5 10.1    4.7 3.7 10.7    6.3 5.0 13.9    9.0 7.6 13.9    11.9 6.3 11.9   15.5 6.1 13.1
0   2   2   2   3   3   3   3   3   3   3   3   1   2   1   0   1   0   1   1   1   0   1   1
```

Figure 3.1: Example of a fully labelled protein with primary and secondary structures highlighted.

As the goal of this project was to train a system to predict the occurrence of secondary structures within a protein given its easier-to-find primary structure, extracting the relevant information from the detailed raw data was key. The sections of each labelled protein's data that were most pertinent to this project are highlighted in the above Figure 3.1.

- The first relevant aspect of a protein is its primary structure, highlighted in green, and is found on the third line of each entry in the raw protein sets. As mentioned in the background theory given in Chapter 2, the primary structure of a protein describes the sequence of amino acids in its chain. The pairs of these amino acids which are in contact within β-sheets comprise the input to the neural network.

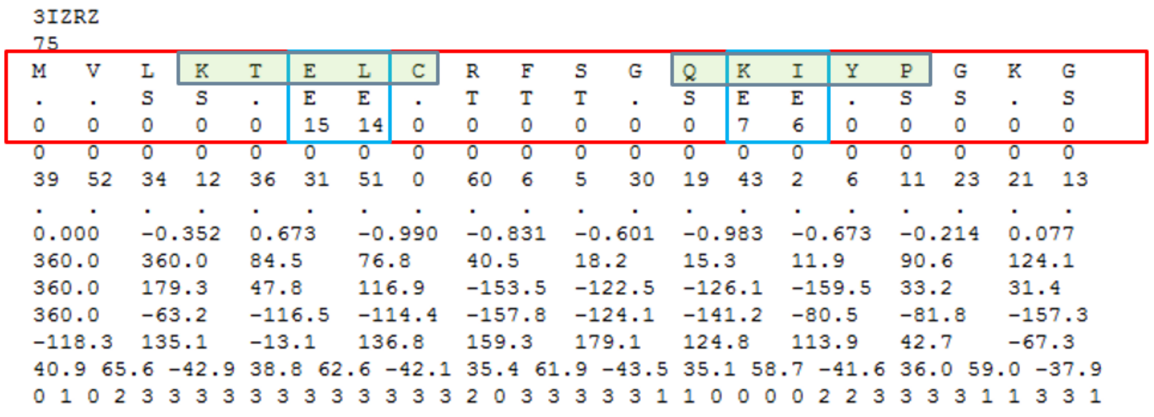
- The next element to examine is provided on the fourth line of Figure 3.1 and is highlighted in red. This attribute identifies the type of secondary structure (if any) that contains the corresponding amino acid on the previous line.
- Lastly, the fifth line highlighted in blue gives the index of an amino acid's contact partner in the chain. This only shows a meaningful value when the corresponding amino acid appears in a secondary structure where molecules can form contact partners, but as this applies to the molecular relationships within β-sheet formations it is important to capture this information.

3.2 Establishing Model for Training and Testing Sets

Once the features of the datasets that were most important to the project had been identified, the next step was to formally define the input/output model that the neural network would be configured to. This specification would then be used when writing scripts to parse and generate training and testing files.

3.2.1 Input Model

Through examining the raw datasets as well as reading the background theory to this project (particularly the research referenced in Section2.2.4) it was clear that the input model of the neural network would consist of 2 series of one or more amino acids. The central entry in each sequence would be the two amino acids thought to be in contact within a β-sheet structure, while the other acids (if any) on either side of the contact partners would represent the context within the chain in which those contact partners were found.



```
3IZRZ
75
M     V     L     K     T     E     L     C     R     F     S     G     Q     K     I     Y     P     G     K     G
.     .     S     S     .     E     E     .     T     T     T     .     S     E     E     .     S     S     .     S
0     0     0     0     0     15    14    0     0     0     0     0     0     7     6     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
39    52    34    12    36    31    51    0     60    6     5     30    19    43    2     6     11    23    21    13
.     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .
0.000       -0.352      0.673       -0.990      -0.831      -0.601      -0.983      -0.673      -0.214      0.077
360.0       360.0       84.5        76.8        40.5        18.2        15.3        11.9        90.6        124.1
360.0       179.3       47.8        116.9       -153.5      -122.5      -126.1      -159.5      33.2        31.4
360.0       -63.2       -116.5      -114.4      -157.8      -124.1      -141.2      -80.5       -81.8       -157.3
-118.3      135.1       -13.1       136.8       159.3       179.1       124.8       113.9       42.7        -67.3
40.9 65.6 -42.9 38.8 62.6 -42.1 35.4 61.9 -43.5 35.1 58.7 -41.6 36.0 59.0 -37.9
0 1 0 2 3 3 3 3 3 3 3 3 3 3 3 3 2 0 3 3 3 3 3 1 1 0 0 0 0 2 2 3 3 3 3 1 1 3 3 1
```

Figure 3.2: Example of a fully labelled protein with contact partners and context window highlighted.

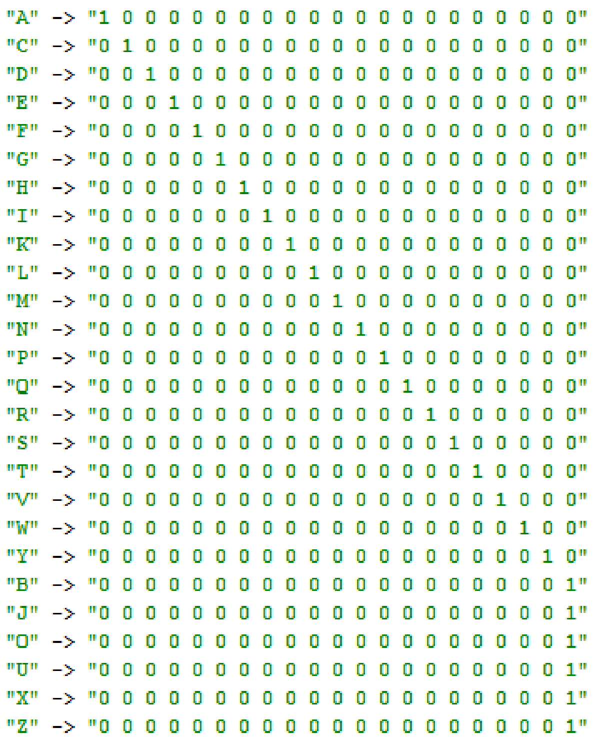
As shown in the above Figure 3.2, the focus of the input-generation script would be on the first 3 lines of structural data in each protein, indicated in red. The script would then search along the protein chain until it found an occurrence of an amino acid in a β-sheet structure (highlighted in blue), before reading the value of the index of that acid's contact partner (given on line 3 of the structural data, as detailed in Section 3.1). Once the pair of contact partners had been found, a sliding context window of arbitrary length (in the case of Figure 3.2, 5 amino acids wide, highlighted in green) would centre on each contact partner and capture the data within the window's borders to be placed in the input dataset along with a classification identifying the pair as being in contact.

As the generated training and testing sets had to have close to a 50/50 distribution of positive and negative examples for the network to most effectively learn, it would be also necessary to write the script functionality such that the number of contact partners found in a set would be recorded and matched by negative examples comprised of random pairings of non-contact partners taken from the same raw protein data.

3.2.2 Encoding the Input

In the primary structure data of each protein, the 20 different amino acids that combine to form the protein chain are represented by different letters of the alphabet. Due to the computational nature of neural networks, it was necessary to re-encode these inputs in a numerical representation. While the simplest approach would have been to simply number the acids from 1 to 20 and, this would have introduced ordinal information and relationships where none had existed before. The acid represented by A may have been assigned a certain relationship to C by the neural network, due to nothing more than their proximity in the alphabet.

To eliminate this issue, a one-hot encoding method was chosen to represent the amino acid data in the network input. Each amino acid type was assigned a unique sequence consisting of 21 characters: twenty of them with a value of 0 and one set to a value of 1 (The twenty-first code was used to represent unidentifiable or extremely rare molecules found within the protein chain). As these could be input into the neural network as a long series of binary values, ordinal relationships like those found within natural numbers were avoided.[23] Once this mapping functionality was included in the input set generation scripts, the training could begin.

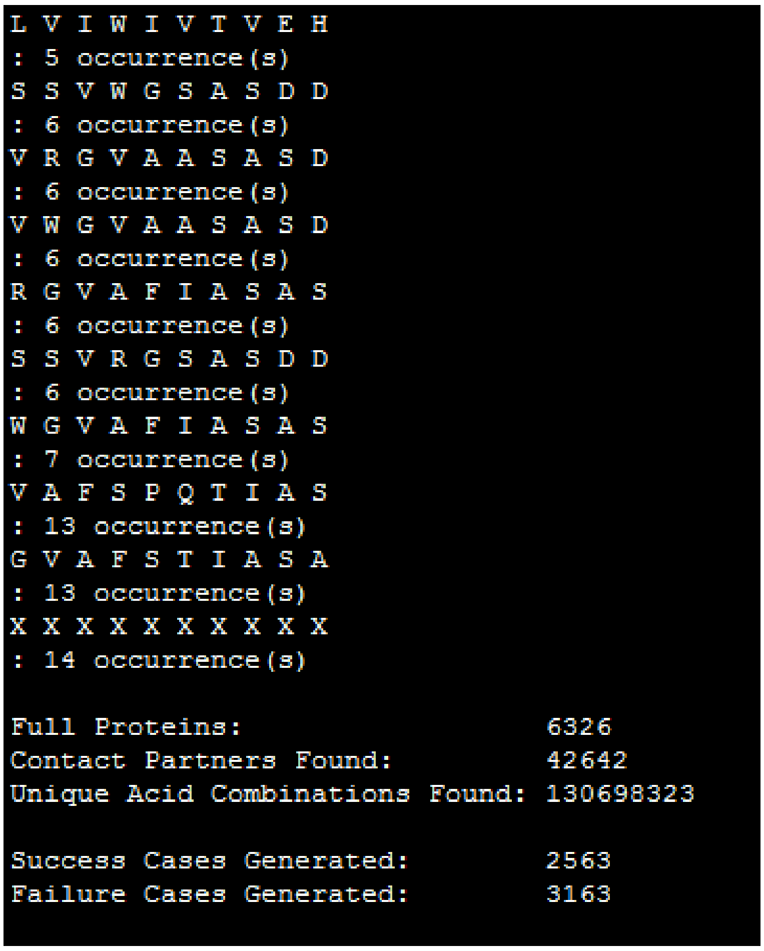


```
"A" -> "1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"C" -> "0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"D" -> "0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"E" -> "0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"F" -> "0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"G" -> "0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"H" -> "0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"I" -> "0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0"
"K" -> "0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0"
"L" -> "0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0"
"M" -> "0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0"
"N" -> "0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0"
"P" -> "0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0"
"Q" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0"
"R" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0"
"S" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0"
"T" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0"
"V" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0"
"W" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0"
"Y" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0"
"B" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
"J" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
"O" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
"U" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
"X" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
"Z" -> "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1"
```

Figure 3.3: Mapping of amino acids ->One-Hot Encoding format.

3.3 Statistics

As part of the examination of the raw protein files and generation of the input sets, functionality was added to the parsing scripts to generate statistics describing the basic metadata of the dataset as well as the distribution of the proteins, amino acids, and contact partners found within it.



```
L V I W I V T V E H
: 5 occurrence(s)
S S V W G S A S D D
: 6 occurrence(s)
V R G V A A S A S D
: 6 occurrence(s)
V W G V A A S A S D
: 6 occurrence(s)
R G V A F I A S A S
: 6 occurrence(s)
S S V R G S A S D D
: 6 occurrence(s)
W G V A F I A S A S
: 7 occurrence(s)
V A F S P Q T I A S
: 13 occurrence(s)
G V A F S T I A S A
: 13 occurrence(s)
X X X X X X X X X X
: 14 occurrence(s)

Full Proteins:                   6326
Contact Partners Found:          42642
Unique Acid Combinations Found:  130698323

Success Cases Generated:         2563
Failure Cases Generated:         3163
```

Figure 3.4: Example of output of statistics generation code run with a context window of size 5.

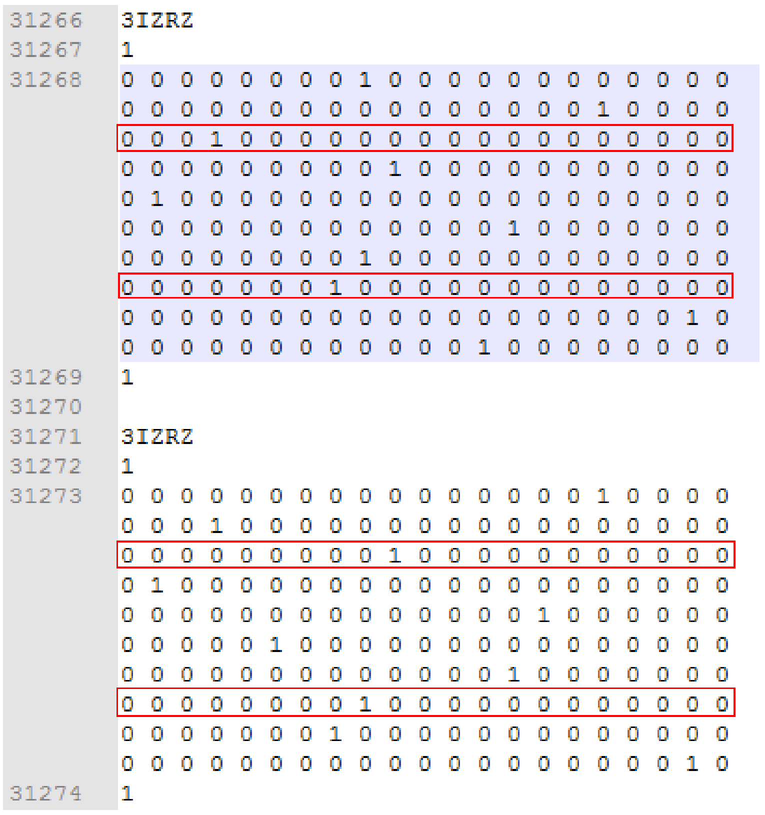
Chapter 4: **Implementation**

This chapter discusses the implementation of the project. First it describes the parsing of the labelled datasets using the model described in chapter 3, before detailing the methodical and repeated process of training the Neural Network under various configurations. Finally, it goes through the collection and analytical processing of the data, and the challenges that were encountered during these steps.

4.1 Building the Training and Testing Datasets

With the input and output formats clearly defined as detailed in Section 3.2.1, a parsing script was written to generate network training and testing datasets from the raw input files. Java was chosen as the script implementation language as this allowed the protein information to be read into the program in the form of strictly-typed data structures. This simplified the manipulation and construction of the full sets of various window sizes, as well as allowing for the analytical processing shown in Section 3.3. Since the 5-fold datasets each needed to be generated only once, the slightly higher performance cost of using Java over a different runtime environment was not an issue.

As described in the dataset generation approach in Section 3.2.1, the script was constructed to parse the raw protein data files, find all pairs of contact partners, and then capture the variable-length context in which each occurred within the chain before encoding them using the One-Hot model.
Once it had found all relevant instances of interaction between amino acids, it then constructed an equal number of negative examples in the set by pairing completely unconnected amino acids together and recording the entry as a failing class example. This allowed the neural network to properly learn to differentiate between the patterns found around the occurrences of contact partners and non-contact partners in the chain.



```
3IZRZ
1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1

3IZRZ
1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1
```

Figure 4.1: Example of the generated input data, taken from the 5-acid context window dataset.

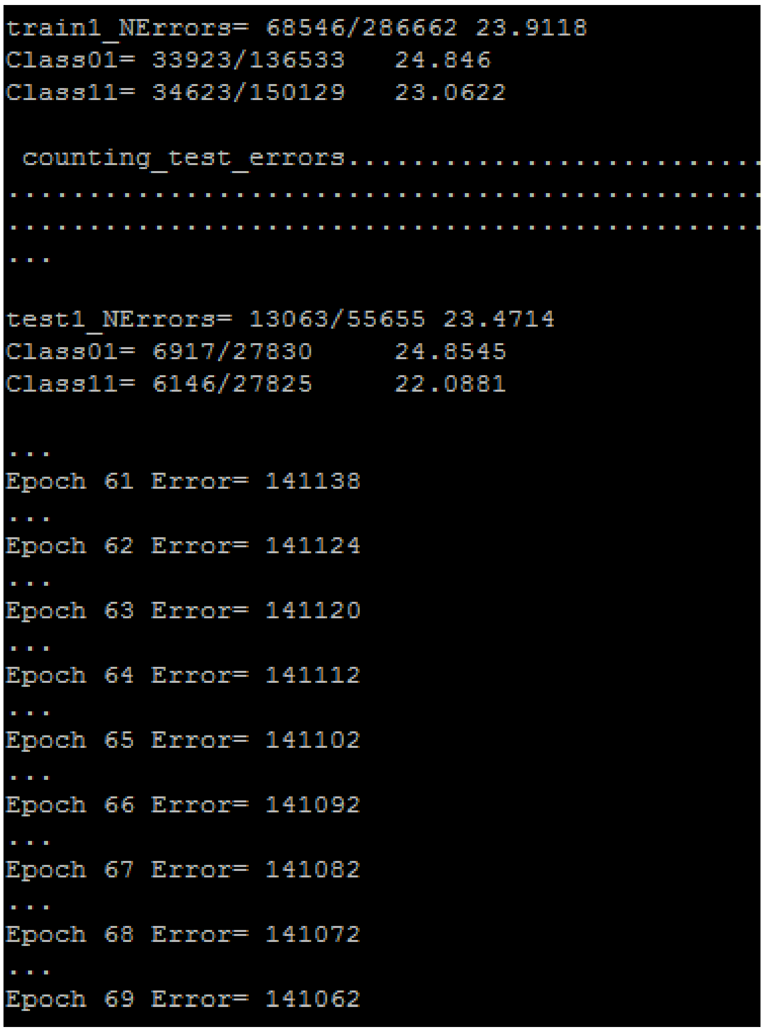
In Figure 4.1 shown above, the primary input to the network consisted of a set of amino acids encoded to remove any unintended relationships between molecules originally represented with an ordinal alphabet of letters. Underneath each set of amino acid inputs, the classification value indicates the presence of contact partners within the strands. These inputs are the result of the raw protein data shown in Figure 3.2 being parsed by the script when given a sliding context window size of 5. As both of these examples contain contact partners (highlighted in red), both are assigned an output classification of 1.

4.2 Running Training Configurations

The majority of training runs of the neural network code were carried out on the distillf Ubuntu server in the UCD Science Building, accessed via SSH.

4.2.1 Initial Network Training

Initially, a small number of preliminary training datasets were constructed and run on the network to confirm that the input model described in Section 3.2.1 was correctly designed. While no attempt was made at this stage to vary the configuration or architecture of the system, the format and nature of the network output was carefully noted so that scripts to properly parse the network output messages could be written in preparation of the experimental phase



```
train1_NErrors= 68546/286662 23.9118
Class01= 33923/136533   24.846
Class11= 34623/150129   23.0622

 counting_test_errors.......................
.............................................
.............................................
...

test1_NErrors= 13063/55655 23.4714
Class01= 6917/27830     24.8545
Class11= 6146/27825     22.0881

...
Epoch 61 Error= 141138
...
Epoch 62 Error= 141124
...
Epoch 63 Error= 141120
...
Epoch 64 Error= 141112
...
Epoch 65 Error= 141102
...
Epoch 66 Error= 141092
...
Epoch 67 Error= 141082
...
Epoch 68 Error= 141072
...
Epoch 69 Error= 141062
```

Figure 4.2: A snapshot of the neural network output while a training run is in progress.

4.2.2 Experimenting with Configurations

Once the input model and training process were fully understood and tested, and the 5-fold training/testing datasets had been generated for every context window up to a length of 21, the major investigative process of the project was carried out. Over several weeks, the network was trained using many configurations and input architectures in different combinations.

Initially, all trainings using a given context window size were carried out individually on each of the 5 datasets generated for that window size. The results for each were then graphed and closely analysed to determine the most accurate performance mark achieved by each of the 5 folds, which were averaged to give a final measure of performance for that size of input under a given network configuration. That architecture and input size was then kept consistent while network parameters such as the learning rate, the size of each batch block, the adaptive threshold, and the number of epochs for which the training was allowed to run were varied and recorded. This methodical, scientific approach to the training process ensured that the results would not become inaccurate or misleading by multiple variables being altered at once.

This approach worked well at first, as the accuracy of the overall system improved as expected with each broadening of the context window size. The system's response to the variations in each of the network configurations was also in line with expectations, and the trends in each variable remained consistent across the changing window sizes.

It appeared that the impact of the adaptive threshold parameter, once set over a very basic level, was negligible on system performance due to the quality and quantity of the provided data. Similarly the effects of variance in the batch block size and epoch number proved heavily dependant on each other, leaving little reason to deviate from the standard practice of balancing the length of the training directly on the performance of a particular architecture.

However, as the context window size grew past 7 and 9 amino acids per contact partner (requiring 294 and 378 input nodes respectively), the performance of the system under all internal configurations began to suffer. After examination of the trends in error values over the course of each training run, and of the discrepancy between the system's predictive accuracy on its own training data and the unseen testing data, it became clear that addition of extra context to the networks input was causing considerable overfitting within the network. The increasingly complex patterns and associations that the improved context allowed the network to learn caused the model to become overly suited to its training data and unable to effectively generalise and classify the input data it had not seen. As each successive window size added more context, the problem had grown very severe by the time the last group of datasets was reached.

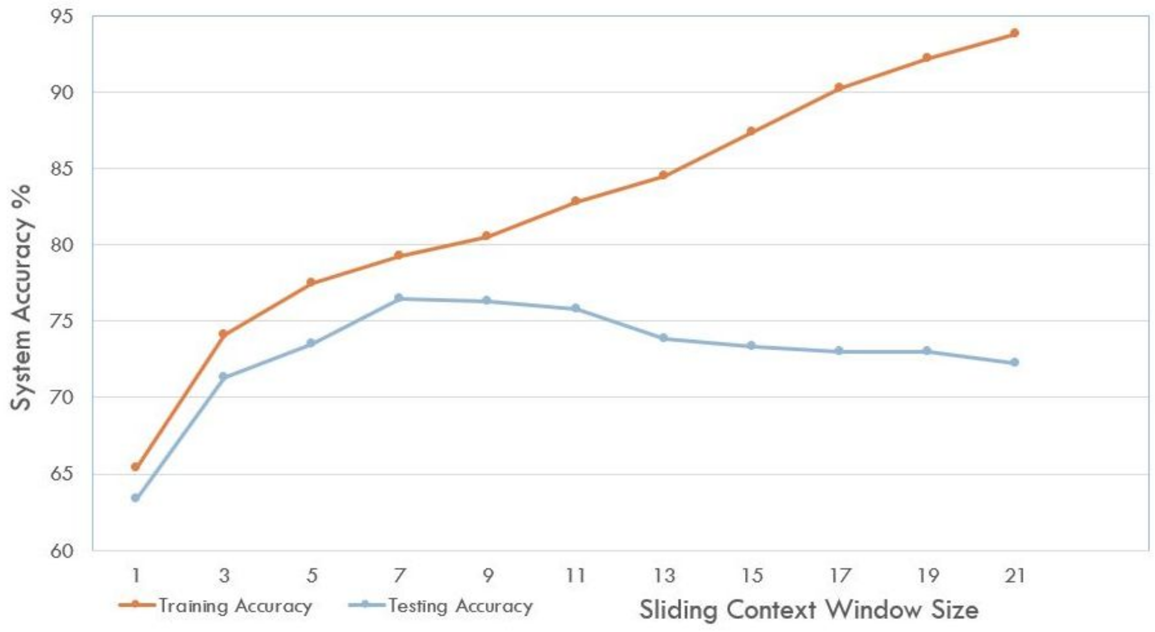


Figure 4.3: Comparison of training/test accuracies over growing window lengths showing severe overfitting in higher contexts.

This issue was remedied by combining two standard approaches to solving overfitting in neural networks. Firstly, the severity of the overfitting was considerably reduced by switching the training process away from averaging the result of separate runs across 5 slices of the data to instead using all available training data to train and test the system in a single run at each window size.[24] The training and testing process using these combined 5-fold datasets was then further refined by examining and varying the number of epochs for which the training was allowed to run. Where applicable, the "stop early"[25] method was employed to further reduce the discrepancy between the training and testing accuracies.
The full effects and reasoning behind these choices is described in detail below in Section 4.4.2, along with a short discussion explaining the emergence of overfitting in the previous training process.

With the overfitting issue resolved, the training process was resumed using the improved, full-fold training datasets to establish the best combination of internal configurations and input layer architecture. On each dataset, the learning rate was the most impactful of the neural network's variable parameters. Even on the best-performing window size of 21, varying the learning rate had a considerable effect on the trends and consistencies of the error values throughout a 1000-epoch training run.

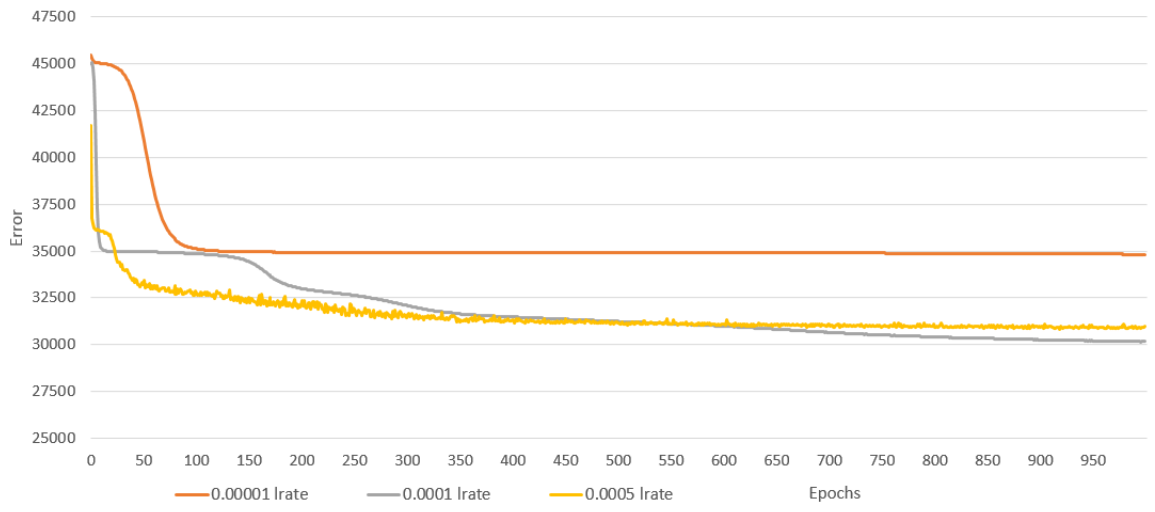


Figure 4.4: Network error values during the course of a 1000-epoch training at 21 acid context window.

While at the end of the training run a learning rate of 0.0001 (grey) was found to be the most effective, a slightly more aggressive learning rate of 0.0005 (yellow) performed better for almost half of the training. However, as variences in network performance from epoch to epoch get more extreme at higher learning rates, the model had less stability than the 0.0001 configuration and never matched te configuration's global minimum. Both approaches outperformed lower learning rates considerably, as shown in Figure 4.4 as the 0.00001 learning rate run clearly became trapped in a local minimum it could not break out of within the first 100 epochs.

This trend held across the range of datasets, with lower learning rates performing poorly and higher rates performing well on less complex sets, but never reaching the predictive accuracy of the medium rates when provided inputs with significant context.

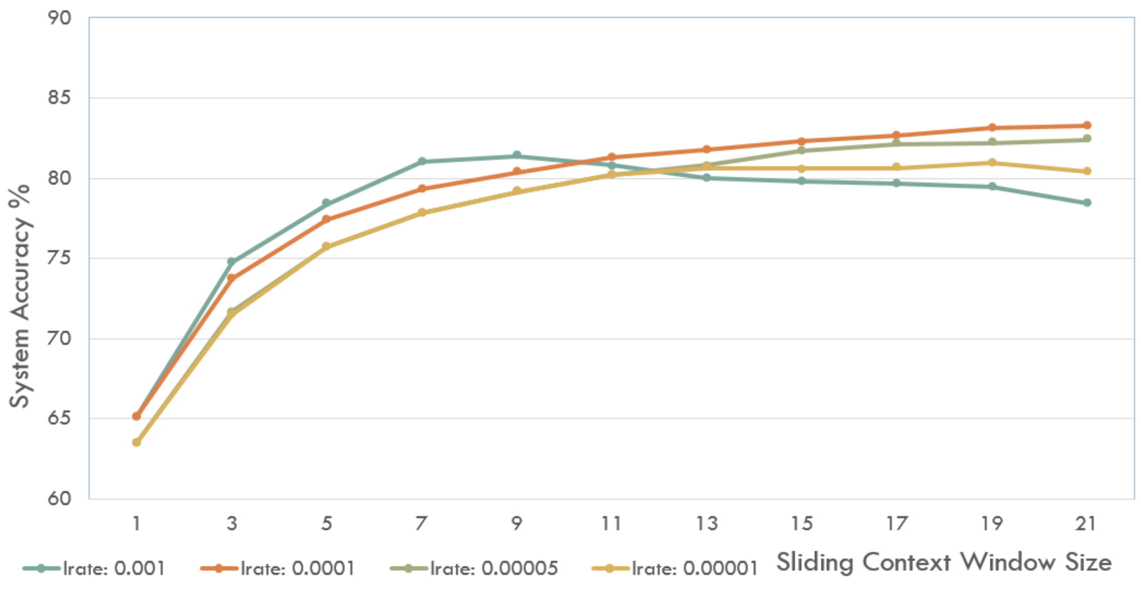


Figure 4.5: Chart comparing lrate value impact across the range of input lengths.

4.3 Collecting and Dissecting Results

After each epoch of the training process, the neural network would output the current network error value. Additionally, after every 10 epochs the total proportions of incorrect predictions made on both training and testing inputs by the network during that span were also output to the console from which the code was run. These outputs were captured and stored in their raw form, before being parsed and prepared for analysis and graphical representation using 2 further Java scripts.

In the final stages of the project, the neural network source code was downloaded from the UCD server and the output functions were edited to better accommodate the graphing process. The system was then compiled and run locally on small training sets to collect needed data and results.

Considerable care was taken when recording the performance of system configuration to not overstate the accuracy achieved.

4.4 Challenges Encountered

Although completing each phase of the project presented its own set of challenges, some issues were of particular importance to the project's success. Detailed below are some of the areas which required extra attention or implementation to resolve.

4.4.1 Validation of Generated Datasets

Due to the enormous importance of correctly generated data in the training process, it was necessary to confirm that the encoded training sets, once constructed, were consistent with the original information found in the raw protein datasets. In addition to the one-hot encoding of the data making it difficult to match the entries by eye, the fact that the full-fold sets grew to hundreds of thousands of input entries made manual checking completely impossible.

In order to verify the validity of the encoded network inputs, another parsing script was written to test the input entries by reading in the finished training datasets and reversing the mapping function originally used to generate them from the raw protein sets. As the only information lost in the generation of the training sets was data irrelevant to purpose of this project, the reverse-mapping produced pairs of amino acids in their full contexts with a binary classification indicating their status as contact partners. The script then compared these decoded entries to the corresponding raw proteins from which they were built and confirmed that there existed molecules with all of their attributes within the raw set.

4.4.2 Overfitting

As mentioned earlier in this chapter, the initial training process to evaluate the performance of the different sliding window sizes was performed individually over a series of five "slices" of each dataset generated by the parsing code.

While this approach worked well for the sets of smaller context windows, adding more information about the surroundings of each contact partner within the chain caused the system to quickly grow too fitted to its own training data. This effect was most severe in the architectures which used the largest amount of context, where the predictive system's accuracy rate actually began to decrease steadily almost from the beginning of its training run.

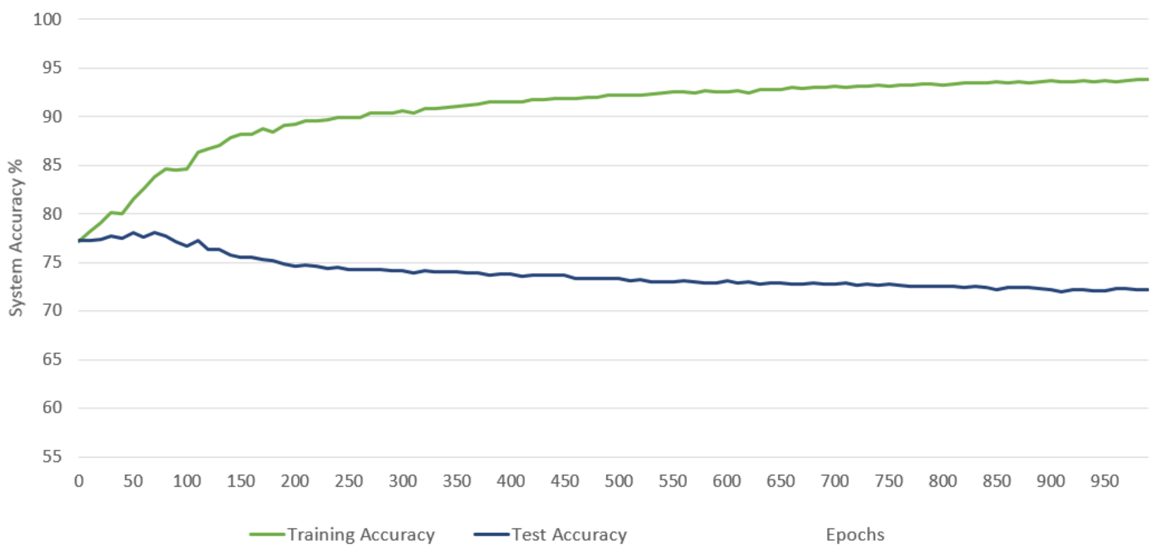


Figure 4.6: Overfitting during the training of a slice of the 21-context window dataset.

The pattern seen in the above Figure 4.6 is a clear indication of overfitting, as the system's accuracy on the training data reached impossibly-high levels by the end of the 1000-epoch training, while the accuracy of the model on unseen test data was actually harmed by the addition of new data, due to its inability to generalise the sequences it had learned.

This problem was reduced mostly by the combination of each of the 5 training/testing folds generated for the different lengths of context windows into one single dataset for each input length. This increased the amount of input training/testing examples that the system saw on each run by a factor of 5, and allowed it to learn to recognise patterns contained in a much more varied and complete set of data. This in turn allowed the network to properly generalise the information found in its inputs and improve the accuracy of the final system at each context window length. The practice of resolving overfitting issues in machine learning by providing the learning system with as much data as possible is sometimes referred to as a "brute force" approach, and while it can make the problem even worse if the root cause lies in the network topology, [26] it can also be a very effective and relatively inexpensive method of improving the network's performance.[24]

Additionally, in cases where the system accuracy seemed to decrease towards the end of the training process, reducing the number of epochs for which the network was trained kept the final error value at its global minimum and further decreased the discrepancies between the training and testing accuracies. This "Early Stopping" approach is a standard and widely employed technique to curb to effects of increased generalisation-error past a certain point of training.[27]

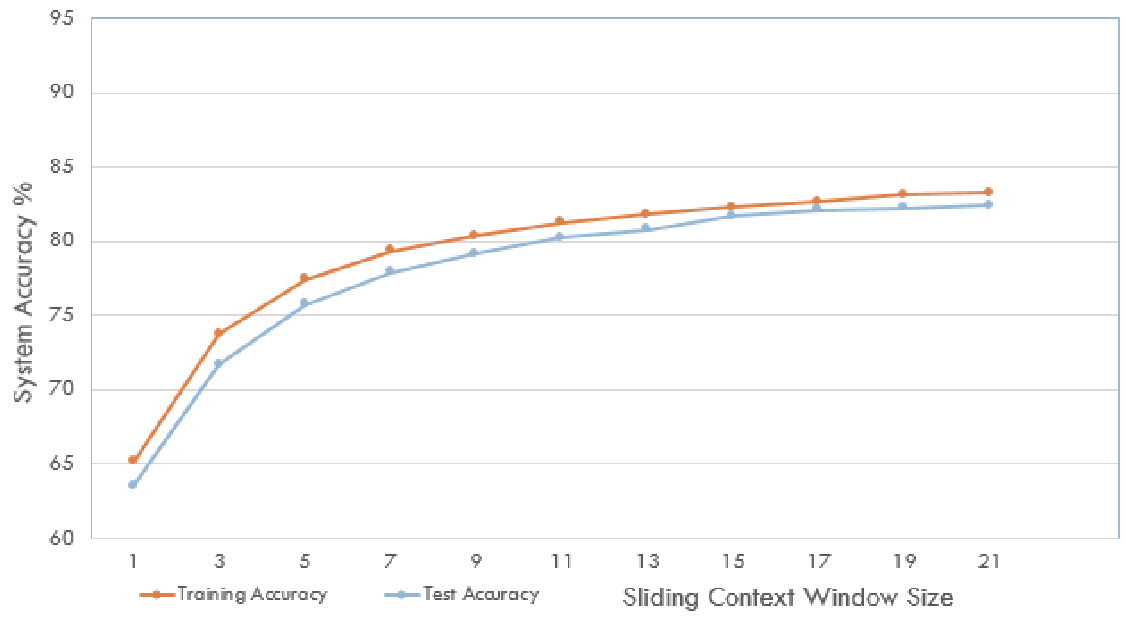


Figure 4.7: Chart comparing the accuracies of system predictions on training and testing data.

As shown in the above Figure 4.7, the approaches described in this section almost completely eliminated the performance gap between predictions on training and test data across all context window lengths. The training accuracy rate no longer exceeded any realistic expectation of the predictive system and the prediction accuracy on test data increased steadily with added context before beginning to plateau once the window size grew too large. This clearly indicated that the predictive system could now properly generalise from its input data.

Chapter 5: **Evaluation**

This chapter details the evaluation of the final system configuration achieved by the project and compares its predictive performance to current research and the state of the art.

5.1 Training with Final System Configuration

The final system was trained over 1000 epochs using a learning rate of 0.0001, an adaptive threshold of 100, and a batch block size of roughly 300 since each full fold dataset contained just over 300,000 training input entries. The neural network was trained using this network configuration on each context window size dataset to establish the best input architecture for the system.

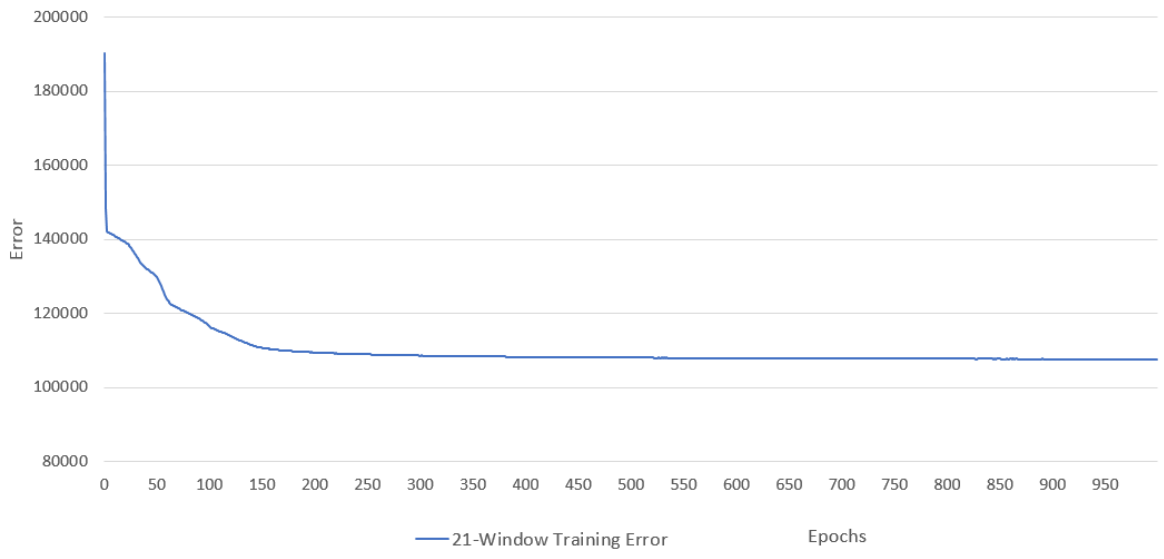


Figure 5.1: Chart displaying the network error value throughout the full 1000-epoch training using the 21-context window dataset.

The above Figure 5.1 shows the consistency and stability achieved in the prediction model during the training run on the 21-context window input data. Although the network error value levelled out considerably with only a small improvement in learning taking place in the latter half of the training, the fact that the error continued to decrease, even slightly, through the whole training showed that the effects of overfitting had been completely negated in the model. Additionally, due to the body of tests and experimental training runs carried out with higher learning rates, it was firmly established that the model wasn't underfitted to the testing data and that the levelling out of the error values was not due to the model becoming stuck in a local minimum.

5.2 Final System Performance

The final system configuration described in the previous section was used to train each combined 5-fold dataset generated for each of the context window sizes. While the improvements in the overall system accuracy began to plateau over the largest context window sizes, the system trained on the 21-window dataset still achieved the highest accuracy at 83.1% when tested on the set of unseen input data.

This figure is comparable to the accuracy levels achieved in similar research which used more complicated models and information for the network input, taking into account features such as multiple sequence alignment to improve predictions.[28][29]. The prediction accuracy accomplished in this project even surpasses several previous systems in the secondary structure prediction research area that had upper limits of accuracy of about 80%.[30]

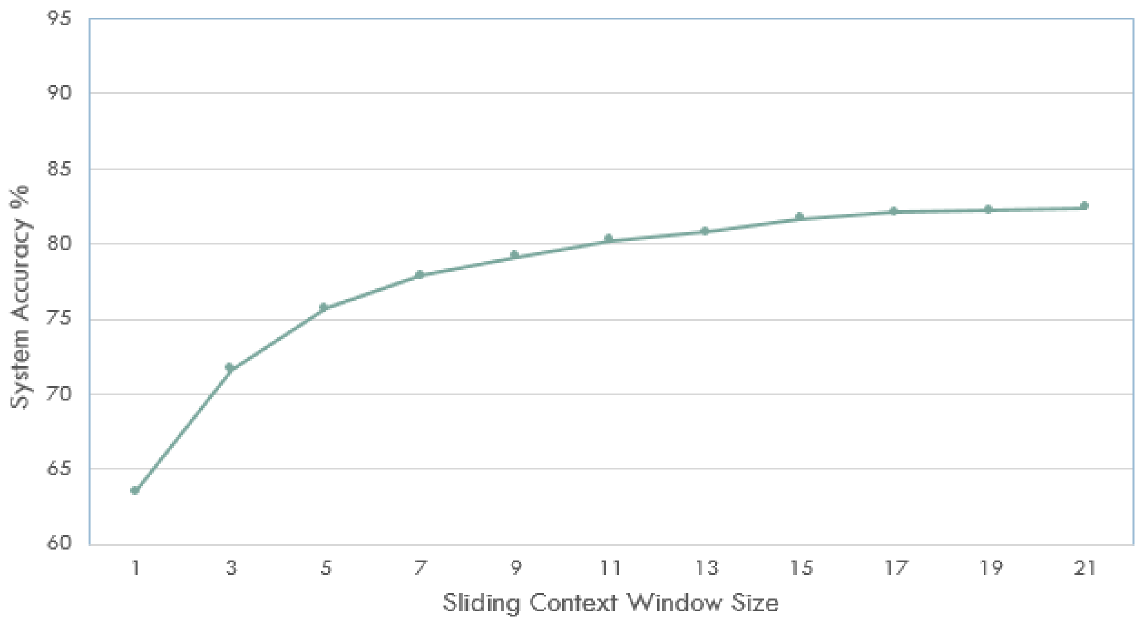


Figure 5.2: Final results of all full datasets using the best-performing neural network configuration.

Taking into account the relatively low amount of knowledge of the target protein required by the system to make a prediction, as well as the supposed functional upper limit of 90% prediction accuracy of β-sheet occurrences,[31] the 83.1% accuracy level reached by this project is acceptably high for public use by the research community.

Chapter 6: **Conclusion**

6.1 Summary

The goal of this project was to develop a predictive system by training a deep neural network to predict the likelihood of a pair of disparate amino acids within a protein chain being in contact with one another. The optimal architecture and network configuration were to be found through a comprehensive series of tests and experimental training runs of the system.

This was achieved by parsing a large corpus of fully solved protein data to generate datasets of amino acid contact partners, along with the variably-sized contexts in which they occurred within the protein chain. This data was then encoded in a format suitable for input into the neural network.

Once the datasets were generated, the network size and architecture, as well as the internal configurations were carefully and methodically varied and all results were recorded. After examining and analysing the output of all combinations and arrangements of the training runs, the best-performing system configuration was identified.

This system was used to train the entire set of data at each level of context around the contact partners. The highest-performing system achieved a predictive accuracy of 83.1% when tested on a set of unseen data, a figure which is comparable with the current state-of-the-art in software within this research area.

6.2 Possible Extensions and Future Work

Given more time, there are many extensions and additions which could be made to this project, due to the enormous scope and complexity of predicting structural aspects of folded proteins. These include:

- Identifying and training the neural network to predict the propensity of amino acids to be part of other types of secondary structures such as α-helices, in addition to the β-sheet prediction achieved in this project.
- Examining other structural features or relationships between similar proteins to further improve upon the 83.1% accuracy achieved in this project.
- Using the predictions made by the system in this project as inputs to other predictive models that use secondary structure information as a stepping-stone to solve more complex structural features within proteins.
- Hosting the final system build on a public facing server for widespread use in research, similar to the services provided on http://distillf.ucd.ie.

Bibliography

[1] Ashraf, Ghulam Md et al. Protein Misfolding and Aggregation in Alzheimers Disease and Type 2 Diabetes Mellitus. (2014)

[2] Peter Y. Chou and Gerald D. Fasman, "Prediction of protein conformation", (1974)

[3] Wei, Leyi, and Quan Zou, Recent Progress in Machine Learning-Based Methods for Protein Fold Recognition. (2016)

[4] Baldi, Pollastri, Andersen, and Brunak, "Protein β-Sheet Partner Prediction by Neural Networks", (2000)

[5] http://www.molfunction.com/bioinformatics.htm (Accessed on 05/04/2018)

[6] "Advanced Protein Secondary Structure Prediction Server", http://crdd.osdd.net/raghava/apssp/ (Accessed on 05/04/2018)

[7] "Sequence-based Prediction of Local and Nonlocal Structural Features for Proteins" http://sparks-lab.org/server/SPIDER2/ (Accessed on 05/04/2018)

[8] "PaleAle 4.0", http://distillf.ucd.ie/porterpaleale/, (Accessed on 05/04/2018)

[9] P. E. Dawson and S. H. Kent, "Synthesis of Native Proteins by Chemical Ligation", (2000)

[10] J. K. Joung, E. I. Ramm, and C. O. Pabo, "A bacterial two-hybrid selection system for studying proteinDNA and proteinprotein interactions" (2000)

[11] J. Claydon, N. Greeves, and S. Warren, "Organic Chemistry", (2012)

[12] G. Scapin, "Structural Biology and Drug Discovery" (2006)

[13] Y. Qi and N. V. Grishin, "Structural Classification of Thioredoxin-Like Fold Proteins" (2005)

[14] "Protein Structures" https://www.nature.com/scitable/topicpage/protein-structure-14122136 (Accessed on 05/04/2018)

[15] http://www.nature.com/scitable/content/ne0000/ne0000/ne0000/ne0000/14711403/U2CP4-2_ProteinStructureDomains_ksm.jpg (Accessed on 05/04/2018)

[16] http://www.web-books.com/MoBio/Free/images/Ch2C7.gif (Accessed on 05/04/2018)

[17] http://edu.isb-sib.ch/pluginfile.php/277/course/section/150/importance_f3.gif (Accessed on 05/04/2018)

[18] K. K Senapati, G. Sahoo and D. Bhaumik, "Algorithm for Predicting Protein Secondary Structure", (2010)

[19] G. Scapin, "Molecular replacement then and now", (2013)

[20] https://image.slidesharecdn.com/jen04talk6-1217966101415381-9/95/disembl-artificial-neural-network-prediction-of-protein-disorder-4-728.jpg?cb=1217940534 (Accessed on 05/04/2018)

[21] N. Qian and T. J. Sejnowski, TJ."Predicting the secondary structure of globular proteins using neural network models", (1988)

[22] H. Kim and H. Park, "Protein secondary structure prediction based on an improved support vector machines approach", (2003)

[23] A. C. H. Choong and N. K. Lee, "Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences using Ordinal versus one-hot Encoding Method", (2017)

[24] "Improve your neural networks", http://adventuresinmachinelearning.com/improve-neural-networks-part-1/ (Accessed on 05/04/2018)

[25] F. Girosi, M. Jones and T. Poggio, "Regularization Theory and Neural Networks Architectures", (2008)

[26] G. N. Karystinos and D. A. Padoso, "Overfitting, Generalization, and Randomly Expanded Training Sets", (2000)

[27] Y. Yao, L. Rosasco, and A. Caponnetto, "Early Stopping in Gradient Descent Learning", (2007)

[28] V. A. Simossis and J. Heringa, "Integrating protein secondary structure prediction and multiple sequence alignment", (2004)

[29] J. A. Cuff G. J. Barton, "Application of multiple sequence alignment profiles to improve protein secondary structure prediction", (2000)

[30] W. Pirovano and J. Heringa, "Protein secondary structure prediction", (2010)

[31] D. Kihara, "The effect of long-range interactions on the secondary structure formation of proteins", (2005)