MSc DISSERTATION ASSESSMENT COVERSHEET

To be completed by student:

Name of student:	Hajar Saihi
MSc Programme:	Bioinformatics
Title of Dissertation:	A Novel Fast Single Sequence Secondary Structure Prediction Algorithm Using Word Embeddings and Deep Neural Networks.
Year of Submission:	2019
Declaration:	**'This research dissertation is submitted for the MSc in Bioinformatics at Queen Mary, University of London'**
Data sharing (delete appropriately)	I **do ~~/ don't~~** allow consent for my project to be shared with future cohorts of students on MSc programmes in the School of Biological and Chemical Sciences and on institutional repositories and websites.
Project Supervisor:	Dr Bob Janes & Elliot Drew

To be completed by Supervisor:

Second Marker Name	
Turnitin score if higher than 17%	

Section A. Continuous Assessment Mark (For supervisor only to assess the student's performance during the practical stages of the project. Note this scheme is inclusive of project type, for example, including field-based, laboratory, modelling and meta-analytic studies).

Mark (%)	Criteria
95 – 100%	Outstanding technical capacity, went beyond expectation in developing protocols, or analytical tools. The engagement was outstanding contributing to lab meetings, and the life of the hosting research group.
85 – 94 %	Exceptional performance showing outstanding technical ability, originality and initiative, high levels of commitment and application, ability to plan and organise the research programme and to contribute substantially to the development of the work.
70 – 84 %	Excellent performance, showing clear evidence of originality, initiative and ability to contribute to the development of the programme.
60 – 69 %	Good performance, notable for steady commitment, sound technical ability and some evidence of initiative and originality. Some contribution to the development of the project but mainly following the advisor's suggestions.
50 – 59 %	Performance generally satisfactory but with some deficiencies in technical ability and/or limited levels of commitment and application to the project. Little or no contribution to the development of the work and very little if any initiative and originality.
40 – 49 %	Performance somewhat weak, characterised by poor technical ability and low levels of commitment and application. Poor understanding of the project and effectively no contribution to the planning and organisation of the work.
20 – 39 %	Unsatisfactory performance. Very poor technical ability amounting to inability to perform routine tasks reliably. Very low levels of commitment and application including unacceptably low attendance.
0 – 19 %	Very poor performance! Careless and totally disorganised, and non-existent technical skills. Unacceptably low attendance.

Section A. Supervisor's Mark Only =_____%

Section B. Assessment of dissertation report (Supervisor and second marker)

Mark (%)	Criteria
95 - 100%	Outstanding performance in producing a document which could be submitted as it for publication from a technical, analytical and editorial perspective.
85 – 94%	Exceptional project report showing very broad understanding of the project area and outstanding knowledge of the relevant literature. Exceptional presentation and analysis of results, logical organisation and ability to evaluate critically and discuss results with insight and originality.
70 – 84 %	An excellent project report showing evidence of wide reading, with clear presentation and thorough analysis of results and an ability to evaluate critically and discuss research findings. Clear indication of insight, understanding and originality. An extremely competent and well-presented report overall, excellent in most aspects.
60 – 69 %	A good project report which shows a clear understanding of the problem and sound knowledge of the relevant literature. Sound presentation and analysis but perhaps not exploiting the results to the full. Relevant interpretation and critical evaluation of results, though with some limitations regarding the scope. Good general standard of presentation and organisation.
50 – 59 %	A satisfactory project report which shows some understanding of the problem but limited knowledge and appreciation of the relevant literature. Presentation, analysis and interpretation of the results at a basic level and showing little originality or critical evaluation. Some weaknesses in the organisation and presentation of the report.
40 – 49 %	A weak project report showing only limited understanding of the problem and superficial knowledge of the relevant literature. Results presented in a somewhat confused or inappropriate manner and incomplete or erroneous analysis in places. Discussion and interpretation of results severely limited, including some basic misunderstandings, and with very little originality or critical evaluation. General standard of presentation weak.
20 – 39 %	An unsatisfactory project report containing substantial errors and omissions. Very limited understanding, or in some cases misunderstanding, of the problem and very restricted and superficial appreciation of the relevant literature. Very poor, confused and, in some cases, incomplete presentation of the results and limited analysis of the results including some serious errors. Severely limited discussion and interpretation of the results revealing little or no ability to relate experimental results to the existing literature. Very poor overall standard of presentation.
0 – 19 %	A very bad project report containing many errors and faults. Virtually no real understanding of the problem and of the literature pertaining to it. Haphazard presentation of results, and in some cases incompletely presented and virtually non-existent or inappropriate or

Mark (%)	Criteria
	plainly wrong analysis. Discussion and interpretation seriously confused or wholly erroneous revealing basic misconceptions.

Section B. Supervisor's Mark =_____%

Section B. Second Marker =_____%

Agreed mark Section B (dissertation) =_____%

If the two marks for section B. (the dissertation) differ by more than 10% a third marker will be needed to adjudicate and for a dissertation mark to be agreed between all three.
(if necessary third markers name: _____)

Overall mark:
= (0.75 x Agreed Mark Section B) + (0.25 x Supervisor's Mark Section A)
= _____

NB Masters students' overall dissertation marks relate to the University descriptive categories as follows:

Mark (%)	**Grade**
70.0% or above	Distinction
60.0 - 69.9%	Merit
50.0 - 59.9%	Pass
Below 49.9%	Fail

Please note the dissertations are second (blind) marked. Markers may mark the work in sequence, one after the other, or in parallel. If they differ in their mark for section B by more than 10%, then a third marker will be needed.

Supervisor's Comments for return to student (please use additional sheets should you wish):

Second Marker's Comments for return to student (please use additional sheets should you wish):

A Novel Fast Single Sequence Secondary Structure Prediction Algorithm Using Word Embeddings and Deep Neural Networks.

Hajar SAIHI | 180279863

Supervised by Dr Bob Janes & Elliot Drew

Abstract

Predicting protein secondary structure given a primary sequence is a major challenge in the field of structural Bioinformatics. Recently, prediction algorithms have taken advantage of the rise in Machine Learning tools namely Artifical Neural Networks to build advanced deep learning models with the ability to accurately predict secondary structures. Accuracy scores have steadily increased over the years and published models are now closing in on the proposed theoretical accuracy scores. Whilst the accuracy scores are high, the time taken to run such predictions are compromised; many algorithms run sequence alignments on the input sequence to obtain a sequence profile as input to the neural network. This process greatly increases the time taken to predict even a single input sequence rendering it computationally inefficient. Here, a novel word embedding approach to represent input sequences was adopted to quickly predict protein structures whilst also retaining relatively high accuracy scores. The outcome from this research was two deep learning models that accurately predict both three and eight states of secondary structures faster than any of the currently published predictors.

Number of words in text: **6373** (pages 1-22 excluding figures, tables and legends)

Acknowledgements

Thank you to both Dr Bob Janes and Elliot Drew for the support and guidance throughout the project. A special thank you to Elliot Drew for support in terms of learning new Machine Learning methods and for lending the book ‘Deep Learning for Python’ by Chollet, F (2018). Thank you to Dr Fabrizio Smeraldi for teaching programming in Python during the course and Professor Conrad Bessant for organising the whole MSc Bioinformatics course.

Contents:

List of all Figures

Sp = Supplementary

List of all Tables

Sp = Supplementary

1. Introduction

Protein secondary structure refers to the three-dimensional fold of local segments in proteins. The secondary structure is determined by the primary amino acid sequence and is held together by hydrogen bonds; in particular, the hydrogen bond between the carbonyl oxygen atom and the amide hydrogen atom. There are two main broad types of secondary structures, α-helices, and β- sheets. The intricate assembly of hydrogen bonds and the careful positioning of amino acid residues to form the secondary structure is vital to dictating the tertiary and quaternary structures, and ultimately the functioning of the protein.

Algorithms exist to predict the secondary structure of each amino acid in a polypeptide sequence. Secondary structure can be classified into one of three-states, α-helix (H), β-strand (E) or other (C), or eight states as defined by DSSP (defined secondary structure of proteins) (Kabsch and Sander, 1983), the three-states and further states; 3-10 helix (G), π helix (I), hydrogen-bonded turn (T), isolated β-bridge residue (B) and bend (S).

1.1 | Why predict secondary structure?

An unprecedented rise in the number of depositions to protein databases such as UniProtKB has widened the gap between the number of protein sequences and resolved protein structures (Mukherjee et al., 2010) found in the Protein Data Bank (PDB) (Berman, 2000). The gap is fuelled by the limitation of experimental techniques and has led to a surge in the need for structure prediction algorithms to keep up with the pace. Besides, the need to predict structural information is critical in fields such as medicine and biotechnology where insight into the secondary structure is important for drug design and enzyme synthesis.

Secondary structure prediction remains a significant challenge in Bioinformatics - developing a ‘successful’ predictor is largely determined by the balance between two factors; the ability to accurately classify each amino acid residue to its correct secondary structure, and to achieve this quickly without compromising runtime. The majority of the current secondary structure predictors fall under the first of these factors, where accuracies are reaching the theoretical limit of 88-90% (Rost, 2001) accuracy for three state prediction, but this means that such predictions can take an inordinate length of time to obtain a result.

1.2 | Generational shifts in secondary structure prediction

First-generation secondary structure prediction algorithms, such as that of Chou and Fasman (Chou and Fasman, 1974) were based on the statistical analysis of how likely an amino acid is to be in a specific secondary structure state. While these methods contributed greatly to the field, they are now obsolete in practise.

Second-generation methods took advantage of the rise in machine learning techniques and were based on the neighbouring effects of amino acids. For example, the Garnier, Osguthorpe, and Robson (GAR) method was based on the information obtained from pairs of residues (Gibrat, Garnier and Robson, 1987). Use of neural networks by Qian and Sejnowski (1988) generated promising results for secondary structure prediction and paved the way for future predictors.

Currently, the third-generation approach of secondary structure predictors dominate the field and use more advanced, deeper learning algorithms. Examples of these methods include Spot-1D, Spider3, NetSurfP-2.0, JPRED, PSIPRED and Porter 5.0. Such predictors exploit evolutionary information in the form of Position Specific Scoring Matrices (PSSMs) as input data to the neural network. The highest accuracy is achieved by Spot-1D; here a three-state prediction has an accuracy of 87% and an eight-state prediction has an accuracy of 77% (Hanson et al., 2018).

1.3 | Deep learning in context

The current degree of accuracy in secondary structure prediction has been made possible because of deep learning technologies, namely artificial neural networks. Such neural networks are made up of three layers; the input layer takes in a representation of amino acids, this is fed into the hidden layer which extracts features and patterns, and infers rules from the data into a format recognisable by the output layer which provides the calculated secondary structure predictions per amino acid (Marr, 2018).

Current prediction algorithms adopt one or an ensemble of two main artificial neural networks – Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs). RNNs build a model by processing information at each timestep and continuously updating the model's state relative to new information being fed in (Chollet, 2018). When training a RNN, the weights (strength of connections) are updated with respect to the loss function (deviation between predicted and actual result); a difficulty can arise where the weights are vanishingly small, preventing a change in its value –

called the 'vanishing gradient problem'. A type of RNN cell called a long short-term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997) is useful because it can overcome this problem by building a 'memory' of long-term sequence interdependencies to yield an output. Further advances in RNNs led to what is known as a Bi-directional recurrent neural network (BRNN) that act by feeding in a sequence in both forward and reverse order; this type of deep learning allows the neural network to retain information based on two states – the forward sequence state and the backward sequence state (Schuster and Paliwal, 1997).

In brief, CNNs are feedforward neural networks that act as feature selectors. Specifically, 1D convolutional layers use 'windows' to iterate over a sequence and extract local patches of patterns and information that can later be recognised elsewhere (Chollet, 2018). Both neural network approaches were adopted in this research.

1.4 | Current predictor limitations

As mentioned before, 'successful' predictors can be broadly identified by two criteria; their ability to accurately classify amino acid residues to their correct secondary structure, and also their ability to do this quickly and efficiently. Published predictors are focused on the first of these criteria because of the use of PSSMs coupled to deep learning tools that lead to evidently high accuracies.

However, PSSMs, generated by carrying out PSI-BLAST runs on input sequences which are then aligned enabling a frequency for each amino acid to be calculated, are tedious and delay prediction; hence the second criteria for a successful predictor is invariably not always met. Furthermore, more sequences will take extra time to process; evolutionary information may not always be available for novel sequences and classification using PSI-BLAST may give an average prediction based on the family of structures rather than the target sequence (Jones, n.d.). Since >90% of known sequences don't share homologous regions (Ovchinnikov et al., 2017a) developing methods based solely on sequence information is pivotal since this could greatly increase the speed of prediction without too much loss of accuracy.

Little development has been made into using only protein sequences as input to the neural network; Spider3-Single is currently one of the few predictors to use sequence information as input to the neural network and make predictions quickly. Yet, Spider3-Single represents each sequence as a one hot encoded vector which is a sparse

representation and computationally inefficient. For example a sequence of 50 amino acids will be represented by a one hot vector of size 20 (*number of amino acids*) x 50 (*sequence length*) (Heffernan et al., 2018). As the input sequence length increases so does the length of the one hot vector and so increasing the sparsity of the input.

To overcome these limitations word embeddings were adopted as an attempt to speed prediction and redefine the shape of sequences as they enter the neural network.

1.5 | Word Embeddings

Word embeddings are a dense and highly-dimensional representation of a word in space learnt from data; interchangeable words are located together in an n-dimensional space and are represented by closely related vectors in comparison to words that are unrelated (*see fig.1a*). Hence the geometric relationships between the words represent the semantic relationships (Chollet, 2018). Word2Vec is a common word embedding model used in natural language processing tasks (Mikolov et al., 2013) to represent words in sentences. Recently, a novel word embedding model for biological sequences was published by Asgari and Mofrad (2015) called BioVec. A sub-class of BioVec includes ProtVec that is used to represent protein sequences. ProtVec word embeddings are a distributed representation where the effect of the neighbouring amino acid residues (± 2 residues) imposes attributes for the central amino acid (Asgari and Mofrad, 2015). Each word is represented by a 100-dimensional vector of its position in a protein space; a 'word' in this context is defined as three amino acid residues, in total there are 8000 words (20x20x20)

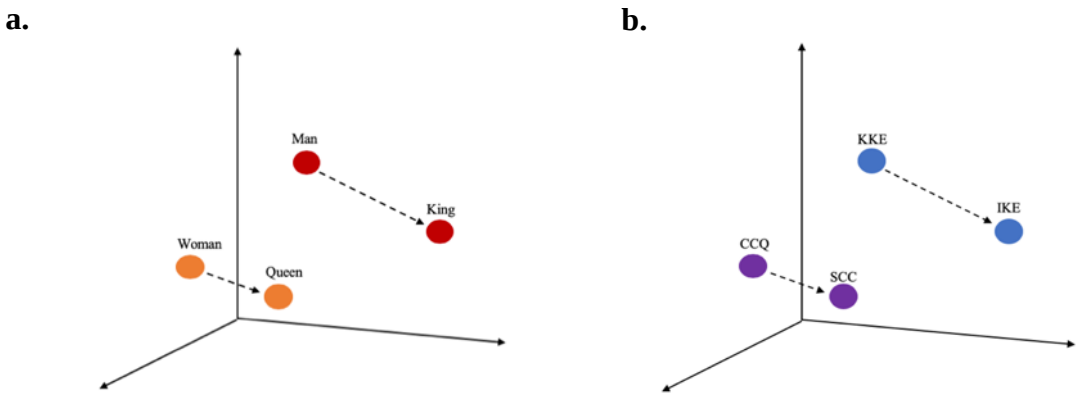


Fig.1: (a) A simplified representation of Word2Vec word-embedding space. Circles represent words, arrows represent the distance between closely related words. Man and Woman are located further apart compared to their neighbouring words (King and Queen respectively). **(b) A simplified representation of ProtVec word-embedding space.** Circles represent words – amino acid triplets, arrows represent the distance between closely related words. KKE and CCQ are located further away compared to their neighbouring words (IKE and SCC respectively).

due to amino acid combinations. Words are located together based on biophysical and biochemical properties and share a closer vector representation compared to words that share no features (*see fig.1b*).

Here, ProtVec word embeddings are used represent protein sequences as input to the neural network as shown in *fig.2*. This approach was expected to act faster by removing the need for sequence alignment and PSSM generation whilst retaining high level accuracy scores. The primary aim of this research was to generate a fast secondary structure prediction tool that represents amino acid sequences using word embeddings. This was achieved by building a deep neural network prediction algorithm to yield an accuracy that matches or competes with current state-of-the-art tools.

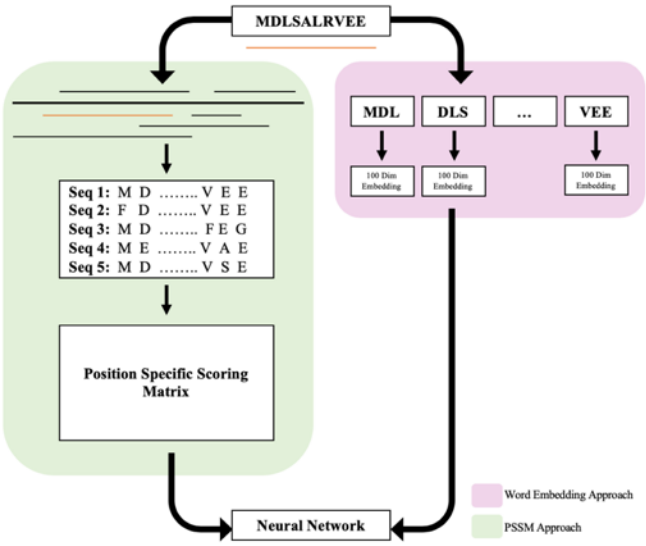


Fig.2: Two amino acid sequence input approaches. Shown in green is the PSSM approach to represent a single sequence, it uses PSI-BLAST to generate a PSSM for input sequences that is fed into the neural network. Shown in purple is the proposed novel word embedding approach that splits the input sequence into 'words' – each word defined as three amino acids. A 100-Dimensional embedding vector per word is extracted and fed into the neural network.

2. Materials and Methods

All work carried out using python 3.7.2 (available at https://www.python.org/downloads/) unless mentioned otherwise. Packages and softwares including versions are outlined in *supplementary table 3*.

2.1 | Data extraction

2.1.1 ProteinNet.

ProteinNet is a recent standardised data set that is publicly available for use (AlQuraishi, 2019). It comprises protein sequences and assigned secondary structures for CASP8-11. Here, the most recent dataset available at the time was used (CASP11). PDB IDs and chain information were extracted across training and validation protein sets from text files (TF Records) found on https://github.com/aqlaboratory/proteinnet. Data splitting was already carefully conducted by ProteinNet. The 30% thinning comprised the training set with a sequence identity threshold of 30%. No secondary structures were available at the time of download so were calculated in section *2.1.3*. The test set was obtained from a publicly available CASP11 target classification publication (Kinch et al., 2016).

2.1.2 PDB file download.

PDB files were downloaded in '.ent' format using BioPython (Cock et al., 2009), in particular Bio.PDB (Hamelryck and Manderick, 2003). mmCIF files with more than 62 chains (Wwpdb.org, 2012) were not used. 17,470 proteins made up the training set, 224 proteins made up the validation set, and 126 proteins made up the test set.

2.1.3 Secondary structure calculation.

DSSP (Define Secondary Structure of Proteins) secondary structures were calculated from each PDB file using the DSSP class in Bio.PDB (Hamelryck and Manderick, 2003). Data was extracted as tuples and files were saved in '.dssp' file format. Each line per DSSP file contained the residue position, chain, index, amino acid residue, secondary structure, phi (φ) angle, psi (ψ) angle, and further bond information. Each amino acid residue per DSSP file was assigned one secondary structure classification out of eight states (B, E, G, H, I, S, T, and C). Any unidentified amino acids labelled 'X' were removed. For three state classification, the eight states were converted into three states as outlined by (Cuff and Barton, 1999); B, G, I, S, T, C were converted into 'Other', whilst H and E remained as α-helix and β-strand respectively.

2.2 | Input features

A ProtVec embedding (Asgari and Mofrad, 2015) dictionary containing words as 'keys' and respective 100-dimensional embedding vectors as 'values' was created. In stage one and two of model training 1195 protein samples made up the training data. RNN and CNN neural network types were trained in stage one and two respectively, in both cases each amino acid was represented using the a 100-dimensional embedding i.e. 100 features.

Stage three and four involved the addition of secondary structure propensities as input features. For all proteins in the training data (17,470), iterative methods were used to extract overlapping words (first and last words were not represented) from protein sequences and corresponding DSSP secondary structures. The number of times a word appeared in a particular secondary structure type (three or eight states) was counted.

Given the frequency of words per structure, the secondary structure propensity per word was calculated giving a total of three or eight additional features. Therefore, each single word was now represented as a 100-dimensional vector from the ProtVec embedding as per before and an additional three or eight secondary structure propensities as features giving 103 or 108 dimensions in total per word. Two embedding dictionaries represented the 103 and 108 features per word and were used in stages three and four. Stage five and six used the best performing neural networks from the previous stages but the sample size increased to 10,000 and 17,000 respectively. For each stage two models were created; a three and eight state secondary structure classifier. A summary of each stage and parameters is shown in *table 1*.

Stage	Neural Network	No. of Samples	Input Dimensions
1	RNN	1195	100
2	CNN	1195	100
3	RNN	1195	103 and 108
4	CNN	1195	103 and 108
5	CNN	10,000	103 and 108
6	CNN	17,000	103 and 108

Table 1: Parameters used for each stage of model training. Two models were created at each stage, a three-state classifier and an eight-state classifier.

2.3 | Visualising embeddings

ProtVec word embeddings represent the position of each word in a 100-dimensional space. PCA was conducted as a form of dimensionality reduction from 100-dimensions into two dimensions to easily visualise geometric relationships between words. This was carried out using the PCA tool as part of the SciKit Learn package (Pedregosa et al., 2011).

Each word was assigned three attributes, the first two were the x and y positional components retrieved from the PCA and the third was a 'target' secondary structure that was the structure with the highest propensity value (*as described in 2.2*). Points were projected onto a two-dimensional space and data binning was carried out to group the continuous data into 'bins'. The number of points per 'bin' was counted and represented by the Matplotlib (Hunter, 2007) contour map feature.

Four contour plots were generated in total; the first two represented clustering of three secondary structure types (E, H, C), whilst the second two represented clustering of eight secondary structure types (B, E, G, H, I, S, T, and C), both include plots before and after the addition of secondary structure propensities as features.

2.4 | Neural Network Architecture

Neural networks were created using TensorFlow (Abadi et al., 2016) and Keras (Chollet, 2015) machine learning libraries. Three sets of data were exposed to the neural network; the 'training' set comprised amino acid representations and respective secondary structure labels from known sequences, the 'validation' set that was used to alter and finetune the model hyperparameters, and finally, the 'test' set that was used to evaluate the final model performance and provide an unbiased evaluation of the model.

2.4.1 Recurrent neural network.

The input shape of the training data had to be manipulated to enter the RNN in the standard machine learning form of: (samples, timesteps, features). In this case, the 'samples' were the number of proteins used in training, a small subset of 1195 proteins were initially used as outlined in *2.2.* The 'timesteps' was the length of amino acids in a sequence. A box plot was generated using python to visualise the distribution of sequence lengths – 1000 words (a length of 1002 amino acids) was used as the number of timesteps. Sequences in which the number of words was longer than the number of timesteps were truncated and shorter sequences were padded with empty vectors. Empty vectors were

masked using the Keras Masking layer (Chollet, 2015) with a 'mask_value' of 0, and gradients were rescaled to a clipnorm value of 1.0 and a clipping value of 0.5 to prevent exploding gradients (caused by really high values for the weights of connections). The number of features used was dependant on the stage of training (*see table 1*).

In brief, the hidden layers of the RNN comprised of stacked bidirectional-LSTM (BLSTM) layers followed by dense layers as shown in *fig.3*. A standard Adam 'categorical cross-entropy' loss optimiser with a learning rate of 0.001 and decay of $1e^{-4}$ was used. ReLU (*rectified linear unit*) activation was used across layers aside from the output layer which used a Softmax activation.

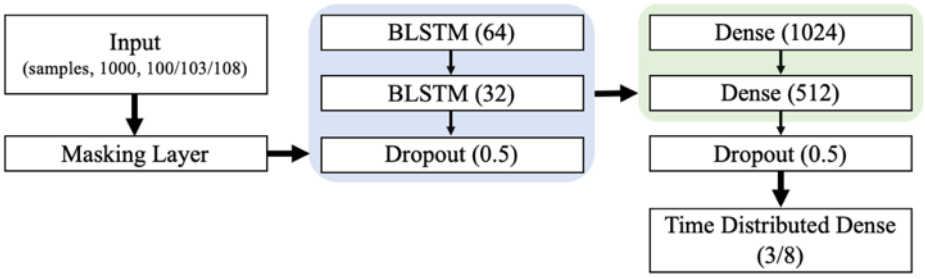


Fig.3: RNN architecture. Each layer of the RNN is shown. The hyperparameters used per layer is shown in brackets.

Secondary structure labels for each amino acid in the sequence were represented as one-hot encoded vectors. The output layer was a time distributed dense layer with three nodes for either a three-state classifier or eight nodes for an eight-state classifier. The output for each amino acid that entered the network was either three or eight probability scores summing to one, each probability score represented the confidence for each secondary structure type.

2.4.2 Convolutional neural network.

The input data for the CNN also had to be manipulated to fit the standard 1D Convolution input shape that is: (batch, steps, channels) for the neural network to recognise the input. A batch represents the total number of amino acids, steps represents the size of the sequence windows and the number of channels refer to the features used to represent each window. The first step was to pad input sequences such that the first and last amino acids

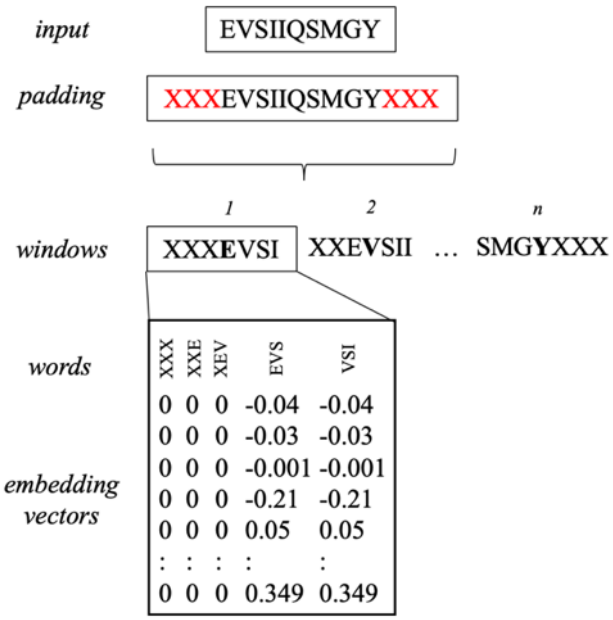


Fig.4: Simplified CNN Input sequence representation. An input sequence is padded with red X's representing the padding value at the start and end of each sequence. Window sizes of 7 (*for illustration only*) are extracted from the input sequence, and overlapping words are extracted from each window. n-dimensional embedding vectors are extracted for each word (*100 dimensions for round one and 103/108 for round two*), this process is continued for each window in the sequence.

are included. The number of padding values ('X') added to the start and end of sequences was (n-1)/2 (*where n represents the window size*), this is shown in *fig.4*. A range of windows (11-21 by increments of 2) were tested to identify the optimal window size. A total of 17 amino acids was selected as the window size, no significant improvements were seen in larger windows. From each window, iterative methods were used to extract overlapping words – any words containing the padding value were represented as empty padding vectors (containing the value '0'), this is indicative of the start and end of sequences. The number of channels or features was dependant on the training stage (*see table 1*).

In brief, the CNN comprised 1D convolutions, batch normalisation and dense layers as shown in *fig.5*. A SGD (*stochastic gradient descent*) 'categorical cross entropy' loss optimiser with a learning rate of 0.01, decay of $1e^{-6}$, and a momentum value of 0.8 was used. In addition, to reduce overfitting a kernel regulariser with an l2 value of 0.01 was added.

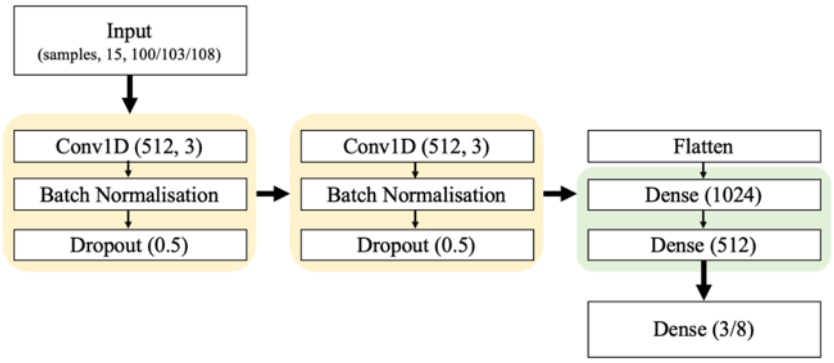


Fig.5: CNN architecture. Each layer of the CNN is shown. The hyperparameters used per layer is shown in brackets.

Activation methods and outputs for the CNN model was identical to the RNN as outlined in *2.4.1*. In this case, secondary structures labels represented the central amino acid in the window. The number of epochs and batch size used for each stage and neural network type is outlined in *supplementary table 1*.

2.5 | Performance Evaluation

Validation loss and accuracy were monitored and recorded to prevent overfitting and underfitting. A final model accuracy was obtained by predicting secondary structures for the CASP11 test data set. A confusion matrix was generated using the SciKit Learn Metrics package to evaluate the quality of the predicted outputs against true outputs for the test data across the final three and eight state models. All output values were normalised to give the percent accuracy.

2.6 | Benchmarking Accuracy Scores

CB513 dataset is a widely used benchmarking test dataset to evaluate secondary structure prediction models. The CB513 set (obtained from http://www.compbio.dundee.ac.uk/jpred/about.shtml) comprises 513 non-redundant protein with a sequence similarity of <25%. Amino acids 'B' and 'Z' (asx and glx) were not present in the ProtVec dictionary, together these accounted for <1% of the total number of amino acids and hence were randomly converted into N/D and Q/E respectively. The Q3 and Q8 scores were calculated using Conv103 and Conv108 models respectively. Undetermined amino acids ('X') that accounted for <1% of total amino acids were also removed.

2.7 | Speed Evaluation

To assess the prediction speed, three proteins from the CASP11 dataset were selected to replicate a typical user experience. 2MCTA was selected as an uncharacterised protein with no PDB hits and 1YN4A was selected as a protein with known PDB alignments and thus expected to run faster. These had a similar length (102 and 99 amino acids respectively). 1QA2A was selected as a larger protein with a sequence length of 554 amino acids.

The time taken to predict three-state secondary structures for each protein was recorded using several published online servers (JPRED, PSIPRED, NetSurfP, Porter5.0) using the workbench outlined in *2.8*. The time taken was compared to the locally run final models in this research. The online servers for Spider3 and Spider3-Single were unavailable at the time of testing thus a locally run version of Spider3-Single was used.

To account for bias in the time taken to submit a sequence to the online servers, two speed tests were conducted, one during the morning and one during the evening were less traffic was expected, the average time between the two was calculated. Where known, no other sequences/jobs were present in the queue. Not all servers offered an eight-state prediction, so this was not carried out.

2.8 | Computational Resources

Stages one to four were trained using a local machine with 8GB ram and an intel core 5 2.5GhZ processor. Stages five and six were trained using a high-performance computing cluster (King, Butcher and Zalewski, 2017); a total of 32GB*4 CPU cores were requested for jobs of 10,000 and 17,000 proteins.

3. Results

3.1 | Contour plots show clustering after feature addition

PCA meant words were projected onto a two-dimensional x,y plane (PC1 and PC2 respectively). Through a process of binning contour plots were generated to visualise the protein embedding space. *Fig.6a* reveals a single dense cluster (between PC1 values of 2-4 and PC2 values 4-8) between all three secondary structure types when representing words using 100-dimensions.

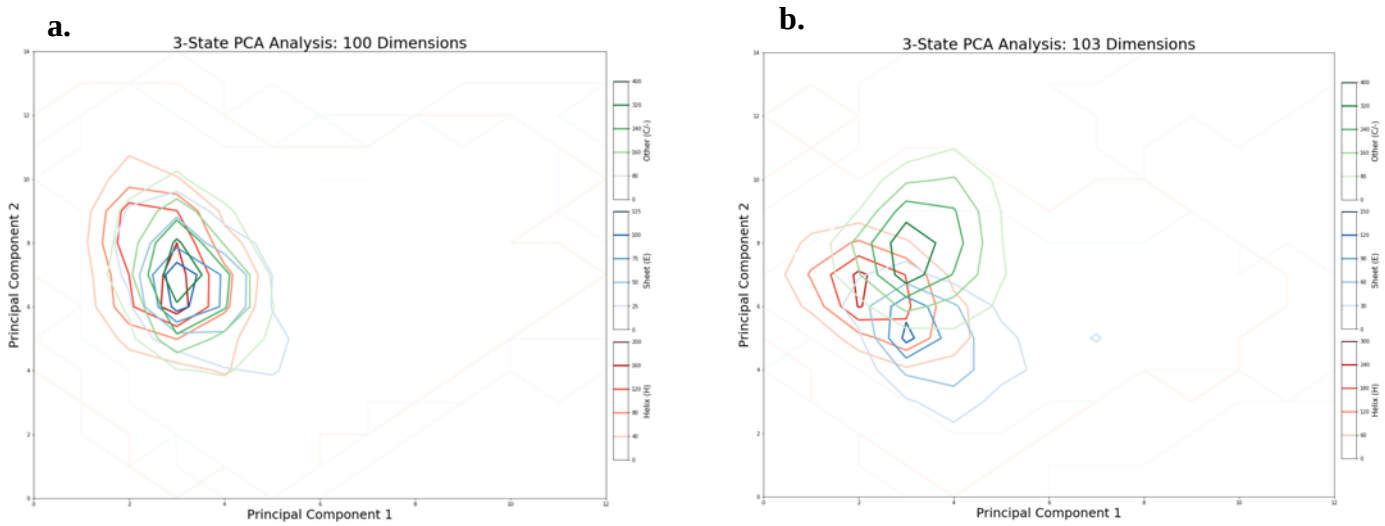


Fig.6: Contour plots representing word clusters for each 3-state secondary structure type. Contour lines represent the distribution of data points, red represents 'Helix' structures, blue represents 'Other' and green represents 'Sheet'. Darker lines represent a greater density of words. **(a.) PCA using a 100-dimensional embedding**. **(b.) PCA using a 103-dimensional embedding**.

After the addition of secondary structure propensities to each word to give 103 dimensions, clear clustering and group separation between denser regions of words attributed to E, H, and C structures is shown in *fig.6b*.

The second set of contour plots represent eight-state secondary structure targets. *Fig.7a* also represents a single dense cluster of words (between PC1 values of 3-5 and PC2 values 5-8) for each eight-secondary structure group when using a 100-dimension representation. Propensities calculated for B, G, H, I and S structure types were not high enough such that these structures were assigned targets for words and hence no plots are shown.

After the addition of eight-state secondary structure propensities to yield 108 dimensions, *fig.7b* shows the clustering and clear separation of words attributed to E and H structures. An overlap is still seen in groups that have C, T, and S as their target secondary structure. Propensities calculated for B, G and I structure types were not high enough such that these structures were assigned targets for words and hence no contour lines are shown.

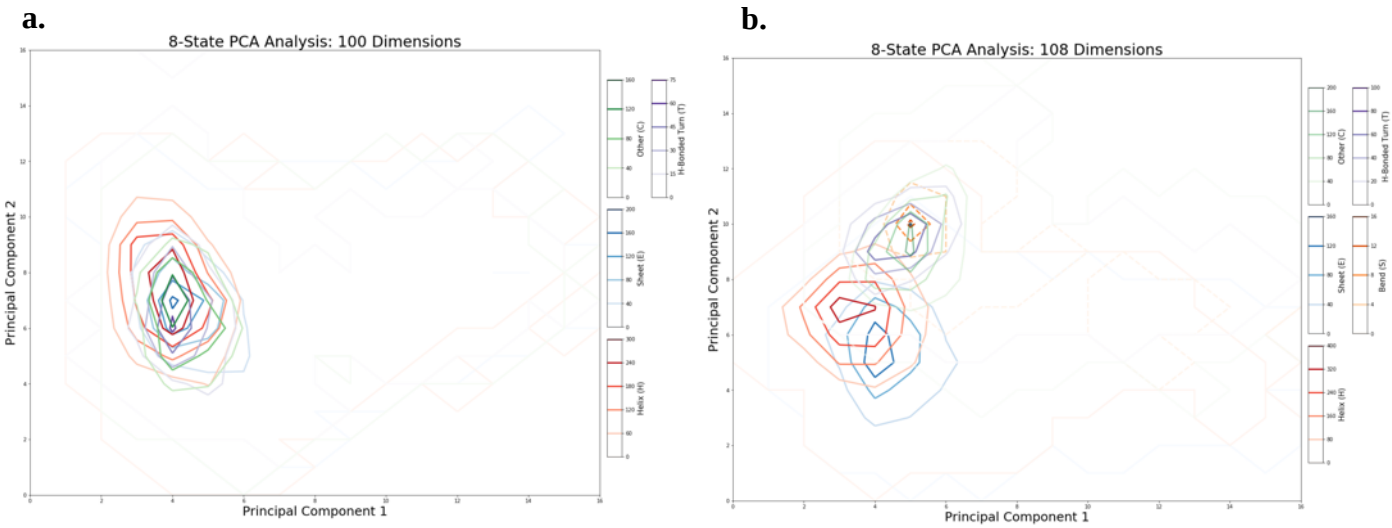


Fig.7: Contour plots representing word clusters for each 8-state secondary structure type. Contour lines represent the distribution of data points, red represents 'Helix' structures, purple represents 'H-Bonded Turn', blue represents 'Other', orange represents 'Bend' and green represents 'Sheet'. Darker lines represent a greater density of words. **(a.) PCA using a 100-dimensional embedding**. B, G, S, and I structures are not shown. **(b.) PCA using a 108-dimensional embedding**. B, G and I structures are not shown.

3.2 | Reporting Q3 and Q8 accuracy scores

Six stages of training took place to increase model accuracy over time (see *fig.8*), for each stage two models were created; a three and eight state secondary structure classifier. The accuracy score (Q3 and Q8) for the final models at each stage is reported in *fig.8*. It is important to highlight that the test set was only used once at the end of training for each stage to obtain a final accuracy score; the test set did not influence the tuning of model hyperparameters during development.

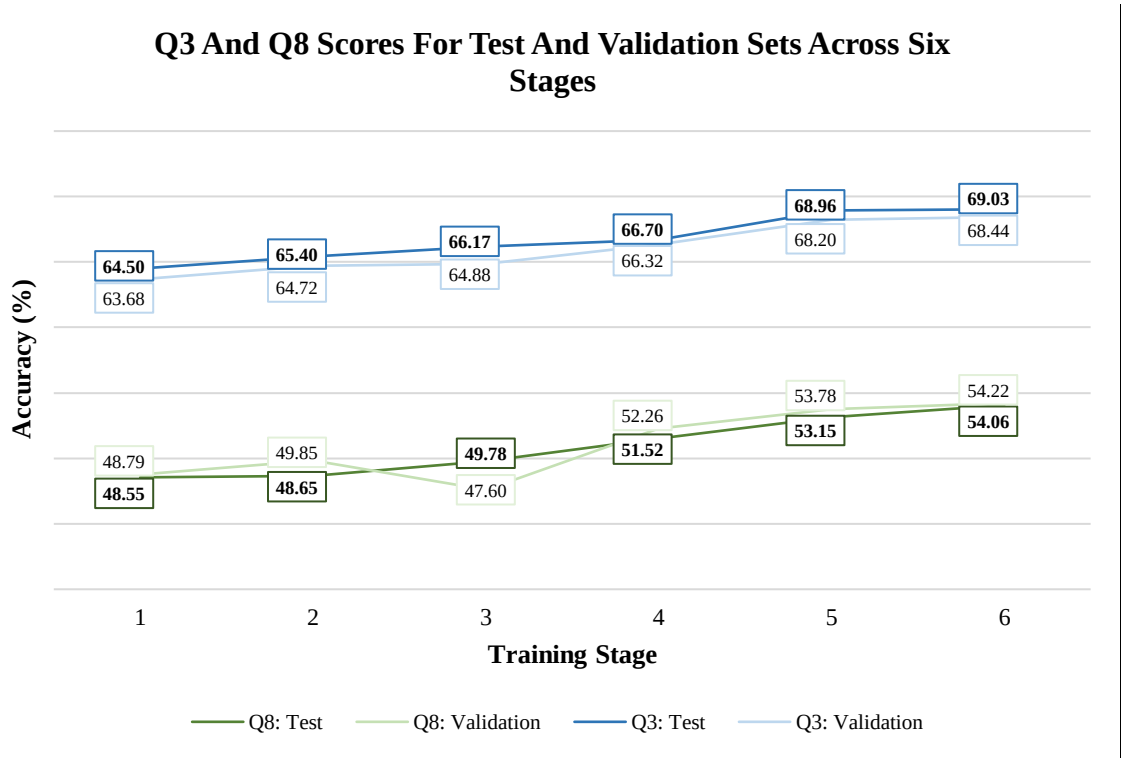


Fig.8: Changes in Q3 and Q8 test and validation accuracy over time. A line graph represents the changes in accuracy based on the validation and test datasets across 6 stages of training. A final Q3 accuracy of 68.4% and 69.03% for validation and test datasets respectively is achieved. A final Q8 accuracy of 54.20% and 54.06% for validation and test datasets respectively is achieved.

Stage one and two of training utilised ProtVec 100-dimensional vectors to represent each word in a sequence for training and validation data, these stages had the lowest accuracy scores across test data. In stages three and four words were represented by either a 103-dimensional embedding or 108-dimensional embedding. *Fig.8* shows that the addition of extra features leads to a 1.67% and 1.3% rise in Q3 accuracy, and a 1.23% and 2.87% rise in Q8 accuracy using RNN and CNN models on the test set respectively. Across all stages the difference between validation and test accuracy is small (<2.5%) indicating optimal models with no overfitting on the validation set.

High Q3 and Q8 scores seen in CNN models stimulated an increase in the number of proteins to 10,000 in stage 5 and 17,000 in stage 6. A final Q3 accuracy of 68.4% in the validation data set and 69.03% in the test dataset and a final Q8 accuracy of 54.2% in the validation data set and 54.06% in the test dataset are reported. The final three and eight-state models are referred to as Conv103 and Conv108 respectively from hereon; model parameters and reported accuracy and loss values for all models are shown in *supplementary table 1*.

3.3 | Variation in predictive power for different secondary structures.

To assess the model performance and bias on a per secondary structure basis a confusion matrix was generated. Confusion matrices were based on predictions from the test dataset (CASP11) and are visualised in *fig.9*. Results from the Conv103 model (*fig.9a*) demonstrate high Q3 accuracies for 'H-H' and 'Other-Other' prediction (0.76 respectively) and a lower accuracy for 'E-E' prediction (0.46). In contrast, prediction accuracy is greatly reduced (<=0.45) across all predicted-true secondary structure groups aside from 'H-H', 'H-I', and 'E-E' using the Conv108 model. Higher 'H-H' and 'E-E' accuracies are seen in the Conv108 model shown in *fig.9b* (0.81 and 0.69 respectively) compared to that achieved by Conv103.

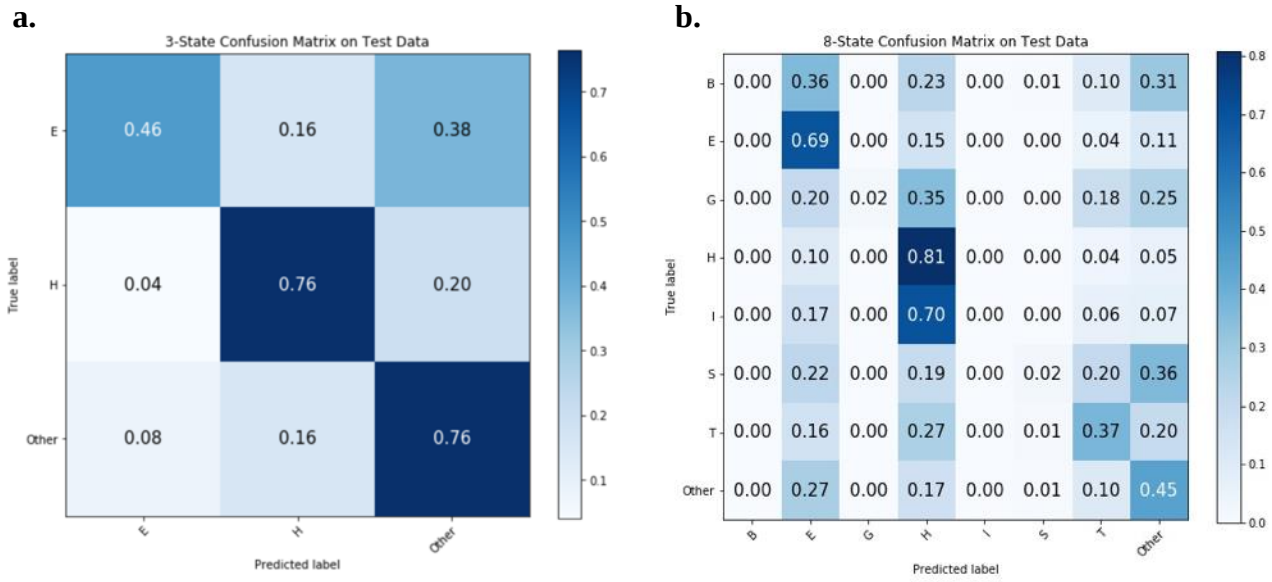


Fig.9: Normalised confusion matrices. Shown is the true and predicted labels using the Conv103 and Conv108 models**.** Darker colours represent higher accuracy scores**(a) Three-state secondary structure prediction confusion matrix using Conv103**. **(b) Eight-state prediction confusion matrix using Conv108**.

3.4 | Published CB513 scores compared to Conv103 and Conv108

CB513 is a benchmarking test set that was used to assess model performance across known published predictors including; NetSurfP, JPRED, PSIPRED, MUFOLD-SS, and Spider3. *Table 2* shows the accuracy scores reported for these predictors adapted from Klausen et al. (2019) and Zhang, Li and Lü (2018) compared to the Conv103 and Conv108 models in this research. A final Q3 score of 69.86% and a Q8 score of 53.15% using Conv103 and Conv108 models is reported in *table 2*.

Method	*CB513*	
	Q3(%)	Q8(%)
NetSurfP 2.0 *(mmseqs)*[1]	85.4	72.3
JPRED[2]	81.7	-
PSIPRED[2]	79.2	-
MUFOLD-SS[2]	82.7	-
Spider3[2]	84.58	-
RaptorX[1]	82.7	70.6
This work	**69.86**	**53.15**

Table 2: Q3 and Q8 accuracy scores using the published CB513 test dataset comparing this work with known published results. [1] Accuracy scores adapted from Klausen et al. (2019). [2] Accuracy scores adapted from Zhang, Li and Lü (2018).

3.5 | The fastest prediction algorithm

The prediction time profiles to assess the speed across known predictors is shown in *table 3*. A fast prediction speed was consistently seen when using the Conv103 model to predict the structure of the three proteins compared to published predictors.

The average time taken to predict 1QA2A with a length of 554 amino acids is 326 seconds, compared to 6.4 seconds using the Conv103 generated in this research. Fast prediction times were also seen when predicting shorter 1YN4A and 2MCTA proteins;

Method	1QA2A *(l:554)*	1YN4A *(l:99)*	2MCTA*(l:102)*
	Time (*seconds*)		
NetSurfP 2.0	300	85	94
JPRED4	235.5	107.5	110.5
PSIPRED	371	129	91.5
Spider3-Single	29.5	24.5	22
Porter5.0	694	241	231.5
This work	**6.4**	**5.4**	**5.9**

Table 3: Time profiles in seconds for the prediction of three proteins (1QA2A, 1YN4A, 2MCTA) using different published prediction methods. Values ≥ 100 seconds are shaded orange, values <100 seconds and ≥ 15 seconds are shaded yellow, and those <15 seconds are shaded green.

117.4 seconds compared to 5.4 seconds and 109.9 seconds compared to 5.9 seconds respectively. It was interesting to note that the Conv103 model performed faster than the locally run version of Spider3-Single that uses only sequence information as input to the neural network.

Out of interest, the time taken to predict 513 proteins (CB513 set) using the Conv103 model was also noted. The prediction took 93.88 seconds which was considerably lower than some of the predictors took to predict one protein sequence. A breakdown of the prediction speed at each time interval and the predicted secondary structure outputs for each protein are reported in *supplementary table 2 and supplementary fig.1*.

4. Discussion and Conclusion

4.1 | Prediction time outperforms published predictors.

Secondary structure prediction is not only important in Biochemistry towards understanding the intricate assembly and thus functional roles of proteins, but it has also manifested in many other important fields such as Biotechnology and Medicine. For example, pharmaceutical companies that mass-produce antibodies tend to predict secondary structures as a screening procedure to remove proteins without desired folds or structures (Darnell, 2015). Results in this research demonstrate that even for a single protein, the time taken by currently published predictors to generate predictions is considerably long; indeed, mass prediction of proteins with the current servers could take up to many days. The development of Conv103 and Conv108 – two fast prediction algorithms, contributes significantly to the field by greatly reducing the time taken to predict a single protein. Future developments could lead to the deployment of an online server that predict single sequences and also process batch jobs to output predictions faster than any of the currently published prediction models.

Some may argue that Spider3-Single already exists as a tool to predict secondary structures in a timely manner as it does not generate PSSMs. However, this research has shown that by using word embeddings there is a greater capacity to yield even faster predictions without compromising on accuracy significantly. Albeit the run time for using Spider3-Single to prediction secondary structures of single proteins is fast (it is important to highlight here that the time is based on a locally run version, not many researchers retain knowledge of how to run code locally), the time taken to run more than one protein could increase greatly. Batch jobs were not explored in this research and could

be a source for future work however the time taken to predict 513 (CB513) proteins was considerably lower than the time taken to predict one protein using the online published predictors. This is a promising result and useful for development of batch predictions.
Further, word embeddings are advantageous as they have been trained on thousands of proteins so retain some level of evolutionary information that may not be available when using one-hot encoding. Others may state that although Spider3-Single is slightly slower, the accuracy scores achieved by Spider3-Single for three and eight state prediction (72.5% and 60% respectively) are higher. Whilst this is true, due to time constraints deeper neural network methods were not explored further, hence room to further improve accuracy scores presented in this research is still available.

4.2 | Embeddings and the protein space

Embeddings represent the geometric location of each word in a protein space and are widely used in natural language processing tasks to frequently yield high accuracy scores. The use of word embeddings in this research did not instantly lead to high accuracy scores after the first two stages of training as shown in *fig. 8*. An accuracy of ~64% and ~54% for three and eight-state is recorded which is considerably lower than published accuracy scores for both three and eight state prediction. Published ProtVec findings reveal clear clustering between words and biochemical and biophysical properties such as hydrophobicity, polarity charge, volume and mass yet no secondary structure clustering was shown (Asgari and Mofrad, 2015). It was expected that after word extraction and visualisation using PCA clustering between secondary structure types would be visible yet this was not the case. Across both three and eight-states, the dense regions of words were overlapping so the data was not representative of secondary structures which could explain why the accuracy scores in stages one and two were low.
The delineation of clusters attributed to a particular secondary structure was vital to better represent these structural groups. By adding secondary structure propensities as features, words were separated in the protein space based on what structure they are most likely to be found in. The result was an increase in accuracy by ~1-2% in both the RNN and CNN.

4.2.1 Input shape

The main challenge in this research was to define the shape of the sequence represented by the embeddings. Using simple 100-dimensional protvec word embeddings directly without padding was ineffective. To improve this, an amalgam of approaches were

adopted to represent the word embeddings such as simple windows of word vectors, through to an average sum of each window represented as a single vector. The final approach adopted to represent sequences (*fig. 4*) proved to be the most successful and also represented the start and end of each sequence.

4.3 | Building deeper neural network models

It was unexpected to see the RNN perform less accurately compared to the CNN model even after the addition of secondary structure propensities. Generally when training neural networks that take sequence data as input a RNN is used, in particular, LSTM cells retain the ability to remember sequential information and have 'gates' that decide on what information to 'remember' and what is not useful. In addition to this, it was expected that bidirectional LSTM cells that retain information on not only what came before in the sequence, but also what came after would far outperform a convolutional neural network. The main reason that could explain why the RNN performed worse was the number of timesteps used (1000). Some reports suggest that a reasonable limit of 250-500 time steps yields the best RNN models (Brownlee, 2017). A large number of timesteps leads to exploding gradients – witnessed on a few occasions in this research. Further, truncating sequences that are longer than 1000 amino acids led to the loss of potentially important information that may be valuable to the model to ascertain predictions. Comparitively, sequence lengths were not truncated when using the CNN model hence sequence information was perserved.

Due to computational resources such as being limited to a local pc, building advanced deeper models was challenging and computationally intensive. Higher accuracy from published can be due to advanced deep learning models. For example, Spot1D uses Residual Networks (ResNet) (He et al., 2015) – in a traditional network data is passed from one layer to the next whereas ResNets also pass the data to layers that are layers away through a 'skip connection' (He et al., 2015).

ResNets are especially effective in building deeper networks to overcome the vanishing gradient problem – after several layers, the accuracy is saturated whereas by concatenating the input to the output information is retained (Fung, 2017). Spot1D, in this case, uses 21 convolutional blocks to build a deep learning model and yield an accuracy of 87% and 77% for three and eight state prediction respectively (Hanson et al., 2018).

Another neural network approach includes building a CNN in series with LSTM cells such that the outputs from the convolutional layer are fed directly into the LSTM cells.
Other advanced methods include CuDNN LSTM cells (Chollet, 2015) that run on GPU with TensorFlow backend and outperform LSTM cells in terms of training time. Independently recurrent neural networks (IndRNN) were also recently developed by Li et al. (2018) and results show that they are robust to training deep networks (21 layers) in addition to processing long sequences (over 5000 timesteps) (Li et al., 2018).

4.4 | Achieving theoretical accuracies

Reaching the theoretical accuracy of 88-90% is only true in instances where many homologous sequences are available and used intensively (Hanson et al., 2018). Yet, with a rise in genome sequencing comes a growing percentage of novel proteins that share little homology with known protein structures (Yang et al., 2016). In fact, in prokaryotic genomes, for example, a third of proteins lack any similarity to known proteins (Aydin, Altunbasak and Borodovsky, 2006). In these instances where there are little neighbouring sequences, PSSMs may yield less effective sequence profiles (Yang et al., 2016). PSIPRED, for example, adopts an approach where if there are a few hits, the profile is based on just the query sequence and appropriate elements correspond to the BLOSUM62 matrix (Jones, 1999).
Research by Hanson et al. (2018) showed that a Neff score of 1 (i.e. few homologous sequences) causes a drop in Q3 accuracy from 86% at Neff 10 to 77%; sequences with no homologous sequences could yield an even lower accuracy. The method outlined in this research overcomes this issue by using word embeddings to represent sequences irrespective of sequence homology.
By analysing the percent accuracy per class basis for Conv103 prediction, high accuracy is seen across H and C, a low score is seen for β-strand prediction (46%) which reduces the overall accuracy to 69.03%. β-strands are harder to predict compared to α-helices, this is because the bonding patterns involved in α-helix formation is based on sequence neighbours whereas those present in β- strand may not be located closely together hence a low accuracy is expected (Rost, 2001). Comparatively, analysis of accuracy scores per class basis using Conv108 are higher for β-strand and α-helix, this could be due to the fact that the data per class is further refined so patterns are clearly identified. *Fig. 9b*

shows that B, G, I and S have low or no prediction scores, by removing these four groups and developing a four-state predictor (E, H, T, C/Other) accuracy could greatly improve.

4.5| Managing underfitting, overfitting and bias

A key challenge encountered during model development was to prevent over or underfitting. Overfitting is when the validation loss supersedes the training loss meaning the model performs well on the training data but lacks generalisation ability and so performs badly on the validation data. Overfitting was avoided in three ways; first by careful monitoring of the validation loss to prevent overfitting on the validation set. One way this can happen is through 'validation leakage', constantly using the validation data to finetune the model means that the model becomes better at predicting the validation set and thus overfits. The second method was to expose the test data to the neural network only once at the end of complete training, this way the test data was not involved in the finetuning of the model parameters. Finally, a kernel regulariser was used and worked by introducing penalties at each layer which were then accounted for during loss optimisation (Chollet, 2015). Underfitting is seen when the training loss is considerably lower than the validation loss. Underfitting was avoided by increasing the capacity of the network by adding a large number of nodes per layer to build sufficiently complex models. Both final models had a validation loss value that was either equal to or slightly less than the training loss (*see supplementary table 1*) meaning no overfitting or bias was introduced.

The primary aim of this research was to generate a fast secondary structure prediction tool by building a deep neural network prediction algorithm to yield an accuracy that matches or competes with current state-of-the-art tools. Two models were created; Conv103 classified amino acids into one of three secondary structures E, H, or Other, whilst Conv108 classified amino acids into an additional B, G, I, S, T, and Other. A significant contribution was made to the field of secondary structure prediction by developing models that can outperform currently published predictors and maintain relatively high accuracy scores. This research provides a stepping stone for the development of advanced models that use word embeddings to represent single sequences.

References

1. Abadi, M., Barham, P., Jianmin Chen, Zhifeng Chen, Davis, A., Dean, J., Devin, M., Sanjay Ghemawat, Irving, G., Isard, M., Manjunath Kudlur, Levenberg, J., Rajat Monga, Moore, S., Murray, D.G., Benoit Steiner, Tucker, P., Vijay Vasudevan, Warden, P., Wicke, M., Yu, Y. and Xiaoqiang Zheng (2016). *TensorFlow: A system for large-scale machine learning*. [online] Google AI. Available at: https://ai.google/research/pubs/pub45381 [Accessed 29 Aug. 2019].
2. Anaconda. (2018). *Anaconda*. [online] Available at: https://www.anaconda.com [Accessed 29 Aug. 2019].
3. Asgari, E. and Mofrad, M.R.K. (2015). Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE*, 10(11), p.e0141287.
4. Aydin, Z., Altunbasak, Y. and Borodovsky, M. (2006). Protein secondary structure prediction for a single-sequence using hidden semi-Markov models. *BMC Bioinformatics*, 7(1), p.178.
5. Berman, H.M. (2000). The Protein Data Bank. *Nucleic Acids Research*, 28(1), pp.235–242.
6. Brownlee, J. (2017). *Techniques to Handle Very Long Sequences with LSTMs*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/handle-long-sequences-long-short-term-memory-recurrent-neural-networks/ [Accessed 25 Aug. 2019].
7. Chollet, F. (2015). *keras-team/keras*. [online] GitHub. Available at: https://github.com/keras-team/keras.
8. Chou, P.Y. and Fasman, G.D. (1974). Prediction of protein conformation. *Biochemistry*, 13(2), pp.222–245.
9. Connor Shorten (2019). *Introduction to ResNets*. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4 [Accessed 25 Aug. 2019].
10. Cuff, J.A. and Barton, G.J. (1999). Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins: Structure, Function, and Genetics, 34(4), pp.508–519.
11. Darnell, S. (2015). Why Structure Prediction Matters | DNASTAR. [online] DNASTAR. Available at: https://www.dnastar.com/blog/structural-biology/why-structure-prediction-matters/.
12. Fung, V. (2017). *An Overview of ResNet and its Variants*. [online] Towards Data Science. Available at: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.
13. Gibrat, J.-F., Garnier, J. and Robson, B. (1987). Further developments of protein secondary structure prediction using information theory. Journal of Molecular Biology, 198(3), pp.425–443.
14. Hanson, J., Paliwal, K., Litfin, T., Yang, Y. and Zhou, Y. (2018). Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. *Bioinformatics*, 35(14), pp.2403–2410.
15. He, K., Zhang, X., Ren, S. and Sun, J. (2015). *Deep Residual Learning for Image Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1512.03385.
16. Heffernan, R., Paliwal, K., Lyons, J., Singh, J., Yang, Y. and Zhou, Y. (2018). Single-sequence-based prediction of protein secondary structures and solvent accessibility

by deep whole-sequence learning. *Journal of Computational Chemistry*, 39(26), pp.2210–2216.
17. Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780.
18. Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), pp.90–95.
19. Jones, D. (n.d.). Bioinformatics: Secondary Structure Prediction. [online] Available at: http://bioinf.org.uk/teaching/c40/pdf/ssplusfoldrec.pdf [Accessed 12 Jul. 2019].
20. Jones, D.T. (1999). Protein secondary structure prediction based on position-specific scoring matrices 1 1Edited by G. Von Heijne. *Journal of Molecular Biology*, 292(2), pp.195–202.
21. Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12), pp.2577–2637.
22. King, T., Butcher, S. and Zalewski, L. (2017). Apocrita - High Performance Computing Cluster for Queen Mary University of London. *Zenodo*. [online] Available at: https://zenodo.org/record/438045#.XTwmOC3Mw1I [Accessed 27 Jul. 2019].
23. Klausen, M.S., Jespersen, M.C., Nielsen, H., Jensen, K.K., Jurtz, V.I., Sønderby, C.K., Sommer, M.O.A., Winther, O., Nielsen, M., Petersen, B. and Marcatili, P. (2019). NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(6), pp.520–527.
24. Li, S., Li, W., Cook, C., Zhu, C. and Gao, Y. (2018). *Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN*. [online] arXiv.org. Available at: https://arxiv.org/abs/1803.04831 [Accessed 25 Aug. 2019].
25. Marr, B. (2018). What Are Artificial Neural Networks - A Simple Explanation for Absolutely Anyone. Forbes. [online] 24 Sep. Available at: https://www.forbes.com/sites/bernardmarr/2018/09/24/what-are-artificial-neural-networks-a-simple-explanation-for-absolutely-anyone/ [Accessed 10 Jul. 2019].
26. Mckinney, W. (2010). *Data Structures for Statistical Computing in Python*. [online] *PROC. OF THE 9th PYTHON IN SCIENCE CONF*, p.51. Available at: https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf [Accessed 29 Aug. 2019].
27. Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. [online] Available at: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf [Accessed 13 Jul. 2019].
28. Mukherjee, S., Szilagyi, A., Roy, A. and Zhang, Y. (2010). Genome-Wide Protein Structure Prediction. Multiscale Approaches to Protein Modeling, pp.255–279.
29. Ovchinnikov, S., Park, H., Varghese, N., Huang, P.-S., Pavlopoulos, G.A., Kim, D.E., Kamisetty, H., Kyrpides, N.C. and Baker, D. (2017a). Protein structure determination using metagenome sequence data. *Science*, 355(6322), pp.294–298.
30. Ovchinnikov, S., Park, H., Varghese, N., Huang, P.-S., Pavlopoulos, G.A., Kim, D.E., Kamisetty, H., Kyrpides, N.C. and Baker, D. (2017b). Protein structure determination using metagenome sequence data. *Science*, 355(6322), pp.294–298.
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É. (2011). Scikit-learn:

Machine Learning in Python. *Journal of Machine Learning Research*, [online] 12(Oct), pp.2825–2830. Available at: http://jmlr.org/papers/v12/pedregosa11a.html.
32. Qian, N. and Sejnowski, T.J. (1988). Predicting the secondary structure of globular proteins using neural network models. Journal of Molecular Biology, 202(4), pp.865–884.
33. Rost, B. (2001). Review: Protein Secondary Structure Prediction Continues to Rise. *Journal of Structural Biology*, 134(2–3), pp.204–218.
34. Schuster, M. and Paliwal, K.K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), pp.2673–2681.
35. Sublimetext.com. (n.d.). *Sublime Text - A sophisticated text editor for code, markup and prose*. [online] Available at: https://www.sublimetext.com [Accessed 29 Aug. 2019].
36. van der Walt, S., Colbert, S.C. and Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2), pp.22–30.
37. Wwpdb.org. (2012). *PDBx/mmCIF General FAQ*. [online] Available at: http://mmcif.wwpdb.org/docs/faqs/pdbx-mmcif-faq-general.html [Accessed 21 Aug. 2019].
38. Yang, Y., Gao, J., Wang, J., Heffernan, R., Hanson, J., Paliwal, K. and Zhou, Y. (2016). Sixty-five years of the long march in protein secondary structure prediction: the final stretch? *Briefings in Bioinformatics*, 19(3), p.bbw129.
39. Zhang, B., Li, J. and Lü, Q. (2018). Prediction of 8-state protein secondary structures by a novel deep learning architecture. *BMC Bioinformatics*, 19(1).

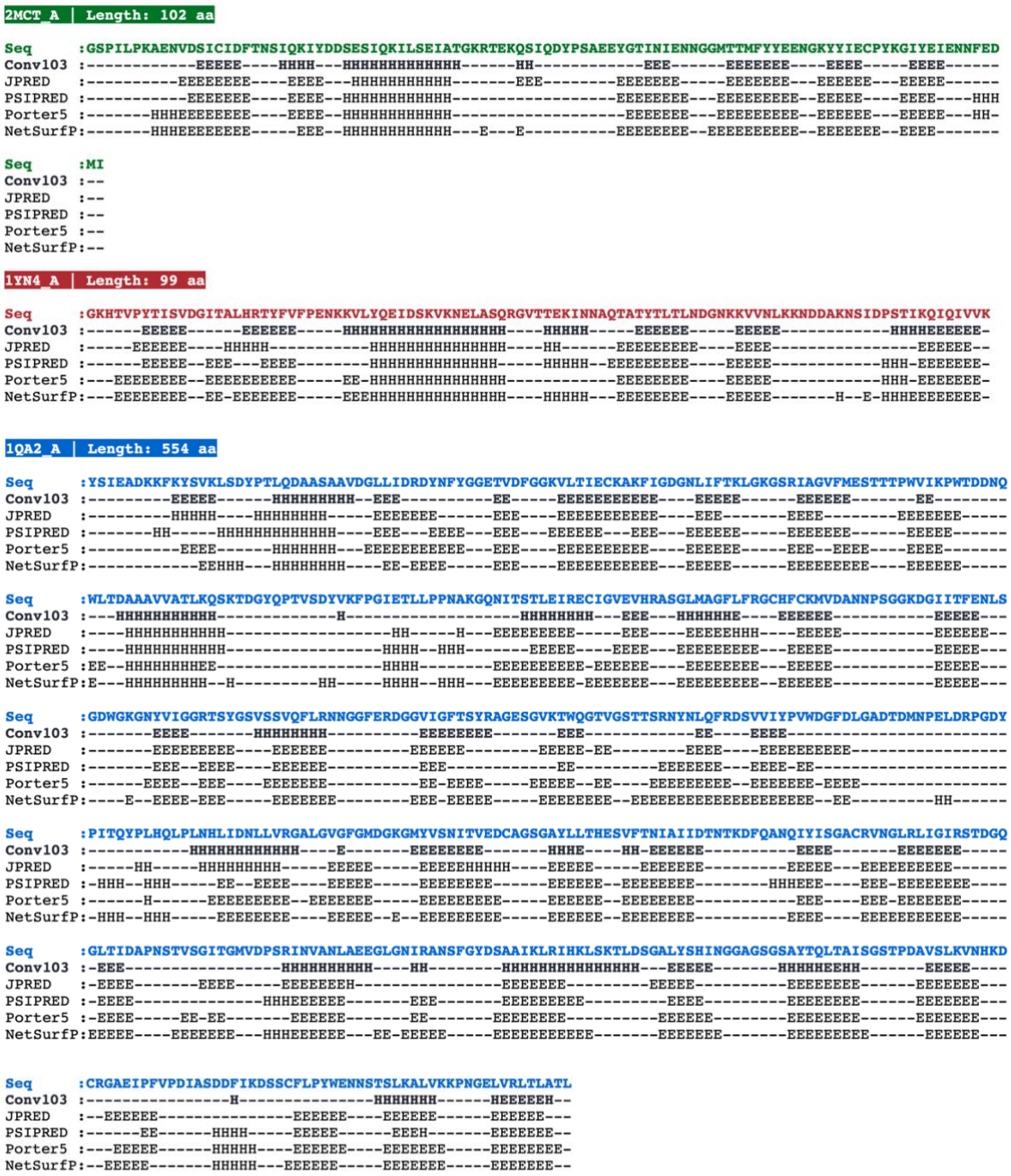
Supplementary Materials

Stage	Model	NN Type	3/8 State	Dimensions	Samples	Batch Size	Epochs	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)
							Scores At The End Of Training:				
1	1	RNN	3	100	1195	64	17	0.1664	65.48	0.1660	63.68
	2	RNN	8	100	1195	64	25	0.2913	50.35	0.2886	48.79
2	3	CNN	3	100	1195	64	3	0.7886	66.96	0.8242	64.72
	4	CNN	8	100	1195	64	3	1.3998	50.88	1.4143	49.85
3	5	RNN	3	103	1195	64	17	0.1581	67.51	0.1630	64.88
	6	RNN	8	108	1195	64	18	0.2893	50.61	0.2900	47.60
4	7	CNN	3	103	1195	64	3	0.7695	67.78	0.7976	66.32
	8	CNN	8	108	1195	64	3	1.3644	52.28	1.3599	52.26
5	9	CNN	3	103	10,000	100	3	0.7457	68.67	0.7558	68.20
	10	CNN	8	108	10,000	100	3	1.3199	53.80	1.3200	53.78
							Final Model Results:				
6	11	CNN	3	103	17,000	100	3	0.7411	68.86	0.7500	**68.44**
	12	CNN	8	108	17,000	100	3	1.3076	54.29	1.3076	**54.22**

***Supplementary Table 1*: Reported model parameters, accuracy and loss values at each stage.** Two models (three and eight-state) were generated at each stage, twelve models were generated in total during six stages of training.

Method	1QA2A *(l:554)*		1YN4A *(l:99)*		2MCTA *(l:102)*	
	Time (*seconds*)					
	am	**pm**	**am**	**pm**	**am**	**pm**
NetSurfP 2.0	107	493	90	80	94	110
JPRED4	259	212	120	95	89	132
PSIPRED	334	408	121	137	32	151
Spider3-Single	30	29	22	27	21	23
Porter5.0	682	706	230	252	216	247
Conv103	**7.07**	**5.82**	**6.4**	**4.35**	**7.03**	**4.82**

***Supplementary Table 2*: Time profiles in seconds for the prediction of three proteins (1QA2A, 1YN4A, 2MCTA) at two time intervals using different published prediction methods**. Speed tests took place in the morning (between 9-11am) and evening (between 7-10pm). Values ≥ 100 seconds are shaded orange, values <100 seconds and ≥ 15 seconds are shaded yellow, and those <15 seconds are shaded green.



***Supplementary Fig.1*: Three-state prediction outputs using published prediction algorithms available online** on three test proteins 2MCTA, 1YN4 and 1QA2A. 'H' represents α-helix structures, 'E' represents β-strand structures and '-' represents Other secondary structure types.

Tool/Package	**Available on:**	**Version**
Python	https://www.python.org/downloads/	3.7.1
Numpy (van der Walt, Colbert and Varoquaux, 2011)	https://numpy.org	1.16.2
SciKit Learn (Pedregosa et al., 2011)	https://scikit-learn.org/stable/install.html	0.20.1
Pandas (Mckinney, 2010)	https://pandas.pydata.org/getpandas.html	0.23.4
Keras (Chollet, 2015)	https://keras.io	2.2.4
TensorFlow (Abadi et al., 2016)	https://www.tensorflow.org/install	2.0.0-alpha0
Sublime (Sublimetext.com, n.d.)	https://www.sublimetext.com/3	3.2.1
Anaconda (Anaconda, 2018) Jupyter Notebook	https://www.anaconda.com/distribution/	1.9.7 5.7.4
MatPlotLib (Hunter, 2007a)	https://matplotlib.org/downloads.html	3.0.2

***Supplementary Table 3*: Packages used throughout this research, site location and version numbers**.