Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Bioinformatics

Predicting the subcellular location of proteins in eukaryotic cells

Udi Ibgui, Student Number 17082666

Department of Computer Science, University College London, London, WC1E 6BT, United Kingdom

COMP0082 Coursework

Abstract

Motivation: In recent years, large numbers of protein sequences have become available. These need to be efficiently analysed to identify the proteins' functionality. Protein subcellular locations are one of several characteristics used to solve this. Traditionally, machine learning algorithms are implemented to infer the subcellular location. The algorithms use features extracted from the sequences' amino acid, with prediction quality correlating with the feature's quality. Hence, domain expertise is needed, preventing some researchers from tackling this problem. Furthermore, most algorithms do not consider the sequence order. We try to solve these issues by introducing a deep learning algorithm that can be trained with limited computational power and data, whilst still making accurate predictions.
Results: We train a BiLSTM with a fully connected network attached to classify sequences into one of five possible locations. We achieved an accuracy greater than 70% on our test set. Further, we get an F1 score of 0.70 and a Matthew correlation coefficient of 0.68.
Contact:zcemyib@ucl.ac.uk
Supplementary information: Sequences Dataset available at *http://www0.cs.ucl.ac.uk/staff/D.Jones/coursework/*

1 Introduction

With the advancements of high-throughput sequencing, an ever-increasing number of protein sequences has become available. Functional annotation projects aim at elucidating the potential roles these proteins play in a cellular context. Due to the large amount of data available, these projects require efficient analysis methods.

Proteins have evolved to function optimally in specific subcellular locations (Murphy *et al.* (2000)). Therefore, a principal property that may illuminate a protein's functionality is its subcellular location. This paper aims at introducing an efficient method that can infer these protein subcellular locations.

In the past, experimental methods, such as fluorescence micro-visualisation and electron-microscopy were commonly used. Due to the increase in the scale of sequence data available, these methods are no longer desirable as they can be costly and cumbersome to implement. Instead, computational methods such as machine learning (ML) have increased in popularity (Zou *et al.* (2007)). Emanuelsson *et al.* (2000), showed that fundamental properties of the sequence can be used as indicators for the protein's subcellular location. These include properties like the sequence's amino acid composition and their molecular weight. ML techniques, such as Support Vector Machines, have traditionally used these properties to determine protein subcellular locations. To do so, researchers tend to extract various features that could aid in prediction. The quality of the model then is heavily reliant on the quality and relevance of the extracted features. Though slightly outdated, Dönnes and Höglund (2004) provides a comprehensive survey on protein subcellular localisation methods. There, they describes these methods in more detail.

Whilst these classic ML methods have shown effective results, they have two main limitations. Firstly, before applying the ML algorithm, a high level of domain expertise is required to perform an effective feature extraction. This creates a high barrier of entry and makes the development of such algorithms limited to a niche of researchers.

Secondly, most of the traditional ML algorithms focus solely on individual amino acid properties and ignore the sequence order information. This seems unnatural and likely to miss out on a key factor that can indicate a protein's subcellular location.

This paper aims to make a contribution by addressing both of these issues. We do this by applying a Recurrent Neural Networks (RNN) to the sequences. This is a form of a neural network used to process sequential information. In particular, a Bidirectional Long Short Term Memory (BiLSTM) architecture is explored.

We investigate a naïve approach, where no features are extracted. Instead, sequences are embedded and processed as are, with the network focusing on sequence order to make its prediction. This method, therefore, allows for researchers with less expertise in bioinformatics to begin tackling the problem. Ultimately, this can facilitate knowledge sharing from experts in other domains.

2 Materials and Methods

2.1 Dataset

A dataset in the FASTA format was provided for this project by the University College London (UCL) Bioinformatics Group. It contained non-homologous protein sequences classed by four major eukaryotic subcellular locations and labelled as such. Namely, 'cytosolic', 'secreted', 'nuclear' and 'mitochondrial'. Several examples of prokaryotic proteins, which can contaminate samples during sequencing, were also provided. These were labelled as 'other'. Overall, our dataset contained 11224 sequences from five classes. Table 1 shows the number of samples of each class, as well as the percentage of the data each class encompasses. It was noted that the data was unbalanced.

Subcellular Location	Number Of Examples	Percentage of Dataset
Cystolic	3004	26.76
Secreted	1605	14.30
Nuclear	1299	11.57
Mitochondrial	3314	29.53
Other	2002	17.84

Table 1. Distribution of different classes in the dataset.

Table 2 describes the distribution of sequence lengths within the full dataset. A mean of 526.11 with a wide variance of 476.38 is observed. As 75% of the sequence are shorter than 650 amino acids, the high variance is due to a small number of extreme outliers. Further analysis showed that more than 89% of sequences are composed of less than 1000 amino acids.

Count	Mean	Std	Min	25%	50%	75%	Max
11224	526.11	476.38	11	262	407	650	13100

Table 2. Sequences Description

Finally, Figure 1 shows the frequency distribution of the different amino acids present within the dataset. There is a negligible number of Aspartic acid/Asparagine (B), Selenocysteine (U) or unknown (X) amino acids. They were therefore ignored and removed from the sequences in this study.

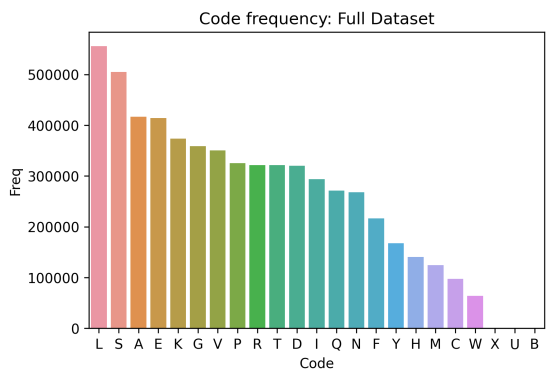

Fig. 1: Distribution of the different amino acids found within the dataset. The number of of B, U and X amino acid is negligible and they were therefore ignored in our analysis.

2.2 Architecture

RNN are a form of neural networks suitable for sequential data. When presented with long sequences, RNNs become very deep and tend to fail due to the "vanishing gradient" problem. This is where as the gradients backpropagate through multiple layers, they become numerically unstable. To overcome this, different modifications to the basic architecture can be adopted. The LSTM cell, first introduced by Hochreiter and Schmidhuber (1997), was designed explicitly to avoid this long-term dependency problem. The cell contains three neural network gates that regulate the flow of information through the cell. These allow the network to 'remember' values over arbitrary time intervals, overcoming the vanishing gradient problem.

LSTMs, and RNNs in general, were extended with the idea of Bidirectional RNNS. Originally introduced by Schuster and Paliwal (1997), Bidirectional LSTMs (BiLSTM) are composed of two LSTM cells running in opposite directions. Given an input $x_1, x_2 ... x_t$, one cell process information in the direction of $x_1, x_2 ... x_t$ whilst the other processes information in the reverse direction, $x_t ... x_2, x_1$. Such an architecture allows to simultaneously provide both backwards and forwards states to the network. This is a natural extension to problems such as protein sequences classification, as the network can process the current amino acid in the context of both past and future amino acids.

In our implementation, one LSTM network receives the first amino acid whilst the other LSTM receive the last. Subsequent amino acids are then passed through the networks and processed. At each pass, the amino acid is non-linearly transformed and the network's memory is updated by the LSTM cell. The final output of the forward and backward LSTM is then concatenated. Figure 2 shows a simplified diagram depicting the BiLSTM architecture. The interested reader is referred to Goodfellow *et al.* (2016) for a more formal description of the LSTM cell.

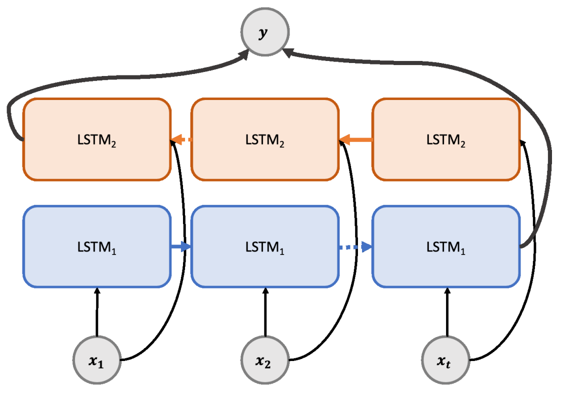

Fig. 2: A diagram showing a BiLSTM. Two networks (represented by blue and orange) process the input sequence $x_1, x_2 ... x_t$ in opposite order. The final output from both networks is contacanated and passed further. This is represented by y.

Once processed through the BiLSTM, the concatenated output is fed forward to a two-layers fully-connected neural network. Finally, a probability distribution over the five classes is computed using a softmax function. Using a categorical cross-entropy loss function, the loss between the predicted and actual class is computed. By 'backpropagating through time' (BTT), the model's parameters are slowly updated to minimise this loss.

2.3 Hyperpartameters

Several hyperparameters were explored to generate the most efficient model. First, the amino acids making the sequence had to be numerically embedded for processing. Initially, one-hot encoding was used as our vocabulary was composed of only 20 amino acids (B, U and X were removed). Experiments showed though, that this embedding resulted in a slow learning model. Instead, randomly generated numerical embeddings were used. These were considered trainable parameters by the model and were updated during training. The size of the embedding was then investigated as one of the hyperparameters. Further, the number of LSTM and FCN nodes were also explored, as well as dropout and the learning rate. A batch size of 16 was selected.

Initial scoping was first manually performed to find suitable ranges for the hyperparameter search. As Google Colab was used to access GPUs, we had a limited time-allocation to the server. To manage an effective search in this limited timeframe, we used the Hyperband search method (Li *et al.* (2018)), which can be up to 30 times faster than Bayesian optimisation. The Hyperband algorithm uses adaptive computation, allocating more resources to promising configurations while quickly eliminating poor ones through a process of sequential halving. Table 3 shows the hyperparameters explored, the ranges searched over and their final optimal values.

	Embeddings	LSTM Units	Layer 1 Nodes	Layer 2 Nodes	Dropout	Learning Rate
Range Explored	32-512	32-512	64-256	32-128	0.1-0.4	$1 * 10^{-2}$ - $1 * 10^{-4}$
Final Value	288	352	192	64	0.1	$1 * 10^{-3}$

Table 3. Hyperparameter search ranges and their final values.

2.4 Training and Evaluation

Before training, the sequences were preprocessed. Whitespace was added between each amino acid code and the B, U and X amino acids were removed. To allow for batch processing, all sequences were truncated and padded to a maximum length of 1024 amino acids. This follows a similar method employed by Sønderby *et al.* (2015). Emanuelsson *et al.* (2000) showed that both the N- and C- terminal regions contain sorting signals that can indicate the subcellular location. Therefore, rather than removing amino acids from the end of the sequence, we removed those from the middle. 10.55% of the data was truncated. We then transformed the amino acid codes into numerical embeddings

The data was randomly shuffled and then split 90/10 into training and test sets. 5-fold cross-validation was used for initial training and exploration of possible overfitting issues. In each fold, 20% of the training dataset was used as a validation set. A maximum of 50 epochs was set for each training round, together with early stopping to avoid overfitting. Having observed no overfitting through cross-validation, we retrained the model on the full training dataset, with a 10% split for validation.

Our model was implemented using the TensorFlow and Keras packages. For GPU accesses, we used Google Colab, and were provided with an Nvidia Tesla-T4 GPU. The Adam optimiser was used, along with the categorical cross-entropy loss function. To overcome the class imbalance issue mentioned before, the training set class weights were calculated. This was passed to the loss function, which reweighed each class accordingly.

After training, the model's weights and amino acids embeddings were saved. The model was then reloaded and used to predict sequence classes on the test set. We recorded the loss, accuracy, F1 score, confusion matrix and Matthew correlation coefficient for these results.

Finally, the model was used to predict the classes of the competition set. This consisted of 20 unseen sequences provided by the UCL Bioinformatics Group. Our model outputted logits and a softmax function was applied on these outputs. This provided us with a probability distribution across the five classes. For each sequence, the class with the highest probability was selected as its predicted label.

A measure of confidence was assigned to each of the predictions. As there were five classes, we were lower bounded with a probability of 0.2 per class. Hence, predictions in the range of 0.2-0.29 were assigned as LOW confidence. Predictions in the range of 0.29-0.36 were assigned as MEDIUM, and anything above was assigned as HIGH level of confidence. The model's predictions for the competition set can be found in the Appendix.

3 Results

Based on the 5 accuracy estimation computed during the 5-fold cross-validation, we found a mean accuracy of 67.91%. Using a t-distribution with 4 degrees of freedom, we found the confidence interval to be ±0.43, giving a boundary of [67.48, 68.34]. As the boundaries were less than 1% away from the mean, it was concluded that there was no overfitting issue. The model was therefore retrained on the full training set with 10% from it set for validation. This took 9 epoch and totalled 3 hours of processing. The newly trained model was then evaluated on the test set. Table 4 records the different accuracies and loss scores achieved.

	Cross Validation Mean	Validation Set	Test Set
Loss	0.8751	0.8681	0.8405
Accuracy	67.91	68.11	70.44

Table 4. Loss and Accuracies obtained with different sub sets.

For evaluation on the test set, the F1 score was calculated. This is defined as

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

where

$$Precision = \frac{\text{TP}}{\text{TP + FP}} \; and \; Recall = \frac{TP}{TP + FN}$$

TP,FP, and FN refer to True Positive, False Positive and False Negative (and TN is true negative). Table 5 shows the scores obtained for both the test set as a whole and for individual classes. When calculated, the scores were weighed according to the class distribution.

	Precision	Recall	F1-Score
All Classes	0.71	0.70	0.70
Cyto	0.52	0.65	0.58
Secreted	0.91	0.89	0.90
Mito	0.85	0.77	0.81
Nucleus	0.71	0.52	0.60
Other	0.78	0.87	0.82

Table 5. F1 metrics per class and for the whole test set.

Finally, a confusion matrix was constructed, shown in Figure 3. This allowed us to identify which classes the model was doing well in and in which less so. To further quantify these results, we used a weighted Matthew Correlation Coefficient (MCC). The MCC is considered a better measure than F1, as explained by Chicco (2017). This allowed us to measure the quality of our model by considering the different sizes of the positive/negative classes. In the general binary classifier case, it is defined as follows;

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FNN)}}$$

As we our model was a multi-label classifier, we usedthe R_k statistic. This is a modified MCC for the multi-label class. See Gorodkin (2004) for further details. A score of 0.6787 was recorded.

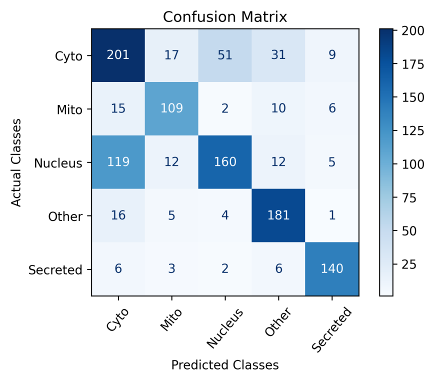

Fig. 3: Confusion matrix for showing expected and predicted labels.

4 Discussion and Future Work

In our paper, we have applied a naïve BiLSTM to protein sequences to predict their subcellular locations. In deep learning term, our dataset was relatively small, and our model could likely be improved with more data. Nonetheless, given the current dataset, we achieved an accuracy of more than 70% on our test set and an F1-Score of 70% across all classes. The MCC was almost 0.7, indicating a positive relationship between the predictions and actual labels.

Based on the F1 score and the confusion matrix, the model best performed on sequences coding for secreted proteins (F1 = 0.9). This may indicate that since these proteins are secreted, their sequences are considerably different from that of the other classes, which remain within the cell. Accordingly, though, a similar score should have been recorded for the 'other' class. Instead, a lower F1 score of 0.82 was observed. It may be argued that this lower score is reasonable as the prokaryotic sequences were not of a distinct category. Rather, many sequence classes were included, with the only commonality being that they originate in prokaryotes.

The model also performed reasonably with the 'mitochondrial' class (F1 = 0.80). Yet, it did not perform very well for predictions of the 'cytosolic' and 'nucleus' classes. On these, the model achieved an F1 scores of only 0.58 and 0.60 respectively. We noted the model misclassified 155 sequences as belonging to the 'cytosolic' class. In particular, it misclassified 119 'nucleus' sequences as 'cytosolic'. This accounted for 76.77% of errors were the model misclassified sequences as 'cytosolic'. Further, the model misclassified 51 'cytosolic' sequences as 'nucleus' sequences, accounting for more than 47% of the misclassifications of actual cytosolic sequences. This bias may potentially indicate a similarity between the two protein sequences and requires further investigation.

Having shown reasonable success in the five class case, it would be interesting to explore how the model performs when trained and tested on more localisation classes. One could also explore the model's performance when presented with only eukaryotic cells sequences. The results of such experiments could indicate similarities or differences between the different cell locations and types.

Finally, several modifications should be explored to potentially improve the model. We present these in ascending order of alteration to the current model:

• In this investigation, we truncated and padded all sequence to 1024 amino acid sequences. This was done to reduce computation cost and allow for batch processing. We shortened slightly over 10% of the dataset. The effects of using shorter sequences, such as the mean length (526 amino acids), or the maximum length, could be explored. This could indicate how much information is encoded by the middle amino acids. Alternatively, all sequences could be kept in their original length and investigated. This would not allow for batch processing and be hugely expensive to compute.

• Further complexity could be added to the model. For example, Convolution Neural Networks (CNN) have shown particular success in feature classification. Although originally developed for image analysis, these have been successfully applied to sequential problems such as natural language by Kim (2014). A similar application could be explored for protein sequences.

• Newer models could also be explored. In particular, the Transformer model could be investigated for this use case. Since its development by Vaswani *et al.* (2017), the Transformer model has revolutionised most natural language processing tasks. This model works particularly well due to its ‘Attention’ mechanism, which allows the neural network to place a greater focus on more important states, whilst fading out the less important ones.

• Whilst we aimed for a naïve implementation that did not require any feature extractions, a hybrid method can be explored. This concept requires combining outputs of the neural network with other forms of extracted feature data, see Ma *et al.* (2019). It is expected that through this a more general model could be achieved. An interesting avenue to explore would be an investigation of the model’s performance when features of increasing complexity are added to it.

5 Conclusion

In this paper, we show a naïve BiLSTM that can predict the subcellular location of proteins in eukaryotic cells. We used a relatively simple structure and categorise sequences over five classes. We trained a model together with embeddings of amino acids and evaluated a test set. Accuracy, F1, Precision, Recall and the MCC were used for this evaluation.

Whilst we did not achieve state-of-the-art results, our model was successful, with a final accuracy of 70.44%. It performed best in the ’secreted’ class, followed by the ’mitochondrial and ’other’ classes. The model underperformed on the ’cytosolic’ and ’nucleus ’ classes and this requires further investigation.

We have managed to show that sequence order is an effective indicator of protein subcellular location and should be used as a feature for protein localisation tasks. Whilst the model is most likely to improve by incorporating other sequence features, we showed that reasonable results can be achieved by passing only the protein sequence to a neural network. We hope this will motivate others to follow our approach of using deep learning techniques to solve such important problems.

References

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. *BioData Mining*, **10**.

Dönnes, P. and Höglund, A. (2004). Predicting protein subcellular localization: Past, present, and future. *Genomics, Proteomics Bioinformatics*, **2**, 209 – 215.

Emanuelsson, O. *et al.* (2000). Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of molecular biology*, **300**(4), 1005—1016.

Goodfellow, I. *et al.* (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gorodkin, J. (2004). Comparing two k-category assignments by a k-category correlation coefficient. *Computational Biology and Chemistry*, **28**(5-6), 367–374.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, **9**(8), 1735–1780.

Kim, Y. (2014). Convolutional neural networks for sentence classification.

Li, L. *et al.* (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization.

Ma, Y. *et al.* (2019). Deeploc: A location preference prediction system for online lodging platforms. In Y. Sun, T. Lu, Z. Yu, H. Fan, and L. Gao, editors, *Computer Supported Cooperative Work and Social Computing*, pages 618–630, Singapore. Springer Singapore.

Murphy, R. *et al.* (2000). Towards a systematics for protein subcellular location: Quantitative description of protein localization patterns and automated analysis of fluorescence microscope images. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, **8**, 251–9.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, **45**(11), 2673–2681.

Sønderby, S. K. *et al.* (2015). Convolutional lstm networks for subcellular localization of proteins. *Lecture Notes in Computer Science*, page 68–80.

Vaswani, A. *et al.* (2017). Attention is all you need.

Zou, L. *et al.* (2007). Prediction of subcellular localization of eukaryotic proteins using position-specific profiles and neural network with weighted inputs. *Journal of Genetics and Genomics*, **34**(12), 1080–1087.

6 Appendix: Challenge sequences Predictions

CHALLENGE SEQ01	Othr	Confidence High
CHALLENGE SEQ02	Extr	Confidence High
CHALLENGE SEQ03	Mito	Confidence High
CHALLENGE SEQ04	Cyto	Confidence Low
CHALLENGE SEQ05	Mito	Confidence Low
CHALLENGE SEQ06	Mito	Confidence High
CHALLENGE SEQ07	Extr	Confidence High
CHALLENGE SEQ08	Mito	Confidence Low
CHALLENGE SEQ09	Extr	Confidence Medium
CHALLENGE SEQ10	Extr	Confidence High
CHALLENGE SEQ11	Cyto	Confidence Medium
CHALLENGE SEQ12	Extr	Confidence High
CHALLENGE SEQ13	Extr	Confidence High
CHALLENGE SEQ14	Cyto	Confidence Medium
CHALLENGE SEQ15	Cyto	Confidence Low
CHALLENGE SEQ16	Cyto	Confidence Low
CHALLENGE SEQ17	Othr	Confidence High
CHALLENGE SEQ18	Extr	Confidence High
CHALLENGE SEQ19	Othr	Confidence Medium
CHALLENGE SEQ20	Nucl	Confidence Medium