Bioinformatics

Predicting the subcellular location of eukaryotic proteins

Jonas Vetterle, Student number 110043050

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

Coursework of COMPGI10 Bioinformatics (MSc Machine Learning optional module)

Abstract

Motivation: The subcellular location of eukaryotic proteins can provide insights into the function of proteins. Recurrent Neural Networks (RNNs) have shown to perform well on classification tasks when sequential data is observed. This paper explores the use of RNNs in classifying proteins as either cytosolic, secreted, nuclear or mitochondrial on the basis of only the sequence of underlying amino acids. The focus of this paper is on finding an RNN architecture that is both high performing and tractable using limited computational resources.
Results: Using a special type of RNNs, a GRU (Gated Recurrent unit) with a single 64 unit layer, it is possible to achieve a cross-validated prediction accuracy of 69.58% with a 95% confidence interval of [68.36; 70.80]. As a by-product, the method also yields low dimensional vector representations of amino acids which allows to cluster similar amino acids together.
Contact: ucabjxv@ucl.ac.uk
Code: Available at https://github.com/jonvet/bioinformatics

1 Introduction

Due to the constant development of ever faster and cheaper ways of sequencing DNA, the number of protein sequences available for analysis has been growing exponentially in recent years. While the amount of raw data available is growing, much remains unknown about those sequences. The biological function of proteins which is determined by their three-dimensional structure is of particular interest to researchers in academia and industry. While manually determining protein structure is achievable, it is not viable on a large scale. In the light of the increasing data available there is therefore a need to find automated ways of predicting protein structure from raw sequence data.

This paper deals with a related problem, which is to predict the subcellular location of eukaryotic proteins. The location of a protein within a cell might yield clues as to the properties and functions of a protein. This is because different locations within a cell provide different biochemical environments which influence the function that proteins will carry out (Murphy *et al*, 2000). The problem of classifying proteins according to their location has been studied for many years. Pioneering work by Nakashima and Nishikawa, 1994, for example studied the amino acid composition and residue-pair frequencies of proteins, on which basis intra- and extracellular proteins were classified. Further research by, among others, Emanuelsson *et al*, 2000 has been directed towards classifying the subcelullar location of proteins. This paper makes a contribution by assessing how a particular kind of recurrent neural networks, the Gated Recurrent Unit, performs in predicting subcellular protein locations.

2 Approach

There are a multitude of machine learning models that have been applied to the problem of classifying subcellular protein location, and classification problems in other domains. For the sake of broadly distinguishing between methods, they can be divided into methods that require some kind of feature engineering (e.g. Support Vector Machines and Logistic Regression), and methods where no feature engineering is required (e.g. Artificial Neural Networks).

Support Vector Machines (SVMs) fall into the former category and have been successfully applied to this problem. In the case of SVMs, it is necessary to manually extract features from the data. The better the features extracted, the better the performance of the model *ceteris paribus*. Features are usually selected on the basis of prior domain knowledge. In the case at hand, possible features include for example the counts of individual amino acids in a protein, or the total mass of all amino acids in a sequence. More bespoke features may include binary variables that indicate whether particular patterns of acids occur in a sequence, or the occurrence of particular patterns at the beginning or end of a sequence. Examples in the literature include Matsuda *et al*, 2005 who successfully used SVMs for protein location classification. Features used included local compositions of amino acids and twin amino acids, as well as local frequencies of distance between successive (basic, hydrophobic and other) amino acids. The author obtained a cross-validated accuracy of over 87% for eukaryotic proteins. Other examples include Park and Kanehisa, 2003, who use an ensemble of SVMs based on amino acid, amino acid pair, and gapped amino acid pair compositions.

The approach taken in this paper is to use a Gated Recurrent Unit (GRU)

which is a special case of a Recurrent Neural Network (RNN). While it is beyond the scope of this paper to describe this method in detail, a summary is given in the next section. Unlike methods such as SVMs or multiclass logistic regression, RNNs require very limited feature engineering, or none at all. Instead, the raw sequence of amino acids can be fed into the RNN, which will then construct a numerical representation of the sequence. The RNN inspects each element of the sequence, and updates the representation of the sequence after each element. Importantly, the order of elements in a sequence matters; if two polypeptide chains have the same counts of amino acids, but the amino acids occur in a different order, then the representations of the sequences will be different. Early examples for the use of neural networks in protein location prediction include Reinhardt and Hubbard, 1998 who achieve an accuracy of 66% for the location of four locations of eukaryotic proteins. More recent work making use of RNNs include Sonderby *et al*, 2015, who deploy a variant of an RNN (Long Short Term Memory) network. Mer and Andrade-Navarro, 2013 use a combination of SVM and neural network to predict protein location.

3 Data

3.1 Train data

The dataset used for training the RNN comprises 9,222 non-homologous eukaryotic proteins which belong to one of four classes depending on their location within a cell - cytosolic, secreted, mitochondrial and nuclear. As shown in Table 1, the mean sequence length of proteins is 425. It can also be observed that there is great variance in the length of proteins, with a minimum length of 11 amino acids, and a maximum length of 13,100. The great majority of sequences is shorter than 2,000 amino acids, with the 75% percentile being 691.

Table 1. Train data - descriptive statistics

Count	Mean	Std	Min	25%	50%	75%	max
9,222	546.8	514.3	11	235	425	691	13,100

I provide a further breakdown of the training data by class and sequence length in Table 2 below. As can be seen, the data set is unbalanced, with Cyto and Nucleus accounting for three quarters observations. Further observations are that most of the long sequences are accounted for by cytosolic and nuclear proteins.

Table 2. Train data - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[0, 100)	44	619	67	35	765
[100, 500)	1,379	716	929	1,626	4,650
[500 - 1,000)	1,047	197	278	1,165	2,687
[1,000 - 1,500)	350	32	20	316	718
[1,500 - 2,000)	104	22	3	106	235
[2,000 - inf)	80	19	2	66	167
Total	3,004	1,605	1,299	3,314	9,222

3.2 Blind test data

The test data comes unlabelled and will be used to evaluate the performance of the final model. It comprises 20 sequences whose descriptive statistics are shown in Table 3 below. The test set is not dissimilar to the train set in terms of the mean and standard deviation of sequence lengths.

Table 3. Test data - descriptive statistics

Count	Mean	Std	Min	25%	50%	75%	max
20	546.6	423.4	141	306	457	596	1,876

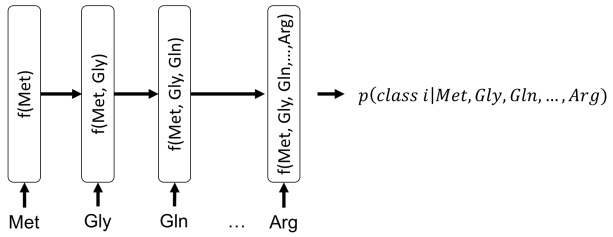
4 Methods

4.1 Gated Recurrent Unit (GRU)

The GRU cell was first introduced by Cho *et al*, 2014 in the context of natural language translation. Like Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) GRUs are a type of Recurrent Neural Network (RNN) that is suited for sequential input data such as protein sequences. Unlike plain vanilla RNNs, GRU and LSTM cells have an internal memory which enables them to store information over longer sequences. In the context of this paper, the internal memory can be thought of as a summary of the sequence of amino acids that the GRU has encountered. The main difference between GRU and LSTM cells is that the former has a simpler structure and is therefore somewhat faster to train.

A schematic overview of the model deployed in this paper is shown in Figure 1. The model receives as input one amino acid at a time. Upon observing the input, the GRU cell updates its internal memory by a nonlinear function $f(.)$ of the current amino acid, and the previous memory state (i.e. it is a function of the sequence of amino acids observed so far). In particular, it decides which parts of its memory it should reset, and how much of the current input it should write to memory. The reader is referred to Cho *et al*, 2014 for a formal description of the GRU cell. After observing the full sequence, the internal memory represents a summary of the observed sequence.

Fig. 1. Schematic functionality of the model



The last memory state is then used to compute a probability distribution over the four classes. This is done by first applying a nonlinear transformation to the memory state (fully connected layer with ReLu activation, (see Nair and Hinton, 2010)), followed by a linear transformation (fully connected layer). Finally, a softmax operation is applied to the logits resulting from the linear transformation to obtain a probability distribution. The model is trained by minimising the cross-entropy between the estimated probabilities and the true class labels and adjusting the weights by backpropagation.

Further refinements include a dropout wrapper (see Srivastava *et al*, 2014) around the GRU cell and the nonlinear layer to avoid overfitting. To increase the stability of the model during training time, gradients are clipped at a norm of 2, and batch normalisation (see Ioffe and Szegedy,

2015)) is applied to the outputs of the GRU, and the outputs of the non-linear layer. The model updates are performed using batch updates and the ADAM optimiser. The learning rate is decayed with a factor of 0.96 every 100 steps to facilitate convergence.

4.2 Hyperparameters

Initial hyperparameter search was conducted using a random sample of the data as validation set. Table 4 below shows the minutes per epoch need to train different GRU and LSTM models on a CPU. The models vary by the number of hidden units, whether or not proteins sequences are truncated as described above, and whether or not the model is uni- or bidirectional. While no attempt has been made to obtain cross-validated prediction accuracies for all of these models, trials using a single validation set suggest that GRU models with additional layers are able to achieve 2-3 percentage points more accurate results. On the other hand, GRU models with more hidden units, or additional layers tend to overfit more easily, even with high dropout probabilities and L2 weight regularisation. In light of substantially less time needed for training, a single layer with 64 units was chosen as final model.

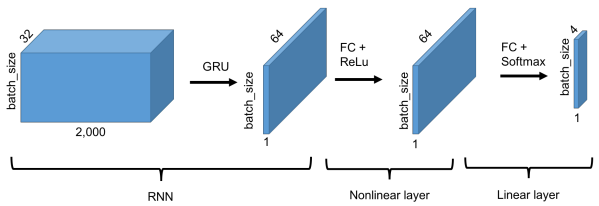
When performing hyperparameter search, it can be shown that randomly selected values is both empirically and theoretically more superior to performing a grid search. This is because the space of different hyperparameters will be explored more thoroughly (see Bergstra and Bengio, 2012). This is therefore also the approach taken in this paper. Different random learning rates between 10^{-4} and 10^{-1} were evaluated. A learning rate of 0.01 was found to be a good compromise between training speed and accuracy obtained. Similarly different dropout probabilities and parameter values for L2 regularisation were tested. With dropout, L2 regularisation was found not to yield an additional improvement in the validation accuracy.

The final hyperparameters chosen for training the model are:

- Batch size: 32
- Dropout probability: 30%
- Learning rate: 0.01
- GRU layers: 1
- GRU units: 64
- Embedding size: 32

An illustration of the network architecture is presented in Figure 2 below.

Fig. 2. Network architecture



4.3 Biological and computational considerations

As mentioned in Section 2, the method used in this paper does not rely on feature engineering. If specific patterns of amino acids are indicative of the location of the protein within the cell, then these patterns should be picked up by the GRU model. For this reason, no information other than the raw amino acid sequence was used as input. It is conceivable to enhance the input by also feeding the GRU information about the properties of amino acids such as the mass, hydrophobicity or the length of the protein.

Specific information contained in the sequence of amino acids itself, that are of particular importance for this task, are patterns at the beginning and the end of proteins. Proteins need to be sorted to the right subcellular compartment to perform fulfil their function. Dong *et al*, 2003 find that sorting usually relies on the presence of an N-terminal targeting sequence. For example, in extracellular proteins, sorting peptides are often found in the first 40 amino acids. Similarly, Matsuda *et al*, 2005 find that both N-terminal and C-terminal parts of sequences were helpful in prediction the subcellular location of proteins.

A simplifying assumption in this paper is that most of the important information about the subcellular location can be found in the beginning and the end of any sequence. Therefore, all sequences have been truncated to be at most 2,000 amino acids long. This method has been employed by other researchers such as Sonderby *et al*, 2015 who truncated sequences at length 1,000. As shown in Table 2, more than 98% of proteins are shorter than 2,000 acids, and are therefore not affected by this modification. For the less than 2% of proteins that are longer than 2,000 acids, this may lead to a lower prediction accuracy as only part of the information contained in the sequence is used. However, the benefit in terms of computational speed of training the model is substantial (see Table 4) and are deemed to outweigh the cost of truncating very few sequences.

In light of the above discussion, when truncating long proteins, the first 1,900 and the last 100 amino acids were concatenated. This mitigates the information that is lost as the potentially helpful information in the C-terminal parts are preserved.

Table 4. Training times per epoch of different models

Model	Units	Sequence	Directional	Min per epoch
GRU	1x64	truncated	uni	11.81
GRU	1x128	truncated	uni	13.15
LSTM	1x64	truncated	uni	14.23
GRU	2x128	truncated	uni	31.65
GRU	2x128	truncated	bi	37.98
GRU	1x64	full	uni	79.16

4.4 Training the model

The model is trained using 5-fold cross-validation. That is, the data set is shuffled randomly and divided into five subsets. Summary statistics of the five validation sets are shown in Tables 9 to 13 in the Annex. I observe that the individual validation sets are balanced in terms of protein location, but that Fold 1 is somewhat skewed towards longer sequences.

Training is then carried out using four of the subsets as training set and the last subset as validation set to assess the model accuracy. This process is repeated five times such that each subset is used once as validation set. Train and validation error and accuracy are tracked after each epoch, and the training is stopped once the model starts overfitting (i.e. when the validation accuracy plateaus), which tends to happen after around 15 epochs (see Figure 6 in the Annex).

4.5 Amino Acid Embeddings

Note that the raw data is a sequence of one-letter codes. However the GRU needs a numerical input for its calculations. A natural choice would be to use one-hot encodings. However the approach taken in this paper is to make the embeddings 32-dimensional and declare them a trainable variable. This means that the GRU will be able to adjust the elements of each embedding vector as it sees fit. It will be shown later that by the end of the training the embedding vectors are adjusted in such a way that similar amino acids tend to be close together in vector space.

5 Results

5.1 Accuracy

I present the accuracy in terms of correct predictions on the validation set for each fold in the first row of Table 5. Using these estimates, I construct the 95% confidence interval as

$$\bar{x} \pm t^* \frac{s}{\sqrt{n}}$$

where $\bar{x}$ is the sample mean of 69.58, s sample standard deviation of 0.98 and t^* is the two-sided critical value with 4 degrees of freedom (2.776). The 95% confidence interval is $[68.36; 70.80]$. With the boundaries of the confidence interval less than 1.5 percentage points away from the mean, the prediction accuracy does not appear to vary much across sub samples of the data.

Table 5. Mean accuracy across validation folds

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
Accuracy - total	70.6%	68.2%	69.1%	70.4%	69.6%	69.6%
Accuracy - <= 2,000	70.4%	68.3%	69.1%	70.4%	69.7%	69.6%
Accuracy - >2,000	82.4%	65.9%	69.0%	69.0%	68.4%	70.9%

Reassuringly, while the predictions for sequences longer than 2,000 amino acids are less precise (apart from the first cross-validation fold), the difference compared to shorter sequences is marginal. This gives some indication that the truncation of amino acids does not have a markedly negative impact on classification accuracy. The fact that the prediction accuracy for long sequences in Fold 1 is substantially higher than in the other folds may in part be explained by the composition of the folds. As can be seen in Tables 9 to 13 (see Annex), Fold 1 contains relatively more long sequences than short sequences. It may be that Fold 1 contains more "easy to classify" sequences than the other folds.

5.2 Precision, Recall, F1 score and AUC

The model is further evaluated using the F1 score, which is defined as

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

with

$$precision = \frac{tp}{tp + fp}, recall = \frac{tp}{tp + fn}$$

and where tp, fp, fn refer to the number of true positives, false positives and false negatives, respectively. The mean precision, recall and F1 scores across all five cross-validation folds are presented in Table 6 below together with the AUC (area under the curve). Please see the Annex for a visualisation of the ROC (Receiver operating characteristic) curves for each of the validation folds.

Table 6. Mean cross-validated precision, recall, F1 score and AUC

	Cyto	Secreted	Mito	Nucleus
Precision	0.58	0.91	0.78	0.69
Recall	0.66	0.92	0.72	0.61
F1 Score	0.61	0.92	0.75	0.64
AUC	0.71	0.95	0.84	0.73

We can infer that the model performs best on secreted and mitochondrial proteins, whereas it perform worse on cytosolic and nuclear proteins. A potential reason for why the model performs better at predicting mitochondrial proteins than cytosolic ones is that mitochondrial proteins have to reach a specific location, the Mitochondria. This may suggest that they have more distinct N- and C-terminal target sequences than cytosolic proteins, which occur everywhere in the Cytosol.

5.3 Confusion matrices

One way of visualising the degree of certainty of the model predictions are confusion matrices. In Table 7, I show the predicted class labels (column) versus the true class labels (row). High values on the diagonal indicate that the model made correct predictions, and high off-diagonal values indicate mistakes. The values are averaged over the 5 validation folds.

Table 7. Mean cross-validated confusion matrix (row: ground truth, column: prediction

	Cyto	Secreted	Mito	Nucleus	Total
Cyto	**396.6**	13.6	29.4	163.0	600.6
Secreted	13.4	**295.8**	7.2	4.6	321.0
Mito	44.8	7.2	**187.0**	20.8	259.8
Nucleus	234.2	7.4	15.0	**406.0**	662.6
Total	687.0	324.0	238.6	594.4	**1,844.0**

The confusion matrix is a different way of looking at the F1 score. For example, across the five cross-validation folds, on average 396.6 sequences were correctly predicted to be cytosolic, out of a total of 687.0 (correctly and incorrectly classified) cytosolic sequences. This corresponds to the precision to a precision of 0.58 as also shown in Table 6.

The confusion matrix confirms that the model is most certain about secreted and mitochondrial proteins, while it confuses cytosolic and nuclear proteins more often.

5.4 Prediction

Before making predictions on the test data, the final model is retrained using all five sub folds of the training data (i.e. all labelled data available) for 14 epochs. This corresponds to the average number of epochs after which each of the sub-models were trained. The model predicts a probability distribution over the four classes for each sequence in the test set. These are shown, together with the predicted class corresponding to the highest probability, in Table 8.

The model predicts mostly cytosolic and nuclear, and only one mitochondrial protein. This is sensible under the assumption that the compositions of proteins in the train and test sets are similar (i.e. there are mostly cytosolic and nuclear, and only few mitochondrial proteins).

Another observation is that predictions of secreted and mitochondrial proteins are usually made with a high probability (>80%), but cytosolic and nuclear proteins are predicted with much less confidence. This is in line with the F1 scores shown in Table 6.

Table 8. Predicted location of 20 proteins in blind test set

	Pr(Cyto)	Pr(Secreted)	Pr(Mito)	Pr(Nucleus)	Prediction
SEQ677	22.7%	56.0%	3.1%	18.2%	Secreted
SEQ231	6.5%	84.0%	0.9%	8.6%	Secreted
SEQ871	0.5%	0.4%	98.7%	0.4%	Mito
SEQ388	48.7%	0.0%	0.4%	50.9%	Nucleus
SEQ122	52.8%	0.5%	3.1%	43.5%	Cyto
SEQ758	15.5%	0.1%	0.2%	84.2%	Nucleus
SEQ333	52.2%	1.0%	6.7%	40.1%	Cyto
SEQ937	72.7%	0.2%	0.9%	26.1%	Cyto
SEQ351	62.0%	0.4%	4.2%	33.4%	Cyto
SEQ202	41.7%	4.9%	23.4%	29.9%	Cyto
SEQ608	37.0%	7.3%	20.7%	35.0%	Cyto
SEQ402	44.5%	0.3%	1.0%	54.2%	Nucleus
SEQ433	1.1%	98.2%	0.1%	0.6%	Secreted
SEQ821	5.2%	88.3%	0.1%	6.4%	Secreted
SEQ322	11.0%	0.0%	0.0%	88.9%	Nucleus
SEQ982	3.6%	0.0%	0.0%	96.4%	Nucleus
SEQ951	79.2%	0.2%	0.3%	20.4%	Cyto
SEQ173	55.3%	0.2%	0.7%	43.8%	Cyto
SEQ862	34.9%	0.2%	18.0%	46.9%	Nucleus
SEQ224	50.3%	0.5%	6.4%	42.8%	Cyto

5.5 Nearest neighbour amino acids

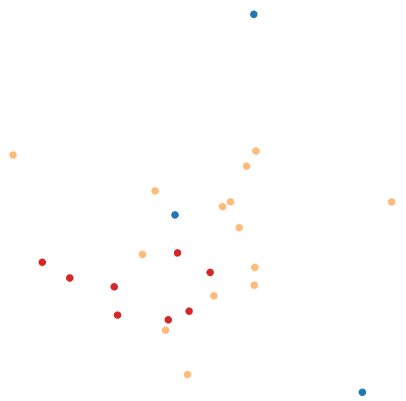
As a by-product of training a GRU classifier, I obtain an embedding matrix which contains 32-dimensional representations of amino acids.

I use the framework TensorFlow to visualise the embeddings as shown in Figure 3. I reduce the dimensionality of the embedding vectors to two and colour hydrophobic amino acids yellow, and hydrophilic amino acids red.[1] [2] It can be observed that hydrophilic amino acids cluster together in the bottom left corner, indicating some degree of similarity between them.

[1] There are three blue scatters for the one-letter code X, as well as an OOV and a PAD symbol, which are needed for technical reasons, but do not have any meaning beyond that.

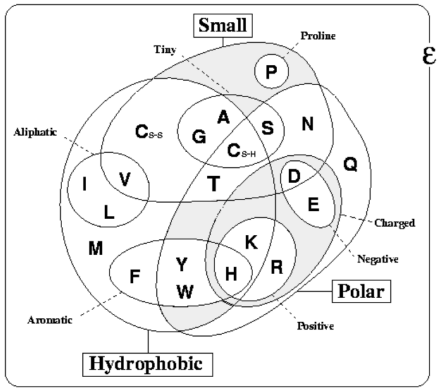
[2] Hydrophobicity as per Taylor's venn diagram

Fig. 3. PCA of trained amino acid embeddings



It is also possible to use Tensorboard to calculate the euclidian distances between the amino acid embedding vectors and to determine their nearest neighbours. This can then be compared to other similarity metrics between amino acids such as Taylor's venn diagram shown in Figure 4. [3]

Fig. 4. Taylor's venn diagram

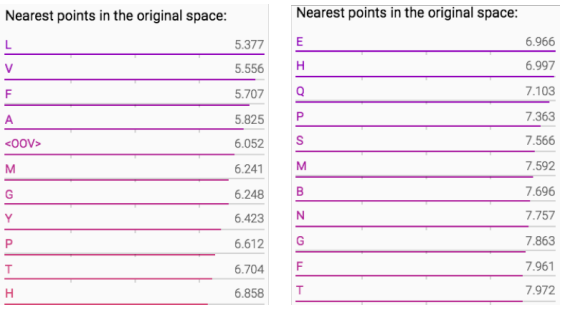


While the quality of embeddings vectors could potentially be improved with more data or a more compelx model, an example for when this works well is Isoleucine. In Figure 5 below I show the 10 nearest neighbours of Isoleucine (disregarding the OOV token). As can be seen, the amino acids closest to Isoleucine are the other two aliphatic amino acids Valine and Leucine. The amino acids that follow are all hydrophobic apart from Proline. This resembles the similarity of Isoleucine to neighbouring amino acids in Taylor's venn diagram.

Similarly, I show in Figure 5 the 10 nearest neighbours of Aspartic Acid. Among the closest neighbours there are Glutamic Acid, and other charged (e.g. Histidine) and hydrophilic (e.g. Proline) amino acids.

[3] adopted from Livingstone and Barton, 1993

Fig. 5. 10 nearest neighbours of Isoleucine (left) and Aspartic Acid (right)



6 Discussion

The main goal of this paper was to find a neural network structure that is able to predict the subcellular location of proteins with good accuracy, and whose training is attainable using limited computational resources. While the resulting accuracy on the validation set is in line with previous research (for example Reinhardt and Hubbard, 1998), there are several ways in which the results of this paper could be improved if more computational resources were available. These include:

- Using LSTM instead of GRU cells (see Sonderby *et al*, 2015): As mentioned in Section 4, LSTM cells have a more complex structure, which also make them computationally more expensive to train. While it is not the case that LSTM cells perform better in general, which cell performs better has to be assessed on a case by case basis;
- Using bidirectional RNN cells: In this paper uni-directional GRU cells were used. While GRU and LSTM cells in general help against the problem of vanishing gradients, bidirectional cells could lead to further improvement. This comes at a significant computational cost;
- Using attention (Bahdanau *et al*, 2014): Attention is a mechanism that can help to further prevent the problem of diminishing gradients, by explicitly enabling the model to review inputs from previous time steps. As amino acid sequences can be potentially very long, this may help to increase performance;
- Using full sequences: As discussed in Section 4, for computational reasons, and because most of the necessary information for location classification is contained in the beginning and end of any sequence, proteins have been truncated. While this does not seem to have affected the prediction accuracy of long sequences markedly, to get the highest performance, it may be helpful to use the full sequences, subject to computational constraints;
- Fusion: An attempt was made to use a model that does not require any feature engineering. However, the performance of neural networks can be improved by merging different types of data at various stages in the model. An example of early fusion would be if information other than the raw amino acid was fed into the RNN at any time stage. An example of late fusion would be if just before predicting the location of the protein, the vector representation of the amino acid (as given by the last memory state) was combined with the vector representation of another source of information. The types of data that could be made use of in such a way include properties of amino acids, such as their mass or their hydrophobicity;
- Ensemble methods: In practice, an ensemble of classifiers often achieved better prediction performance than any individual classifier. Examples in the literature include Park and Kanehisa, 2003, who use an ensemble of SVMs.

7 Conclusion

In this paper I have shown that RNNs, and in particular GRUs, can be used to predict the subcellular location of eukaryotic proteins. The approach was to use a relatively simple GRU model with one layer of 64 units, and embedded amino acid sequences of dimension 32. Training this model is computationally light and yet achieves a mean cross-validated accuracy of 69.58% with a 95% confidence interval of [68.36; 70.80].

The model adopted in this paper achieves similar prediction accuracy for short and long proteins (less than, or more than 2,000 amino acids respectively). There is however a clear difference in prediction accuracy for certain types of proteins. In particular, the model performs better at predicting the location of secreted and mitochondrial proteins than for cytosolic and nuclear proteins. This suggests that there are more distinct patterns in the amino acid sequences of secreted and mitochondrial proteins.

As a by-product, the model yields amino acid embedding vectors. I have shown that when projected onto a low-dimensional space, hydrophobic and hydrophilic amino acids tend to cluster together. It is also possible to calculate the differences between the embedding vectors in the original space. Amino acids with similar properties are shown to be close together in terms of euclidian distance.

References

Bahdanau, D., Cho, K., Bengio, Y., (2014). Neural Machine Translation by jointly learning to align and translate.

Bergstra, J., Bengio, Y., (2012). Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research 2012*, **13**, 281-305.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y. (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation, *EMNLP 2014*,

Dong, C.-W., Feng, Z.-P., Yang, X.-G., Luo, R.-Y. (2003) Predicting subcellular locations of eukaryotic proteins based on stepwise discriminant analysis. *APBC '03 Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003*, **19**, 86-93.

Emanuelsson, O., Nielsen, H., Brunak, S., (2000) Predicting subcellular localization of proteins based on their N-terminal N-terminal amino acid sequence *Journal of Molecular Biology (2000)*, **300** 1005-1016.

Hochreiter, S., Schmidthuber, J., (1997) Long Short-Term Memory, *Neural Comput*, **9**, 1735-1780.

Ioffe, S., Szegedy, C.,(2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Livingstone, C. D., Barton, G. J. (1993) Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation *Bioinformatics 1993* **9(6)** 745-756

Matsuda, S., Vert, J.-P., Saigo, H., Ueda, N., Toh, H. and Akutsu, T. (2005) A novel representation of protein sequences for prediction of subcellular location using support vector machines *Protein Science*, **14**, 2804?2813.

Mer, A. S., Andrade-Navarro, M. A. (2013) A novel approach for protein subcellular location prediction using amino acid exposure. *BMC Bioinformatics 2013*, **14**, 342.

Murphy,R.F., Boland, M.V., Vellis, M. (2000) Towards A systematic for protein subcellular location: quantitative description of protein localization patterns and automated analysis of fluorescence microscope images. *Proc Int Conf Intell Syst Mol Biol*, **8**, 251-259.

Nair, V., Hinton, G. E., (2010). Rectified linear units improve restricted Boltzmann machines. *ICML 2010*, 807-814

Nakashima, H., Nishikawa, K., (2015) Discrimination of Intracellular and Extracellular Proteins Using Amino Acid Composition and Residue-pair Frequencies *Journal of Molecular Biology (1994)*, **238:1** 54-61.

Park, K.-J., Kanehisa, M. (2003). Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, **19**: 1656?1663.

Reinhardt, A., Hubbard, T., (1998). Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Res*, **26**: 2230?2236.

Sønderby, S. K., Sønderby, C. K., Nielsen, H. Winther, O., (2015) Convolutional LSTM Networks for Subcellular Localization of Proteins *Algorithms for Computational Biology 9199 (2015)*, **68**.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research 2014*, **15**, 1929-1958.

Annex

Fig. 6. Validation accuracy of 5 cross-validation folds

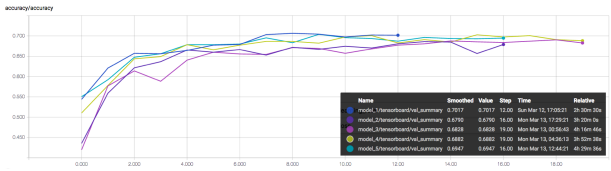


Fig. 7. ROC curve of 5 cross-validation folds (cyto)

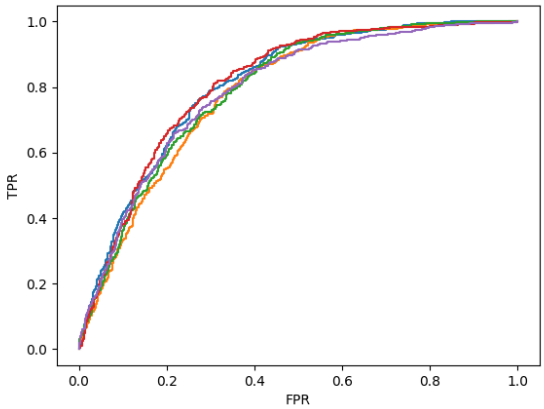


Fig. 8. ROC curve of 5 cross-validation folds (secreted)

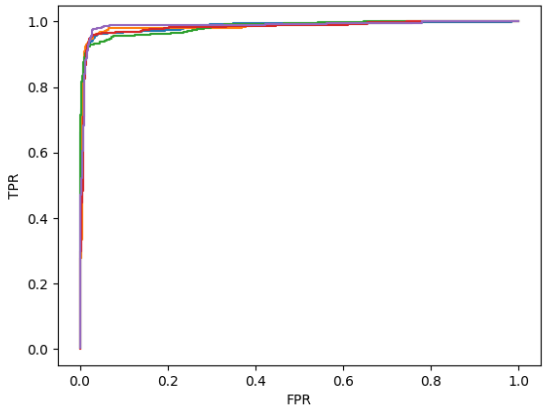


Fig. 9. ROC curve of 5 cross-validation folds (mito)

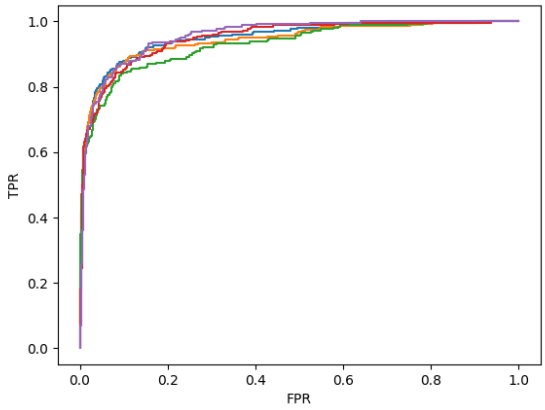


Fig. 10. ROC curve of 5 cross-validation folds (nucleus)

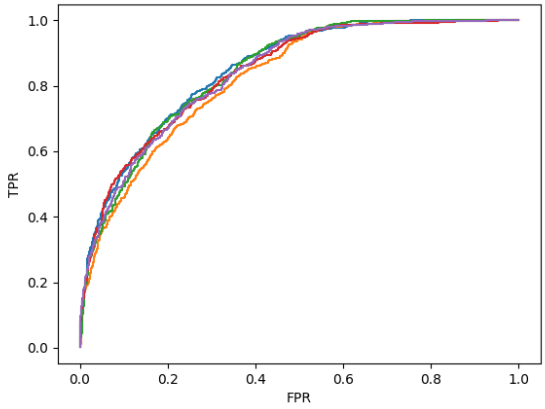


Table 9. Validation fold 1 - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[[0, 100)	9	3	3	6	21
[[100, 500)	267	172	120	274	833
[[500, 1000]	210	114	95	236	655
[[1000, 1500)	69	31	28	81	209
[[1500, 2000)	27	6	6	28	67
[[2000, inf]	18	12	4	25	59
[Total	600	338	256	650	1844

Table 10. Validation fold 2 - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[[0, 100)	90	48	54	106	298
[[100, 500)	277	134	113	304	828
[[500, 1000]	158	71	74	176	479
[[1000, 1500)	53	19	27	62	161
[[1500, 2000)	14	8	10	17	49
[[2000, inf]	7	8	5	9	29
[Total	599	288	283	674	1844

Table 11. Validation fold 3 - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[[0, 100)	120	78	57	122	377
[[100, 500)	329	200	146	413	1088
[[500, 1000]	109	47	45	125	326
[[1000, 1500)	3	5	4	15	27
[[1500, 2000)	7	2	1	3	13
[[2000, inf]	4	3	2	4	13
[Total	572	335	255	682	1844

Table 12. Validation fold 4 - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[[0, 100)	16	10	8	13	47
[[100, 500)	338	166	134	340	978
[[500, 1000]	195	118	83	195	591
[[1000, 1500)	51	25	16	51	143
[[1500, 2000)	13	12	10	14	49
[[2000, inf]	13	8	5	10	36
[Total	626	339	256	623	1844

Table 13. Validation fold 5 - breakdown by class and sequence length

Bin	Cyto	Secreted	Mito	Nucleus	Total
[[0, 100)	5	6	0	11	22
[[100, 500)	294	149	137	341	921
[[500, 1000]	233	93	78	232	636
[[1000, 1500)	50	36	22	70	178
[[1500, 2000)	14	16	7	20	57
[[2000, inf]	10	5	5	10	30
[Total	606	305	249	684	1844