An Analysis of the Prediction for Protein Subcellular Localization

Group 7
Wei Meng, Yaoxiang Li, Zewei Xiong

ABSTRACT

Background: As a result of large-scale genome sequencing efforts in recent years, protein data has accumulated in public data banks at an increasing rate. Subcellular localization is a crucial functional attribute of a protein [1]. Since cellular functions are often localized in specific compartments, predicting the subcellular localization of unknown proteins may be used to obtain useful information about their functions and studying the subcellular localization of proteins is also helpful in understanding disease mechanisms and for developing novel drugs. So the aim of this study is: 1) perform necessary visualization of 11 types of proteins for preliminary analysis; 2) show that the LSTM, Bi-directional LSTM, and Gated recurrent units models are efficacy for predicting subcellular localization of proteins from the sequence; 3) explore the best hyperparameters practice for those models we proposed.

Methods: The proteomics dataset was summarized by the SWISS-PROT database [4], and the dataset contains 5959 proteins annotated to one of 11 different subcellular locations which are: chloroplast, cytoplasm, endoplasmic reticulum, extracellular space, Golgi apparatus, lysosomal, mitochondrion, nucleus, peroxisome, plasma membrane and vacuole proteins. To pre-process the data, we first removed N- and C- terminal regions from those gene sequences. Moreover, those pre-processed sequences were truncated into length 401 for comparability and reduction of computational complexity. We performed these three learning models concerning binary classification, three categories classification and four categories classification to see the performance of these models. The selection of categories was based on the frequencies of some sequences of each protein. We considered 80% and 20% split for the dataset and 100-fold cross-validation in our study. The evaluation index for assessing the performance of the model was accuracy, precision, recall and F1 score in this study.

Results: The proteins visualization showed the distribution of amino acid for 11 types of the protein sequence. From the visualization plots for each type of protein, we can find the different distribution pattern of the amino acid of different protein which can be a rationale to explain whether it is reasonable to make the prediction of subcellular localization between different particular proteins. For the three types of classification (binary, three classes, four classes), in general, the performance of three models was pretty well. For binary classification, the accuracy, precision, recall and F1 score were around 0.9, and the highest value can be reached 0.986. GRU and Bi-LSTM performed a little bit better than LSTM in this case. For three and four categories classification, LSTM and Bi-LSTM model showed better performance.

Conclusion: Based on the evaluation indexes that we obtained in results part, we can demonstrate that GRU model, LSTM model, and bi-directional LSTM model are practical approaches to perform the prediction of the subcellular localization of proteins for binary classification, three categories classification and four categories classification.

1. INTRODUCTION

As a result of large-scale genome sequencing efforts in recent years, protein data has accumulated in public data banks at an increasing rate. Subcellular localization is a crucial functional attribute of a protein [1]. Since cellular functions are often localized in specific compartments, predicting the subcellular localization of unknown proteins may be used to obtain useful information about their functions and to select proteins for further study. Moreover, studying the subcellular localization of proteins is also helpful in understanding disease mechanisms and for developing novel drugs. Analyzing protein data to extract useful knowledge is thus essential for projects like automatic annotation. It is desirable to have an automated and reliable system for predicting subcellular localization of proteins from amino acid sequences [1].

Knowledge of the subcellular localization of a protein can significantly improve the target identification during the drug discovery process. For example, secreted proteins and plasma membrane proteins are easily accessible by drug molecules due to their localization in the extracellular space or on the cell surface. Bacterial cell surface and secreted proteins are also of interest for their potential as vaccine candidates or as diagnostic targets. Aberrant subcellular localization of proteins has been observed in the cells of several diseases, such as cancer and Alzheimer's disease. Secreted proteins from some archaea that can survive in unusual environments have industrially essential applications. By using prediction, a high number of proteins can be assessed to find candidates that are trafficked to the desired location [2].

Protein function prediction, also known as protein sorting or subcellular localization, has attracted significant interest in the bioinformatics field [3]. Moreover, it is usually considered as a multi-label classification problem. Different computation methods were proposed and attempted in previous researchers in the last few decades for this problem [6-12].

Currently, there are several popular methods for extracting useful information from primary sequences and infer functional information based on a comparison of new sequences to existing sequences of known function. Among them, Basic Local Alignment Search Tool (BLAST) is the first and the most widely used one [13]. This method gathers the homologous proteins' function information by searching the query sequence against the existing protein databases as it contains experimentally determined protein function information. For instance, the methods of Gotcha [14], OntoBlast [15], and Goblet [16]. Besides BLAST method, others use the tool PSI-BLAST [13] to find the remote homologous, such as the PFP method [17].

Another kind of wild-applied methods is network-based methods. These methods are under the assumption that interacted proteins share similar functions [10, 18-23]. Protein-protein interaction networks construct protein function prediction. An alternative network such as gene-gene interaction network and domain co-occurrence network is also used by researchers [10, 24].

Though the methods mentioned above have been proved successful in protein predicting field, this paper used the long short-term memory network (LSTM), the state-of-the-art machine learning method—deep learning which uses multiple layers' representation and abstraction of data to analyze biological sequences and predicts to which subcellular compartment a protein belongs. The performance would be compared to several LSTM models and other baseline models by using only the protein sequence information. So the aim of this study is:

1. Perform essential visualization of 11 types of proteins for preliminary analysis;
2. To show that the LSTM, Bi-directional LSTM (Bi-LSTM) and Gated recurrent units (GRU) models are efficacy for predicting the subcellular localization of proteins from sequences;
3. Hyperparameters in deep learning are many, tuning them will take weeks or months. Generally speaking, researchers do this tuning and publish a paper when they find a beautiful set of architecture which performs better than other. In this process, we are trying to find the best hyperparameters practice for those models we proposed. However, due to the limitation of time, it may not be the best practice.

2. METHODS

2.1 Data description

The proteomics dataset was summarized by the SWISS-PROT [4] database release 42 (2003–2004) by which obtained extracting all animal, fungal and plant protein sequences. A data pre-processing process such as homology-reduced was completed by Hoglund et al. [5] to train the MultiLoc algorithm they published. The dataset contains 5959 proteins annotated to one of 11 different subcellular locations which are: chloroplast, cytoplasm, endoplasmic reticulum, extracellular space, Golgi apparatus, lysosomal, mitochondrion, nucleus, peroxisome, plasma membrane and vacuole which represented proteins of plants cell and fungal cell while animal cells shared all localizations with them, but have lysosomes instead of vacuoles. The only variable we intend to consider is protein sequence. Table 1 below is the description of data.

Table 1. Description of dataset

Protein Type	Number of Sequences	Protein Type	Number of Sequences
chloroplast	449	mitochondrial	510
cytoplasmic	1411	nuclear proteins	837
ER proteins	198	peroxisomal	157
extracellular proteins	843	plasma membrane	1238
Golgi apparatus	150	vacuolar proteins	63
lysosomal	103	**Total**	**5959**

2.2 Data pre-process

The protein data was encoded using 1 of K encoding, and as a collective sense that the N- and C-terminal regions are known to contain sorting signals. To remove this sorting signals, we merely removed those terminal regions from the gene sequences. Moreover, we truncated those pre-processed sequences into length 401 for comparability and reduction of computational complexity.

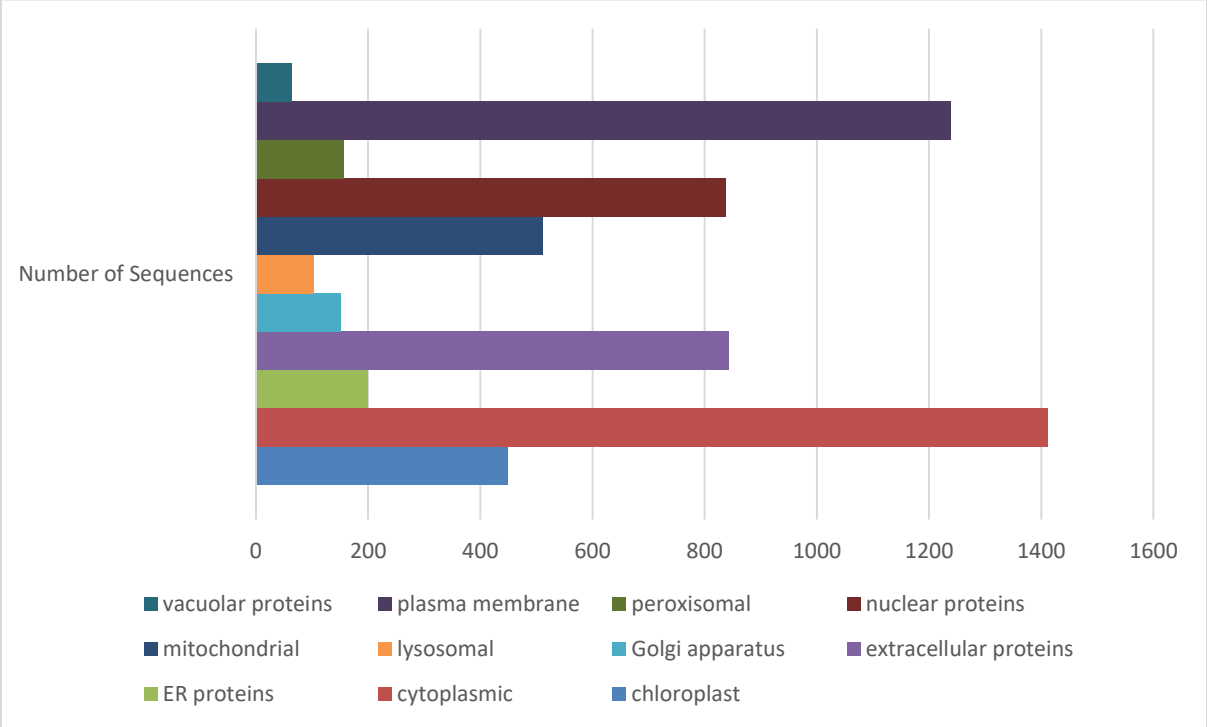

Figure 1. Frequency table of number of sequences of each protein

Figure 1 shows the frequency table of some sequences of each type of proteins. From the frequency table, we can find that the distribution of these classes is not uniform and some of them even less than 200, e.g., vacuolar, peroxisomal, lysosomal, Golgi apparatus and so on. As we know that a small dataset will not get a good result for deep neural networks since its architecture decides that model often have more than millions of parameters to estimate and it is tough to converge on a tiny dataset such as less than 1000 examples per class. If we do not do augmentation of the train data and do not implement batch normalization, we have to reduce the question into a smaller one.

So firstly we reduced this problem into a binary classification problem and using some of the standard subcellular localization classes, including chloroplast with plasma_membrane, chloroplast with cytoplasmic, chloroplast with nuclear, ER with plasma_membrane, ER with

cytoplasmic and ER with cytoplasmic. Then we step further to consider three and four classes classification. For the three class multi-label classification scenario, we choose the three subcellular localization classes with the top 3 frequencies which are cytoplasmic, nuclear and plasma membrane. For the four label classification problem, we use the same criterion to choose the classes with top 4 frequencies which are the above mentioned three classes plus extracellular. We considered 80% and 20% split for the dataset and 100-fold cross-validation in our study.

2.3 Evaluation index

For binary classification, we considered the accuracy, precision, recall and F1 score as the evaluation indexes to compare the performance of three models in our study. Then for three categories and four categories classification, we considered the precision, recall and F1 score as the evaluation indexes to compare the performance of these models.

In a classification task, the precision for a class is the number of real positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of actual positives divided by the total number of elements that belong to the positive class (i.e., the sum of true positives and false negatives, which are items which were not labeled as belonging to the real class but should have been). The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system.

Given positive/negative rates for each class k, the resulting score is computed this way:

$$\text{Precison}_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FP_k}$$

$$\text{Recall}_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FN_k}$$

$$\text{MeanFScore} = \text{F1score} = \frac{2\text{Precision}_{micro}\text{Recall}_{micro}}{\text{Precision}_{micro} + \text{Recall}_{micro}}$$

2.4 Experimental setup

2.4.1 Loss function

The commonly known softmax function is:

$$\sigma(y_i) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

Which can take a vector of real values and transform it into a vector of values between 0 and one that sums to one. Moreover, we choose the cross-entropy loss with softmax function:

$$Loss_i = -\log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}) = -f_{y_i} + \log\sum_j e^{f_j}$$

2.4.2 Optimization method and learning rate

We implemented the Adam algorithm for parameter estimation, and the best practice learning rate for this dataset was set up as learning rate=0.1, beta1=0.9, beta2=0.999, epsilon=1e-08.

2.4.3 Size of RNN hidden state

In the Long short-term memory unit and GRU recurrent network cell, we set up the size of RNN hidden state to 2000. Moreover, the number of layers in the RNN was 4.

2.4.4 Dropout design and probability

To reduce over-fitting, we applied dropout before the readout layer with the probability of 0.6. The initial weights were sampled uniformly distributed from the interval [-0.2, 0.2].

2.4.5 Environment

All of the training and testing were doing on a local workstation, with following adequately configured: NVIDIA Graphics Drivers v375.66, NVIDIA CUDA Toolkit v8.0, NVIDIA cuDNN v5.1, Python3 v3.5.2, Tensorflow v1.4, NumPy v1.13.3, pandas v0.20.3, Matplotlib v2.1.0.

2.5 Gated recurrent units

A gated recurrent unit was proposed by Cho et al. [25] to make each recurrent unit to capture dependencies of different time scales adaptively. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having separate memory cells.

The activation h_t^j of the GRU at time t is a linear interpretation of the previous activation h_{t-1}^j and the candidate activation $\hat{h}_t^j$.

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j$$

Where an update gate z_t^j decides how much the unit updates its activation or content. The update gate is computed by

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j$$

This procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit. The GRU, however, does not have any mechanism to control the degree to which its state is exposed but exposes the whole state each time.

The candidate activation $\tilde{h}_t^j$ is computed similarly to that of the traditional recurrent unit

$$\tilde{h}_t^j = \tanh(Wx_t + U(r_t \otimes h_{t-1}))^j$$

Where is a set of reset gates and $\otimes$ is an element-wise multiplication? When off (r_t^j close to 0), the reset gate efficiently makes the unit act as if it is reading the first symbol of an input sequence, allowing it to forget the previously computed state.

The reset gate r_t^j is computed similarly to the update gate

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$$

See Figure 1 for the graphical illustration of the GRU [26].

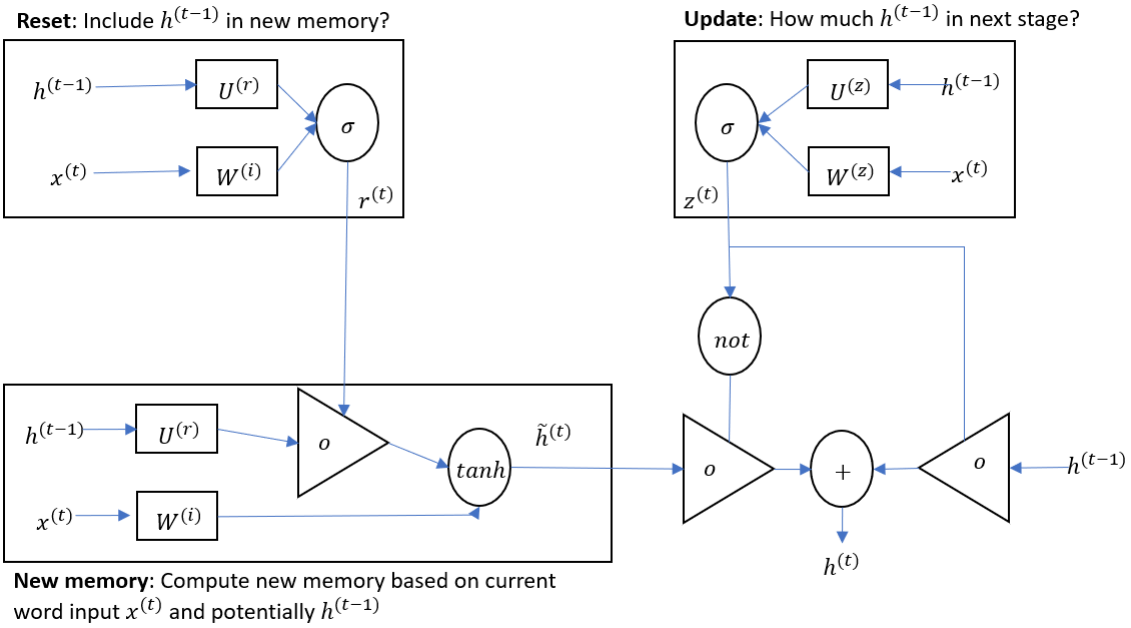

Figure 2. The detailed internals of a GRU network

2.6 Long short-term memory network

The Long Short-Term Memory unit was initially proposed by Hochreiter and Schmidhuber in 1997 [27]. Since then, some minor modifications to the original LSTM unit have been made. We follow the implementation of LSTM as used in Graves [28].

Unlike to the recurrent unit which computes merely a weighted sum of the input signal and applies a nonlinear function, each *j*-th LSTM unit maintains a memory c_t^j at time t. The output h_t^j, or the activation, of the LSTM unit, is then

$$h_t^j = o_t^j \tanh(c_t^j)$$

Where is an output gate that modulates the amount of memory content exposure? The output gate is computed by

$$\sigma_t^j = \sigma(W_\sigma x_t + U_\sigma h_{t-1} + V_o c_t)^j$$

Where is a logistic sigmoid function and V_o is a diagonal matrix?

The memory cell c_t^j is updated by partially forgetting the real memory and adding a new memory content $\tilde{c}_t^j$:

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j$$

where the new memory content is

$$\tilde{c}_t^j = \tanh(W_c x_t + U_c h_{t-1})^j$$

The extent to which the real memory is forgotten is modulated by a forget gate f_t^j, and the degree to which the new memory content is added to the memory cell is modulated by an input gate i_t^j. Gates are computed by

$$f_t^j = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j$$

$$i_t^j = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j$$

Note that V_f and V_i are diagonal matrices.

Unlike to the traditional recurrent unit which overwrites its content at each time-step, an LSTM unit can decide whether to keep the current memory via the introduced gates. Intuitively, if the LSTM unit detects an essential feature from an input sequence at an early stage, it comfortably carries this information (the existence of the feature) over a long distance, hence, capturing potential long-distance dependencies [29]. See Figure 2 for the graphical illustration of the LSTM [26].

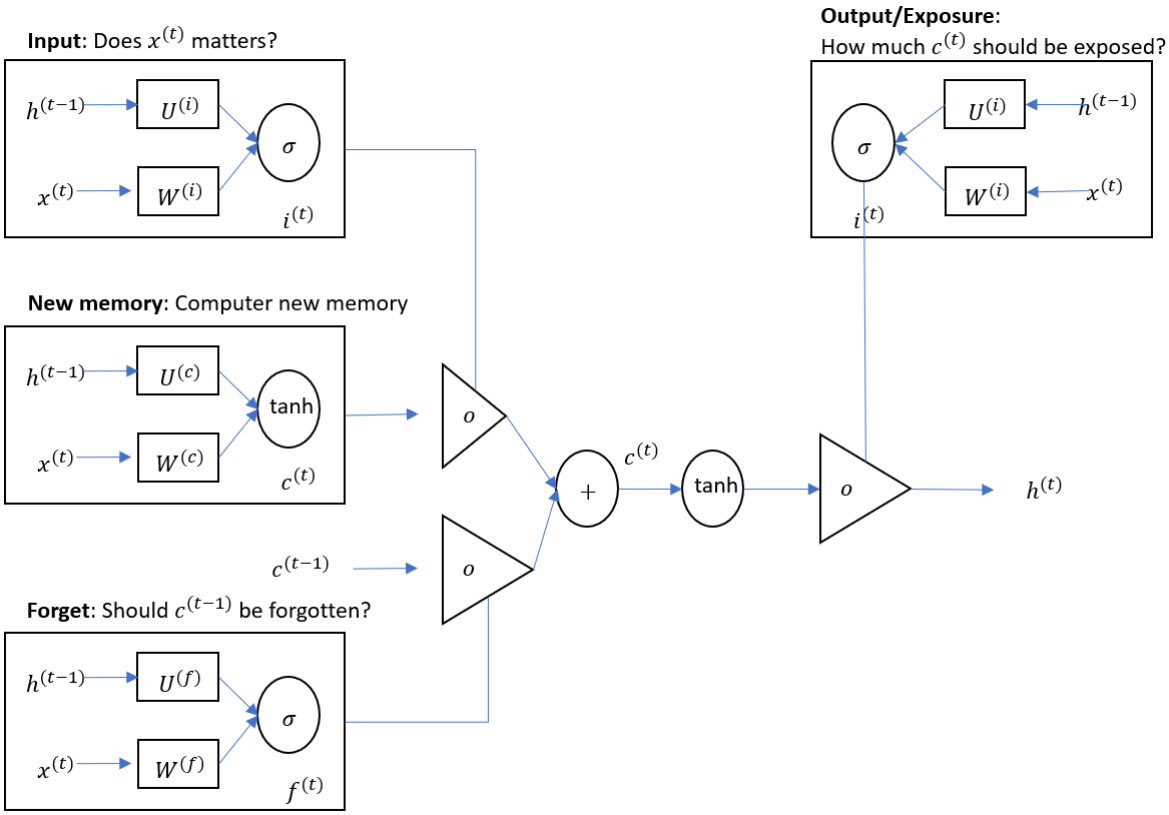

Figure 3. The detailed internals of an LSTM network

2.7 Bi-directional extended short-term memory network

Bi-directional LSTM networks extend the unidirectional LSTM networks by introducing a second layer, where the hidden to hidden connections flow in opposite temporal order. The model is, therefore, able to exploit information both from the past and the future [30]. These two sub-layers compute forward and backward hidden sequences $\vec{h}$, $\overleftarrow{h}$ respectively, which are then combined to compute the output sequence y (see Figure 3), thus:

$$\vec{h}_t = H(W_{x\underset{h}{\rightarrow}} x_t + W_{\underset{h}{\rightarrow}\underset{h}{\rightarrow}} \vec{h}_{t-1} + b_{\underset{h}{\rightarrow}})$$

$$\overleftarrow{h}_t = H(W_{x\underset{h}{\leftarrow}} x_t + W_{\underset{h}{\leftarrow}\underset{h}{\leftarrow}} \vec{h}_{t+1} + b_{\underset{h}{\leftarrow}})$$

$$y_t = W_{\underset{h}{\rightarrow}\,y} \vec{h}_t + W_{\underset{h}{\leftarrow}\,y} \overleftarrow{h}_t + b_y)$$

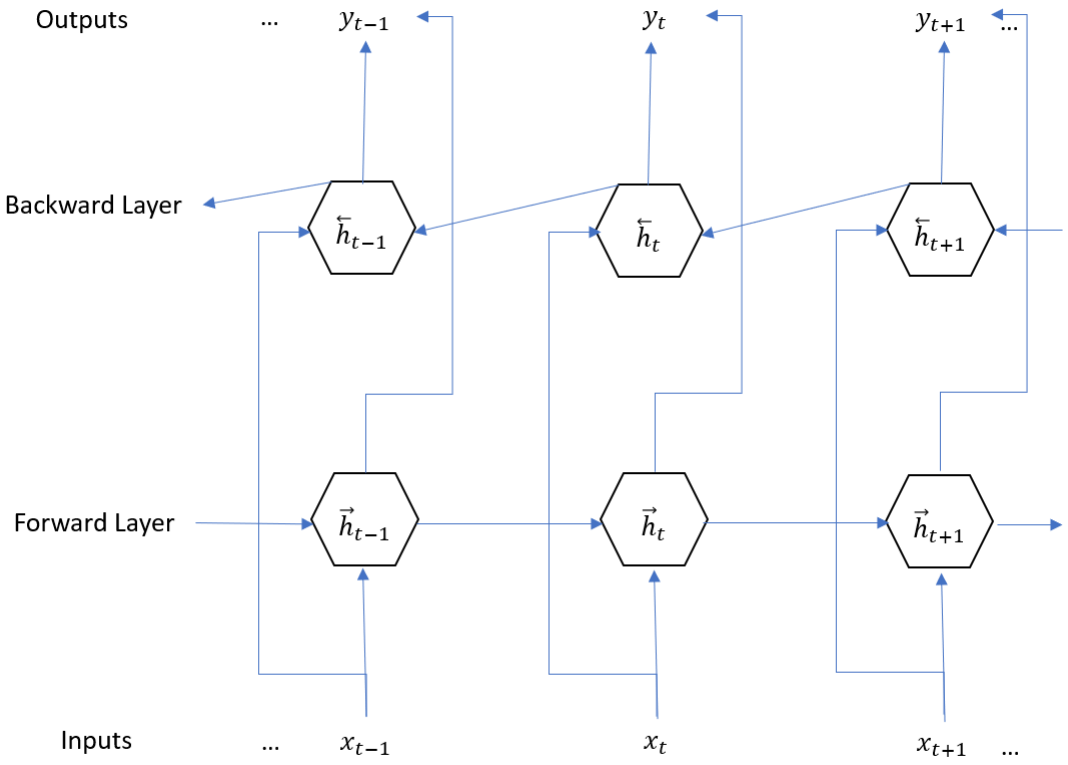

Figure 4. Architecture of a Bi-LSTM network

3. RESULTS

3.1 Proteins visualization

Figure 5 shows the distribution of amino acid for 11 types of the protein sequence. Figure 6-27 are plots which visualized the distribution of amino acid in each protein type separately. As it can be seen from each plot, the different color represented different amino acid and based on the degree of a particular color; we can know the different amino acid distribution pattern for a different protein. Moreover, it can be a rationale to explain whether it is reasonable to make the prediction of subcellular localization between different particular proteins. In general, the model will have a good performance when the distributions of an amino acid of each protein are substantially distinct. Take the cytoplasmic and ER proteins as an example, it can be clearly seen from Figure 8 and Figure 9 that the mainly distributed amino acid in cytoplasmic is represented by color ‘yellow’ and the mainly distributed amino acid in ER proteins is represented by color ‘pink’ (see Figure 10 and Figure 11) which means that the biology process implemented by this two proteins are distinctly different so that it can be an excellent choice to perform classification. The distribution of amino acid in other proteins can be explained in the same way.

Figure 5. All proteins categories

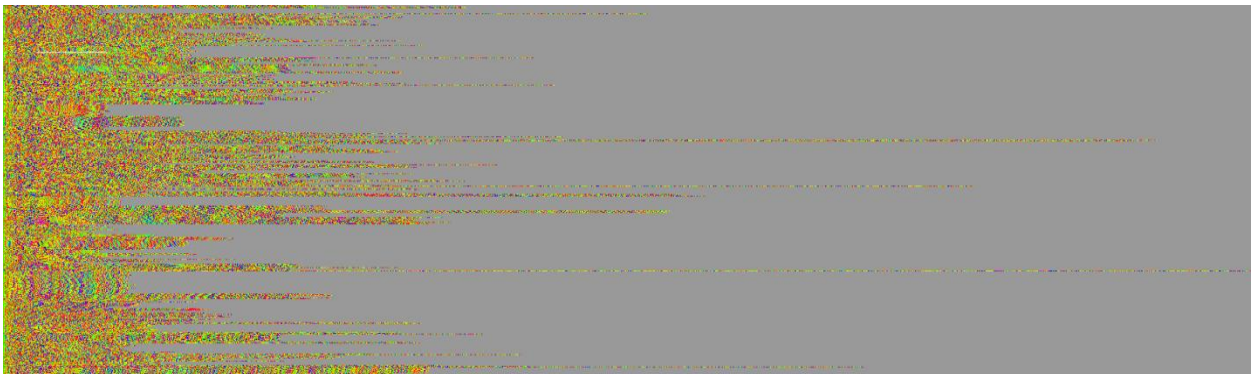

Figure 6. chloroplast

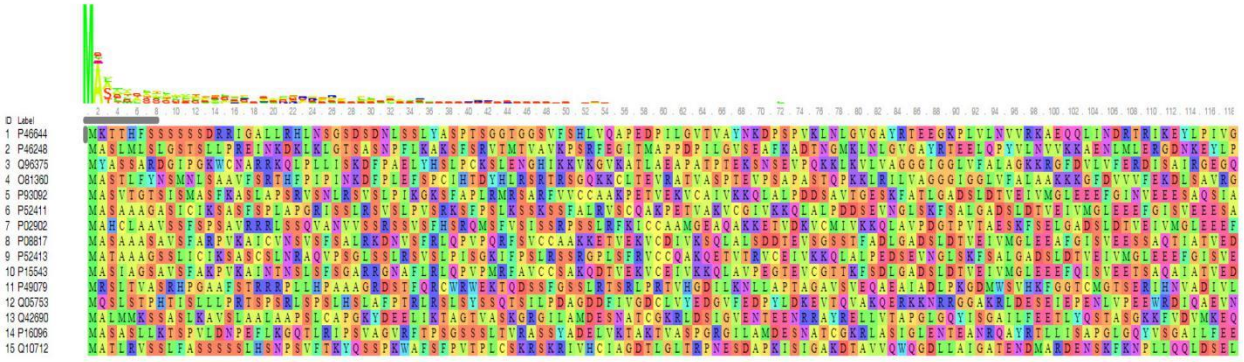

Figure 7. chloroplast

Figure 8. cytoplasmic

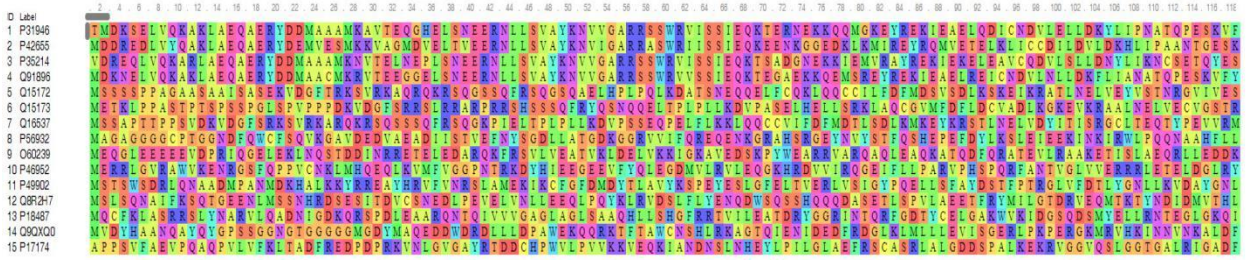

Figure 9. cytoplasmic

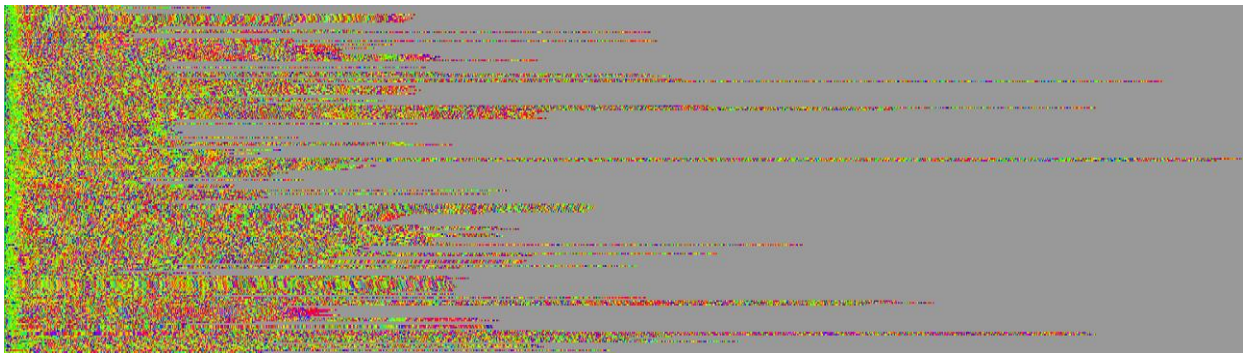

Figure 10. ER

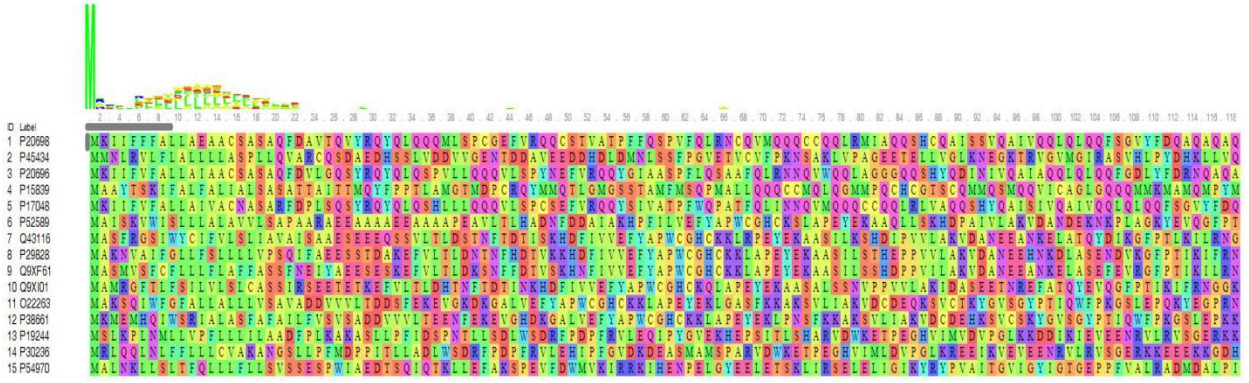

Figure 11. ER

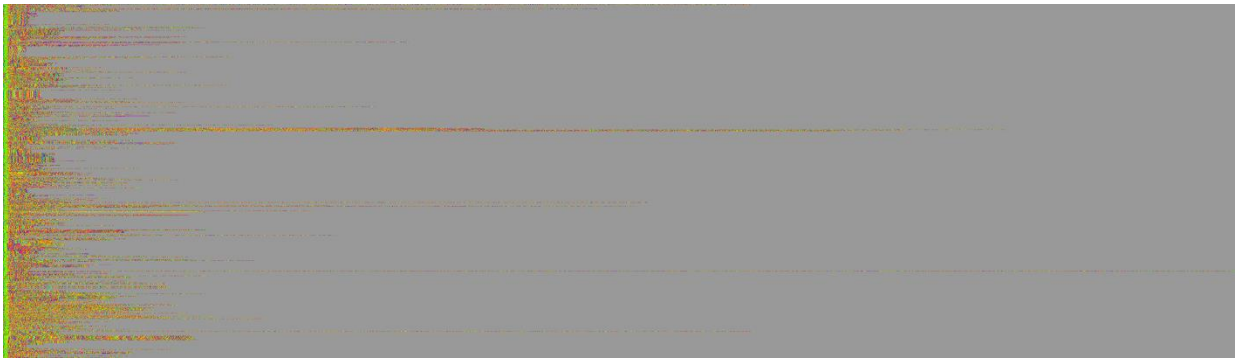

Figure 12. extracellular

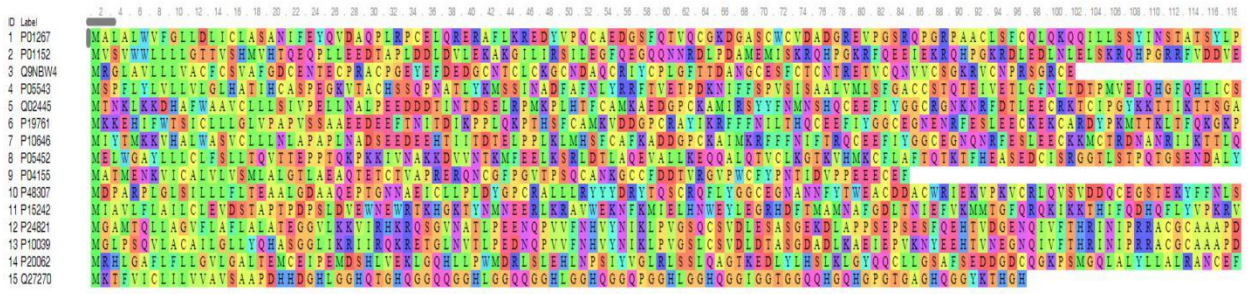

Figure 13. extracellular

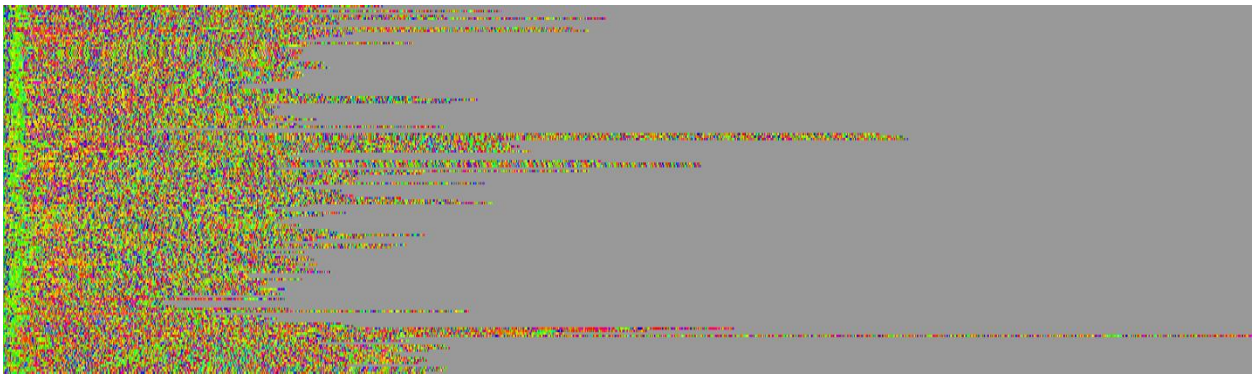

Figure 14. Golgi

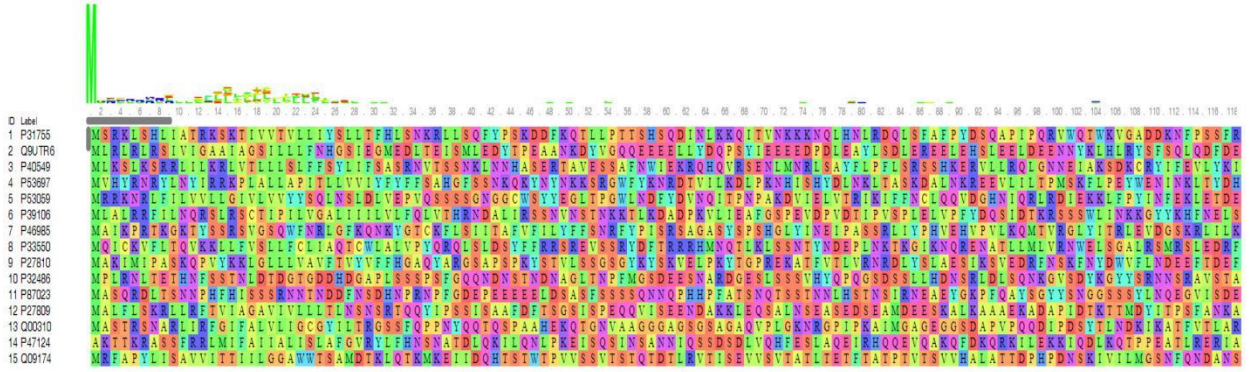

Figure 15. Golgi

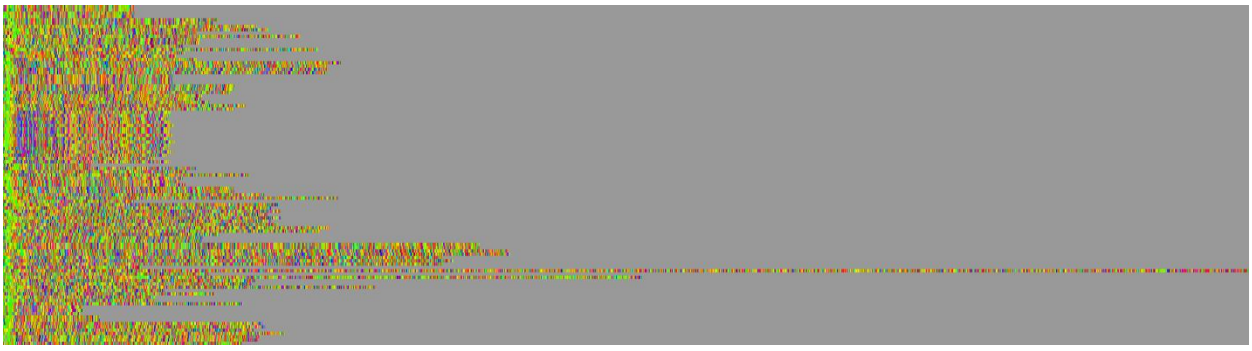

Figure 16. lysosomal

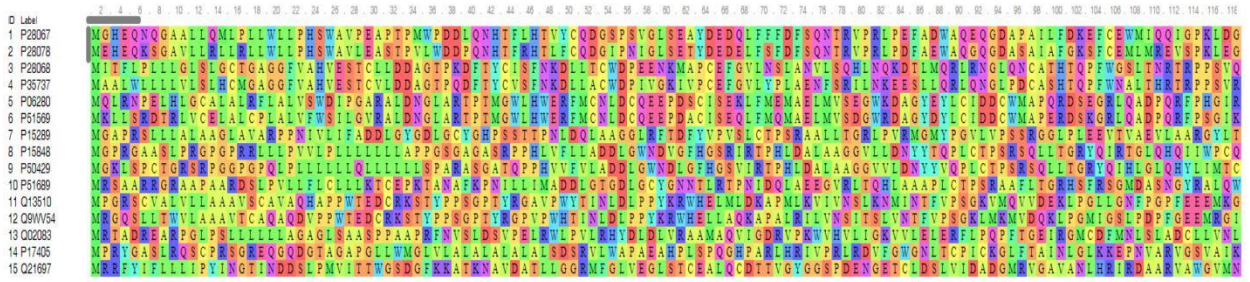

Figure 17. lysosomal

Figure 18. mitochondrial

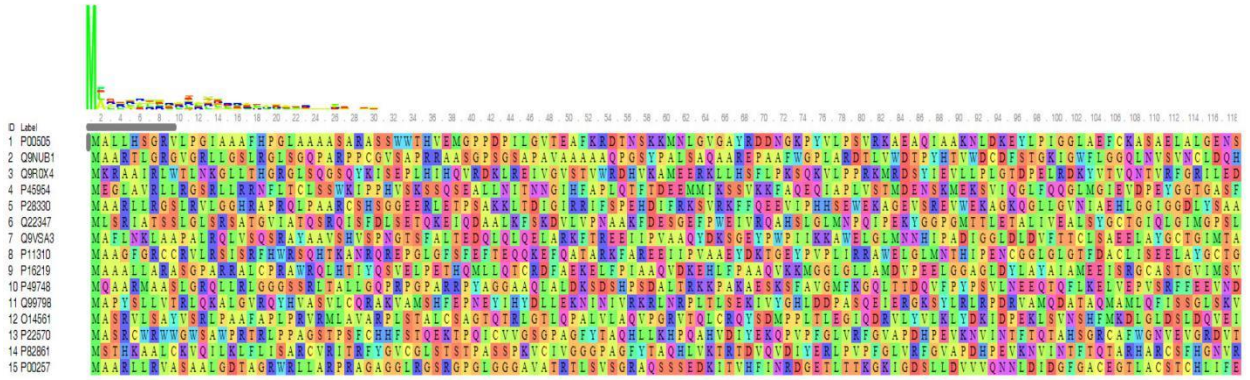

Figure 19. mitochondrial

Figure 20. nucleus

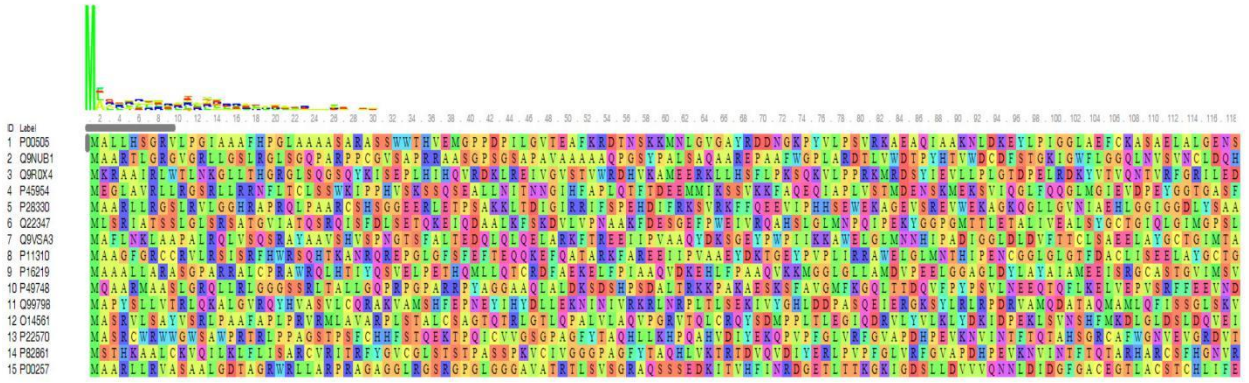

Figure 21. nucleus

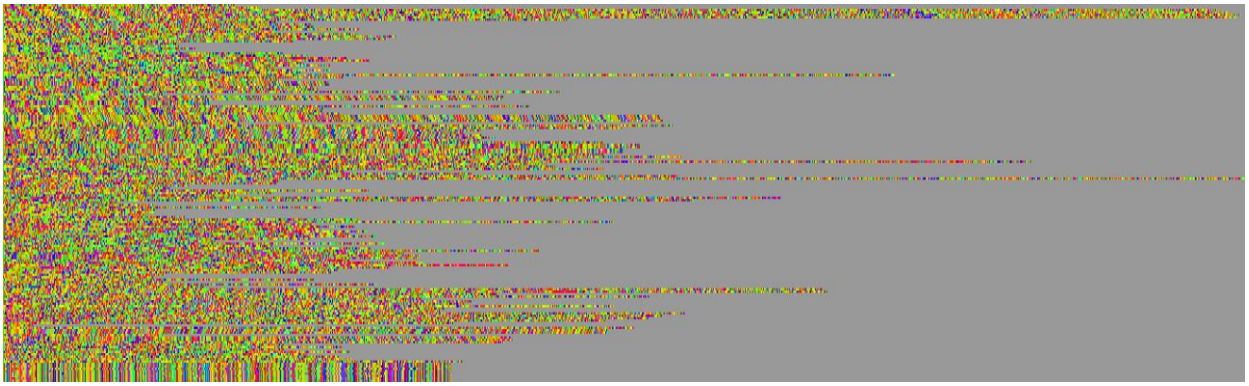

Figure 22. peroxisomal

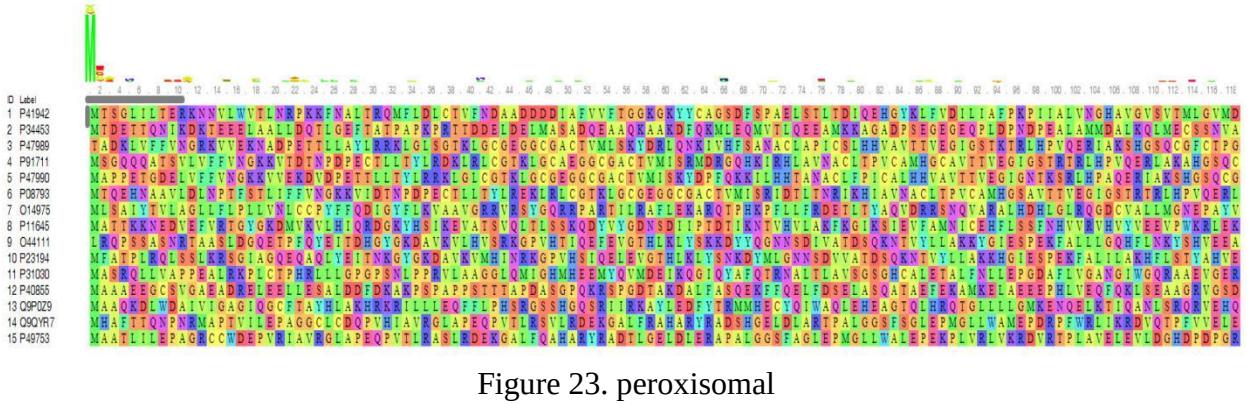

Figure 23. peroxisomal

Figure 24. plasma membrane

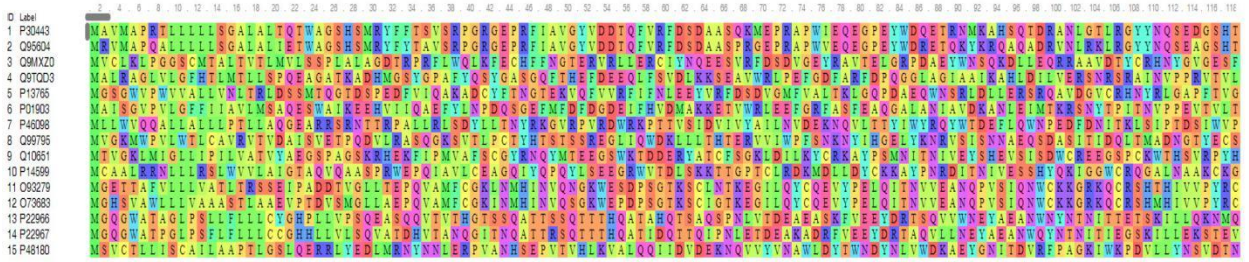

Figure 25. plasma membrane

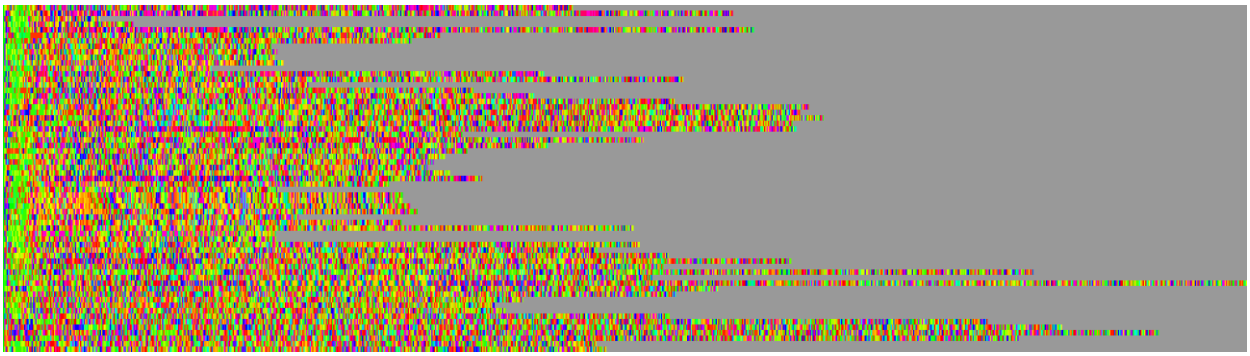

Figure 26. vacuolar

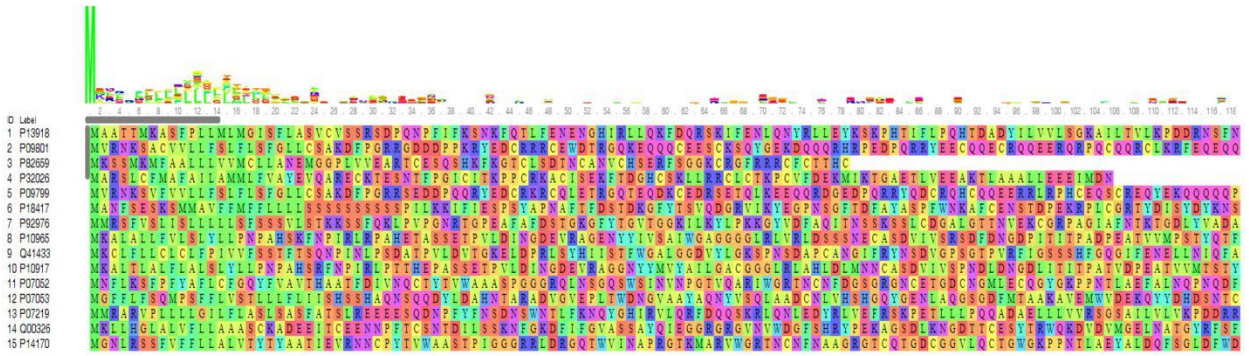

Figure 27. vacuolar

3.2 Performance of three deep learning models

In this study, we considered three cases: binary classification, three classifications, and four classifications by using GRU, basic LSTM and Bi-LSTM models to predict the subcellular localization of the targeted proteins. Table 2-4 shows the performance of each model.

For the binary classification (see Table 2), in general, all models performed well in this case as the high accuracy, precision, recall and F1 score values which were around 0.9 and the highest value can be reached was 0.986. To step further to compare these three methods, we found that

when predicting the subcellular localization of proteins between chloroplast with the plasma membrane and chloroplast with cytoplasmic, the bi-directional LSTM model performed better than the other two models. However, when it came to the prediction of subcellular localization between chloroplast with nuclear, ER with the plasma membrane, ER with cytoplasmic and ER with nuclear, GRU model showed better performance than the LSTM and Bi-LSTM model based on the values of accuracy, precision, recall and F1 score.

Table 2. Comparison the performance of for LSTM model, Bi-LSTM model and GRU model for two subcellular localization categories outcome

	Binary classification	Accuracy	Precision	Recall	F1 Score
LSTM	chloroplast & plasma	0.899	0.818	0.744	0.780
Bi-LSTM	membrane	**0.985**	**0.958**	**0.933**	**0.945**
GRU		0.896	0.818	0.900	0.857
LSTM	chloroplast & cytoplasmic	0.950	0.953	0.976	0.964
Bi-LSTM		**0.967**	**0.973**	**0.986**	**0.979**
GRU		0.912	0.945	0.841	0.890
LSTM	chloroplast & nuclear	0.856	**0.964**	0.769	0.856
Bi-LSTM		0.928	0.931	0.909	0.920
GRU		**0.943**	0.931	**0.929**	**0.930**
LSTM	ER & plasma membrane	0.768	0.748	0.777	0.762
Bi-LSTM		0.882	0.899	**0.865**	0.882
GRU		**0.916**	**0.951**	0.858	**0.902**
LSTM	ER & cytoplasmic	0.922	0.959	0.878	0.917
Bi-LSTM		0.923	0.938	0.882	0.909
GRU		**0.957**	**0.979**	**0.959**	**0.969**
LSTM	ER & nuclear	0.910	0.979	0.909	0.943
Bi-LSTM		0.907	0.952	0.866	0.907
GRU		**0.946**	**0.979**	**0.936**	**0.957**

For the three categories classification (see Table 3), in general, similar as the results for prediction of binary subcellular localization of proteins, all models performed well as the high accuracy, precision, recall and F1 score values which were around 0.8. To step further to compare these three methods, we found that the when predict the subcellular localization of proteins among cytoplasm, nucleus and plasma membrane proteins, the LSTM model performed better than the other two models. Moreover, for the four categories classification (see Table 4), in

general, all models performed well as the high accuracy, precision, recall and F1 score values which were around 0.9. Moreover, when compared these three methods, it can be concluded that the when predict the subcellular localization of proteins among cytoplasm, nucleus, plasma membrane and extracellular proteins, the Bi-LSTM model performed better than the other two models under this circumstance.

Table 3. Comparison the performance of for LSTM model, Bi-LSTM model and GRU model for three subcellular localization categories outcome

	cytoplasm		nucleus		plasma membrane	
LSTM	precision	**0.875**	precision	**0.842**	precision	0.833
	recall	**0.827**	recall	**0.809**	recall	**0.791**
	F1 score	**0.851**	F1 score	**0.830**	F1 score	**0.812**
Bi-LSTM	precision	0.857	precision	0.825	precision	0.817
	recall	0.792	recall	0.753	recall	0.751
	F1 score	0.823	F1 score	0.787	F1 score	0.782
GRU	precision	0.860	precision	0.841	precision	**0.841**
	recall	0.815	recall	0.777	recall	0.779
	F1 score	0.837	F1 score	0.808	F1 score	0.809

Table 4. Comparison the performance of for LSTM model, Bi-LSTM model and GRU model for four subcellular localization categories outcome

	cytoplasm		nucleus		plasma membrane		extracellular	
LSTM	precision	0.791	precision	0.805	precision	0.791	precision	0.868
	recall	0.834	recall	0.788	recall	0.778	recall	0.753
	F1 score	0.812	F1 score	0.796	F1 score	0.784	F1 score	0.807
Bi-LSTM	precision	**0.886**	precision	**0.870**	precision	**0.870**	precision	**0.890**
	recall	0.849	recall	**0.815**	recall	**0.817**	recall	**0.783**
	F1 score	**0.867**	F1 score	**0.842**	F1 score	**0.843**	F1 score	**0.833**
GRU	precision	0.817	precision	0.830	precision	0.817	precision	0.886
	recall	**0.856**	recall	0.814	recall	0.806	recall	0.783
	F1 score	0.836	F1 score	0.822	F1 score	0.811	F1 score	0.831

4. CONCLUSIONS AND DISCUSSION

In this paper, we first performed necessary visualization of 11 types of proteins for preliminary analysis to see the distribution pattern of an amino acid within each protein. After we made the

visualization of the 11 types of, we introduced three different deep learning models which were GRU model, LSTM model, and bi-directional LSTM model separately. Then we used these three models to perform the prediction of subcellular localization of proteins with binary classification, three categories classification and four categories classification to see the performance of this model.

We have experimented with some neural network architectures to train a classifier for the task of protein family identification. Based on the evaluation indexes that we obtained in results part, we can demonstrate that GRU model, LSTM model, and bi-directional LSTM model are practical approaches to perform the prediction of the subcellular localization of proteins for binary classification, three categories classification and four categories classification.

Another thing is hyper-parameters in deep learning are many and tuning them will take weeks or months. Generally speaking, researchers do this tuning and publish a paper when they find a beautiful set of architecture which performs better than other. In this study, we tried to find the best hyper parameters practice for those models we proposed. Although these three models in our study performed pretty well in this study, if we compared the precision of train dataset with the precision of test data, we can noticed that the precision of train data can be much higher than that value of test data which apparently means the existence of over-fitting and this may be due to we didn't perform batch normalization for the current models in our study due to the resource and computational time limitation. In the further study, batch-normalization of the data can be considered to deal with this problem. Moreover, the decrease of drop-out probability and the decrease the number of layers in the recurrent neural network (RNN) can also be considered to handle this issue.

Another improvement of the current model will be focused on trying a state-of-the-art softmax function or introducing another attention mechanism into the LSTM model. Since in this study, the dataset compared with the sample size needed in general deep learning model and features are relatively small, and the distribution of the 11 types of proteins are not even. Therefore we consider trying some data augmentation mechanism in further study to avoid this problem. Another way to overcome this problem is transfer learning using those pre-trained nets, layers of which already knows how to extract features, we can don't have to tune the hyper parameters. Since they are already trained of some dataset, their pre-trained weights provide the first initialization of weights, and because of this, our RNN will converge very fast required less training data which otherwise can take days on these deep architectures.

REFERENCE

1. Jiren Wang, Wing-Kin Sung, Arun Krishnan & Kuo-Bin Li (2005). Protein subcellular localization prediction for Gram-negative bacteria using amino acid sub alphabets and a combination of multiple support vector machines. BMC Bioinformatics, 6:174.
2. Protein subcellular localization prediction, Wikipedia.
3. Emanuelsson, Olof, Brunak, Søren, von Heijne, Gunnar, and Nielsen, Henrik (2007). Locating proteins in the cell using TargetP, SignalP, and related tools. Nat. Protoc., 2(4):953–971.
4. Bairoch, A., & Apweiler, R. (2000). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. Nucleic acids research, 28(1), 45-48.
5. Höglund, A., Dönnes, P., Blum, T., Adolph, H. W., & Kohlbacher, O. (2006). MultiLoc: prediction of protein subcellular localization using N-terminal targeting sequences, sequence motifs, and amino acid composition. Bioinformatics, 22(10), 1158-1165.
6. Rost, Burkhard, et al. "Automatic prediction of protein function." *Cellular and Molecular Life Sciences* 60.12 (2003): 2637-2650.
7. Watson, J. D., Laskowski, R. A., & Thornton, J. M. (2005). Predicting protein function from sequence and structural data. *Current opinion in structural biology*, *15*(3), 275-284.
8. Friedberg, Iddo. "Automated protein function prediction—the genomic challenge." *Briefings in Bioinformatics* 7.3 (2006): 225-242.
9. Lee, D., Redfern, O., & Orengo, C. (2007). Predicting protein function from sequence and structure. *Nature Reviews Molecular Cell Biology*, *8*(12), 995-1005.
10. Wang, Z., Zhang, X. C., Le, M. H., Xu, D., Stacey, G., & Cheng, J. (2011). A protein domain co-occurrence network approach for predicting protein function and inferring species phylogeny. *PloS one*, *6*(3), e17906.
11. Rentzsch, R., & Orengo, C. A. (2009). Protein function prediction–the power of multiplicity. *Trends in biotechnology*, *27*(4), 210-219.
12. Wan, S., Duan, Y., & Zou, Q. (2017). HPSLPred: An ensemble multi-label classifier for human protein subcellular location prediction with the imbalanced source. *Proteomics*.
13. Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, *25*(17), 3389-3402.
14. Martin, D. M., Berriman, M., & Barton, G. J. (2004). GOtcha: a new method for prediction of protein function assessed by the annotation of seven genomes. *BMC Bioinformatics*, *5*(1), 178.
15. Zehetner, G. (2003). OntoBlast function: From sequence similarities directly to potential functional annotations by ontology terms. *Nucleic acids research*, *31*(13), 3799-3803.

16. Groth, D., Lehrach, H., & Hennig, S. (2004). GOblet: a platform for Gene Ontology annotation of anonymous sequence data. *Nucleic acids research*, *32*(suppl_2), W313-W317.
17. Hawkins, T., Chitale, M., Luban, S., & Kihara, D. (2009). PFP: Automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. *Proteins: Structure, Function, and Bioinformatics*, *74*(3), 566-582.
18. Deng, M., Zhang, K., Mehta, S., Chen, T., & Sun, F. (2003). Prediction of protein function using protein-protein interaction data. *Journal of Computational Biology*, *10*(6), 947-960.
19. Letovsky, S., & Kasif, S. (2003). Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics*, *19*(suppl_1), i197-i204.
20. Vazquez, A., Flammini, A., Maritan, A., & Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks. *Nature biotechnology*, *21*(6), 697-700.
21. Hishigaki, H., Nakai, K., Ono, T., Tanigami, A., & Takagi, T. (2001). Assessment of prediction accuracy of protein function from protein-protein interaction data. *Yeast*, *18*(6), 523-531.
22. Chua, H. N., Sung, W. K., & Wong, L. (2006). Exploiting indirect neighbors and topological weight to predict protein function from protein-protein interactions. *Bioinformatics*, *22*(13), 1623-1630.
23. Zeng, X., Zhang, X., & Zou, Q. (2015). Integrative approaches for predicting microRNA function and prioritizing disease-related microRNA using biological interaction networks. *Briefings in bioinformatics*, *17*(2), 193-203.
24. Cao, R., & Cheng, J. (2015). Deciphering the association between gene function and spatial gene-gene interactions in 3D human genome conformation. *BMC Genomics*, *16*(1), 880.
25. K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. (2014). On the properties of neural machine translation: Encoder-decoder approaches.*arXiv preprint arXiv:1409.1259*
26. Milad Mohammadi, Rohit Mundra, Richard Socher & Lisa Wang (2017). Natural Language Processing with Deep Learning.
27. S. Hochreiter and J. Schmidhuber (1997). Long short-term memory. *Neural Computation, 9(8):1735–1780.*
28. A. Graves (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850.*
29. J. Chung, C. Gulcehre & K. Cho (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555.*

30. A. Mousa and B. Schuller (2017). Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: A Generative Approach to Sentiment Analysis. *Association for Computational Linguistics, 1: 1023 – 1032.*