Subcellular location prediction of eukaryotic proteins using deep learning

Coursework of COMPGI10 Bioinformatics

Yan Song†

Abstract— **Protein plays an essential role in cells of organisms and research on subcellular location, which is believed to be useful to infer its functionality, has been carried out extensively. Analogous to human language, biological organisms use their own encoded language to convey information (Zeng et al. [2015]). Also inspired by sophisticated sequence processing techniques in Natural Language Processing, we implement various sequential models focusing on the text-based feature of protein. The experiment shows, on four major subcellular locations: Cytosolic, Secreted, Nuclear and Mitochondrial, our model achieves averaged 71% accuracy on the validation set. Code available online.***

I. INTRODUCTION

With the huge improvement of computational power, deep learning LeCun et al. [2015] has become a set of powerful models to learn a representation of large data. It gains its power mainly from back-propagation to perform gradient descent which effectively updates the internal parameters of the network. Furthermore, extensive research on Recurrent Neural Network (RNN) gives impressive results in various sequential data applications such as machine translation (Bahdanau et al. [2014]) in the natural language processing field.

Within bioinformatics, biological sequences share similarities with human language (Ho et al. [2019]). They both consist of fundamental units with high order structure, applying sequence analysis can find a hidden representation for predicting both of their functionality and structure. With these analogies, deep learning is used in various biological sequential tasks, such as DNA (Shen et al. [2018]) and RNA(Meng et al. [2019]) sequence analysis.

In this paper, we particularly focus on predicting the subcellular location of eukaryotic proteins using deep learning. Proteins are molecular devices that maintain biological function within living organisms. The advance in next-generation sequencing has revolutionised the way protein can be studied. Notably, large-scale proteomic studies have generated a vast amount of sequence data and annotation of these data represent a major obstacle (Dönnes and Höglund [2004]), such as subcellular location. The subcellular location provides a unique biochemical environment and is believed to affect protein functionality once it has changed (Murphy et al. [2000]). Hence, predicting the subcellular localization is of great importance to protein analysis and it has been studied by many researchers. (Emanuelsson et al. [2007], Petersen et al. [2011], Wan and Mak [2015]).

The computational approaches for subcellular location are mainly carried out from a few perspective Dönnes and Höglund [2004]. —(1) Amino acid composition; (2) target peptide; (3) sequence homology; (4) other features. However, many of these methods such as feed-forward neural network and support vector machine (Yu et al. [2006]), require feature engineering and can only take inputs with a fixed length. As an alternative approach, Sønderby et al. [2015] uses a long short term memory network which is a special type of recurrent neural network, combined with a convolutional filter enabling short motif detection in the input sequence. Based on which Almagro Armenteros et al. [2017] improved the model by applying an attention decoding layer. Attention network (Bahdanau et al. [2014]) has been widely used in deep learning which allows the model to focus more on certain regions of the input and less on the surrounding context. In protein prediction, attention detects the sorting signal and gives it a dominant impact on final prediction (Almagro Armenteros et al. [2017]). Their works on deep learning approach have shown that relying only on sequence information can keep up with or even outperform the state-of-the-art algorithm, including those relying on homology information.

This paper set out to assess the performance of different deep learning models. Inspired by the achievement in natural language processing, we see protein sequences as a special language with 23 alphabets corresponding to 23 amino acid code. We examine how LSTM, attention network and convolutional layer behave on the classification task of four major subcellular locations: Cytosolic, Secreted, Mitochondrial, and Nuclear. To unveil the black box, we also generate attention plots to explore how regions of the sequence contribute to the prediction task.

II. APPROACHES

This section introduces the embedding layer, LSTM cell, CNN cell and also the attention mechanism.

† MSc. CSML programme, University College London, WC1E 6BT, UK; Email: ucaby31@ucl.ac.uk

*https://github.com/YanSong97/subcellular-location-prediction

A. *Embedding*

Word embedding is an essential first step for its role of converting text message into numeric. One way to do this is to use one-hot vector, which is a fixed length sparse vector representation of i^{th} token(word) with value 1 at i^{th} entry and 0 elsewhere. For instance:

$$\texttt{Alanine}_{hot} = [1\ 0\ \cdots\ \cdots\ 0]^T \in \mathcal{R}^{vocab\ size}$$

$$\texttt{Proline}_{hot} = [0\ 0\ \cdots\ 0\ 1\ 0\ \cdots\ 0]^T \in \mathcal{R}^{vocab\ size}$$

However, this representation has a large dimension and the notion of distance between words is not presented either, as all words are equidistant from each other. A method to address these is to have a dense vector for each word with continuous value at each entry, such as:

$$\texttt{Alanine}_{dense} = [0.66\ 0.37\ \cdots\ 0.27]^T \in \mathcal{R}^d$$

where dimension d is a hyperparameter that can be set by the user.

In our experiment, we use the `nn.Embedding` package from *Pytorch* library(Paszke et al. [2017]). Although the vocabulary size in the data is relatively small, using larger embedding dimension can project each amino acid code into higher feature space where more similarity relationship can be discovered from the distance.

B. *Long Short Term Memory (LSTM)*

Long Short Term Memory was first introduced by Hochreiter and Schmidhuber [1997] which is a special type of recurrent neural network but with better long-term dependencies and vanishing gradient solved. A simplified overview of the model is shown in Figure 1.

Each amino acid code is firstly embedded into a numerical vector as an input along with results from the previous state to current LSTM cell, where linear and non-linear operations f take place. The result generated by current LSTM cell (say at time t) consists of a hidden state $\boldsymbol{h}_t$ and a cell state $\boldsymbol{c}_t$. They are both sent to the next LSTM cell and meanwhile, hidden state $\boldsymbol{h}_t$ is outputted and buffered for later usage. Having finished reading the whole sequence, the hidden state of final LSTM cell $\boldsymbol{h}_T$ is fed to a multi-layer perceptron after which a probability distribution on class labels is generated.

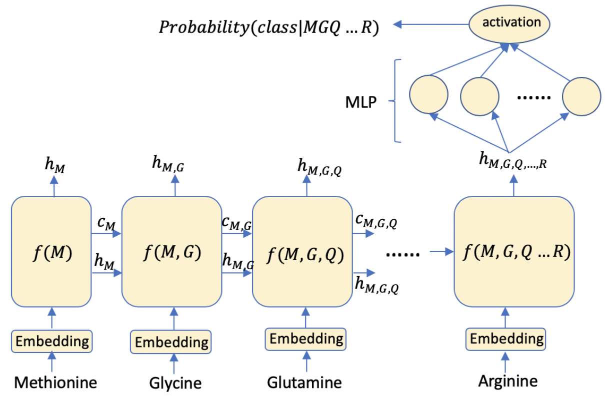


Fig. 1: LSTM with embedding layer. Each large block is a LSTM cell with the same internal parameters and functions setting. The network sweep through the amino acid sequence and generate a probability distribution for each class label at the end. h and C represent recurrent hidden state and cell state for LSTM, f represents transition function.

C. *Convolutional Neural Network(CNN)*

In image recognition and language modeling, convolutional neural network has shown state-of-the-art performance in many tasks(Krizhevsky et al. [2012], Kalchbrenner et al. [2014]). In image analysis, the primary purpose of a convolutional layer is to extract features from the input image, such as detecting edges or color. In language modeling, a convolutional layer extracts feature from tokens(words). Similarly, in protein sequence prediction, it can be interpreted as a motif detector regardless of the location in the sequence (Sønderby et al. [2015], Almagro Armenteros et al. [2017]). The illustration of CNN filter is shown in Figure 2. In our experiment, we use three different sizes of filters on the embedding matrix of sequences.

D. *Attention Mechanism*

Bahdanau et al. [2014] used attention mechanism together with LSTM in machine translating task and has obtained significant improvement. The idea behind it is that instead of using only the final hidden state ($f(M, G, Q, ..., R)$ in Fig 1) to make prediction, the attention network also utilise output at each time step during the roll-out of LSTM cell. Different to Bahdanau et al. [2014] where the generated output is a sequence of words such that attention need to be re-calculated for each predictive token, whereas our final output is simply a probability distribution. Hence, we only need to apply the attention network after the whole sequences are scanned.

Denote $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_T\}$ as all the hidden states of LSTM, which are hidden representation of the data. The attention scores $\boldsymbol{e}$ and normalised weights $\boldsymbol{\alpha}$ can be written as:

$$\boldsymbol{e}_t = \boldsymbol{W}_{att} tanh(\boldsymbol{W}_t \boldsymbol{h}_t + \boldsymbol{v}_T \boldsymbol{h}_T + \boldsymbol{b}_T) \tag{1}$$

$$\boldsymbol{\alpha}_t = \frac{\boldsymbol{e}_t}{\sum_{i=1}^{T} \boldsymbol{e}_i}\ ,\ 1 \leq t \leq T \tag{2}$$

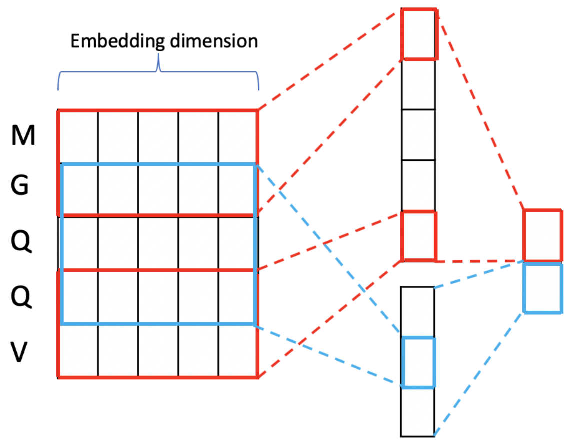


Fig. 2: Convolutional layer with multiple filters. The grid on the left represents the embedding matrix of the sequence. The blue and red rectangle represents two CNN filter with different size scanning over the embedding matrix. In the experiment we use max pooling and the resulting vector representations from different filters are concatenated.

where $\boldsymbol{W}_{att}, \boldsymbol{W}_t, \boldsymbol{v}_T, \boldsymbol{b}_T$ are trainable parameters.

The normalised weight $\boldsymbol{\alpha}_t$ can be seen as assigning importance with respect to the prediction distribution, to hidden state $\boldsymbol{h}_t$ as well as each position in the input sequence. Position with higher attention weight shows more dominant impact on subcellular location of the protein sequence.

The weighted sum (context vector in NLP) of hidden states is denoted as:

$$\boldsymbol{c} = \sum_{i=1}^{T} \boldsymbol{\alpha}_i \boldsymbol{h}_i \tag{3}$$

III. Models

In this section we propose three model variants illustrated in Fig 3, all three models are trained with back-propagation (LeCun et al. [1988]).

A. LSTM

During the experiment, we found out that normal LSTM model which takes the hidden state at the final time step as input for the decoder (as illustrated in Fig 1) is insufficient to train the data. Hence we take hidden state $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T\}$ at each time step in the recurrent layer into consideration when predicting the class labels. We append a max pooling layer to convert all hidden states into a pooled vector for label prediction.

As shown in Fig 3(a), Max pooling layer looks for the highest value along the sequence in each hidden dimension (row of matrix) and generates a new hidden state vector. Intuitively, this can be seen as integrating the most prominent feature in each entry of hidden state. Following the pooling layer is a two-layer MLP which, alternatively speaking, performs linear transformation on the new hidden state $\boldsymbol{h}_{new}$:

$$output^L = \boldsymbol{W}_2(\boldsymbol{W}_1 \boldsymbol{h}_{new} + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

The final softmax layer $\boldsymbol{\sigma}$ normalise the linear layer output o^L to generate a probability distribution written as:

$$P(class_i) = \boldsymbol{\sigma}(o^L)_i = \frac{exp(o_i^L)}{\sum_{i=1}^{4} exp(o_i^L)} \ , \ i = 1, 2, 3, 4$$

We have also implemented bi-direction LSTM which contains a backward recurrent model scanning from the end of the sequence to the start (dashed arrow in Fig 3(a)), doubling the hidden dimension and the representativeness of the mode as well.

B. LSTM with attention

Almagro Armenteros et al. [2017] proposed multi-layer attention decoder. However, that will inevitably increase the model complexity and computational burden, thus for simplicity reason, we only implement a single attention layer.

Based on LSTM model, attention network establish an extra non-linear transformation on each of the hidden states $\boldsymbol{h}_i$ to obtain their corresponding attention score $\boldsymbol{e}_i$. The normalised score $\boldsymbol{\alpha}_i$ is also computed expressing the importance of $\boldsymbol{h}_i$ with respect to the prediction task. Differs from LSTM proposed in section III-A, attention net generates a new hidden representation $\boldsymbol{c}$ by taking the weighted sum over $\boldsymbol{h}_i$ (eq. 3). An advantage of this is that it can be straightforwardly interpreted as a weighted sum over each of the amino acid codes whereas a Max-pooling layer focus on the most dominant feature value in each of the feature space dimension. Meanwhile, an attention plot is more interpretable to see how each amino acid code contributes.

C. CNN-LSTM

As shown in Fig 3(c), a convolutional layer with multiple filters gives extra ability detecting the feature before sending embedding matrix to LSTM network. Each filter is a fixed size window sliding over the embedding matrix, after which the results from different filters are concatenated to form a new feature matrix and sent to the following LSTM network. The width of filters, says 3, can be understood as detecting how important 3-gram(3 consecutive amino acid code combination) is for the ultimate classification task.

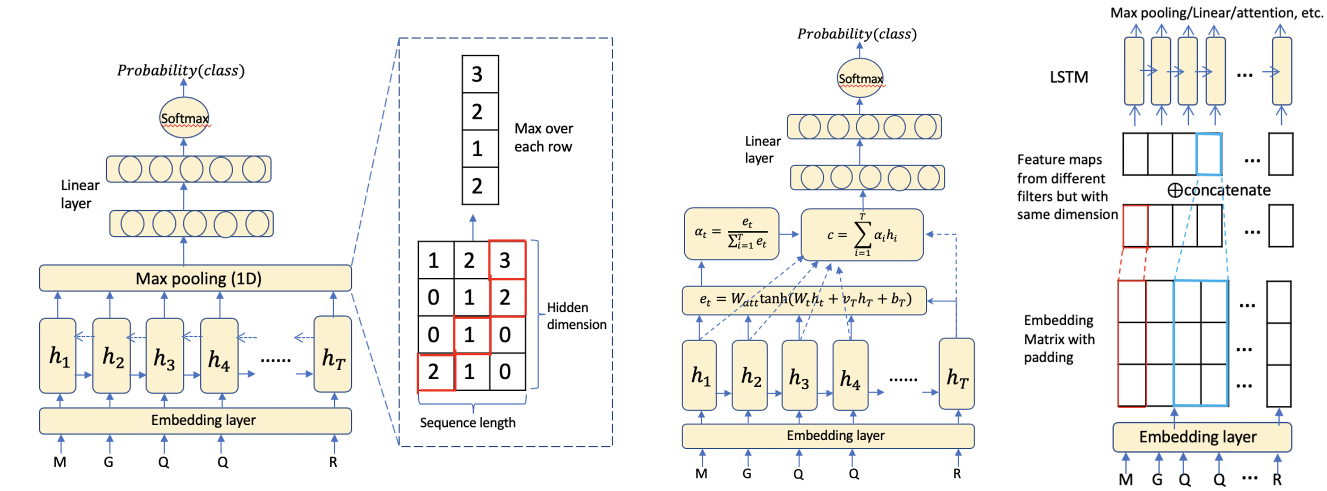
IV. Data

The protein data used is in Fasta format and we assume every sequence to be unique(non-homologous). Sequences are classified into four subcellular locations: Cytosolic, Secreted, Mitochondrial and Nuclear.

A. Train data

The training data contains 9,222 eukaryotic proteins sequences among which the numbers for each subcellular location are 3004 (Cyto), 1065 (Secreted), 1299 (Mito) and 3314 (Nuclear). Table I shows the statistics of sequence

Fig. 3: Model plots



(a) LSTM with max pooling on every hidden states. Note that the reversed dashed arrows represent backward LSTM which shows up in bi-directional LSTM. See section III-A

(b) LSTM with attention mechanism. See section III-B

(c) Convolutional layer with multiple filters followed by LSTM. See section III-C

.

length. The distribution of length is shown in Fig 4 (left). As most of the sequences are within 2,000 length, in the experiment we truncate the training sequence longer than 2,000 so that the dimension of model parameters is largely reduced.

	Min	Max	mean	std
Length	11	13100	546.85	514.32

	Cyto	Secreted	Mito	Nuclear	Total
$0 \leq l < 100$	44	619	67	35	765
$100 \leq l < 500$	1379	716	929	1626	4650
$500 \leq l < 1000$	1047	197	278	1165	2687
$1000 \leq l < 1500$	350	32	20	316	718
$1500 \leq l < 2000$	104	22	3	106	235
$2000 \leq l < 2500$	44	9	1	27	81
$2500 \leq l$	36	10	1	39	86
Total	3004	1605	1299	3314	9222

TABLE I: training data statistics

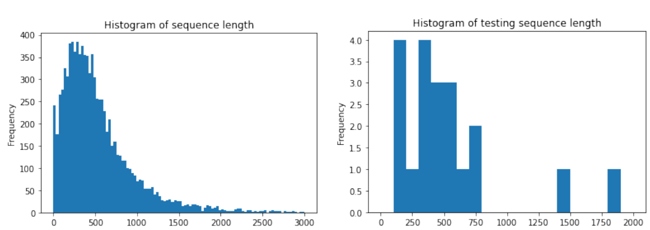


Fig. 4: Histogram of training(Left) and testing(Right) data length

B. *Blind test data*

The testing data contains 20 unlabelled sequences with maximum length at 1,876 and minimum at 141. The distribution is shown in Fig I (right).

V. Experiments and Results

To reduce the computational time and the parameters dimensions, we keep the training sequence within 2,000 length and truncate the longer sequences. When a sequence is truncated, the first 2,400 and the last 100 amino acid codes are kept and those from the middle are removed. This way of truncation will not discard the peptide signal present at N-terminus and N-terminus. For sequences shorter than 2,000 we pad them with a special padding token <PAD> in the middle which enables mini-batching and sending multiple sequences to the model at the same time.

First step of the experiment is to split the data into training and validation set. We randomly shuffle the original data and split it into five folds. The first four are training data and the last fold is the validation set for evaluating the performance of different models. Next, we train various models separately using the same training set and evaluate on the validation set during each epoch. After that, we select the one with the best performance on the validation set and perform 5-fold cross-validation.

A. *Data pre-processing*

The original data in letters is converted to sequences in numbers by building up a vocabulary dictionary and mapping each letter to a unique integer. <PAD> and <OOV> are two special tokens representing padding and out-of-vocabulary letters.

After the data splitting, the size of training data is 7,378 and the size of validation set is 1,844. Throughout the model comparison the both of these data set are fixed.

B. *Training and validation*

Four models are trained and validated using the same splitted data:

1) LSTM (model III-A)

2) Bi-directional LSTM
3) LSTM with attention mechanism (model III-B)
4) LSTM with convolutional filters (model III-C)

The hyperparameters setting is:
- Batch size: 64
- Hidden dimension: 128
- Embedding dimension: 64
- learning rate: 0.001
- filter size(LSTM-CNN): [3,5,9]

Fig 5 shows the loss and accuracy of validation data during training. Among the models convolutional layer with LSTM has the lowest training loss and highest accuracy at convergence, whereas attention mechanism performs the worst during training. However, all models converge to generally the same validation loss and accuracy.

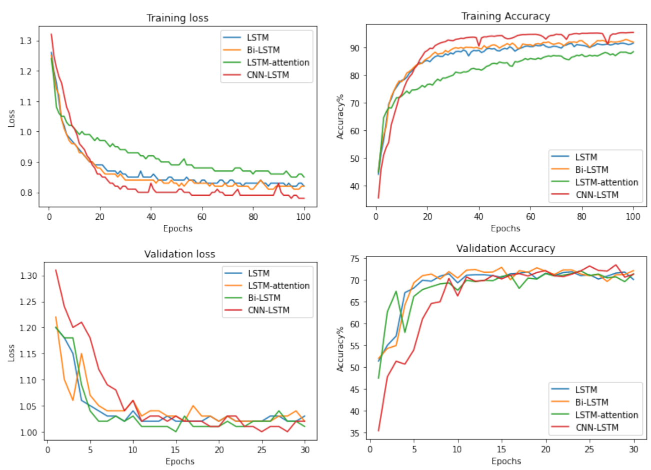
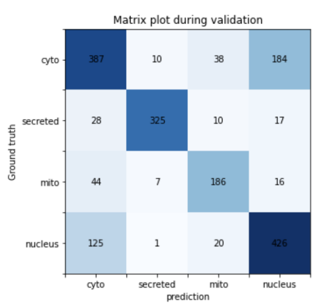
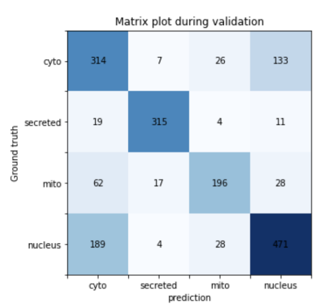


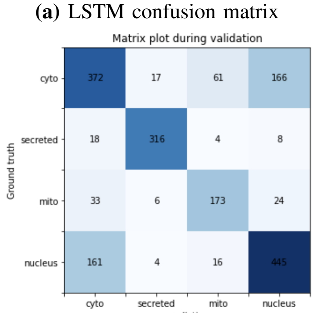
Fig. 5: Plots of validation loss and accuracy



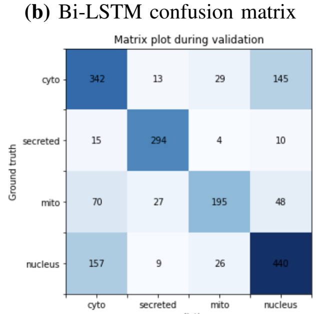
(a) LSTM confusion matrix



(b) Bi-LSTM confusion matrix



(c) LSTM-attention confusion matrix



(d) CNN-LSTM confusion matrix

Fig. 6: Confusion matrix during validation at convergence

Furthermore, in Fig 6 we have plotted the confusion matrix during validation for each models, from which we can observe that the models are more likely to mix up Cytosolic and Nuclear protein sequences while differentiating well between secreted and Mitochondrial sequences. To verify the assumption we can calculate the precision, recall and F1 score for each protein classes using formula:

$$\texttt{Precision} = \frac{\texttt{true positive}}{\texttt{true positive} + \texttt{false positive}}$$

$$\texttt{Recall} = \frac{\texttt{true positive}}{\texttt{true positive} + \texttt{false negative}}$$

$$\texttt{F1} = \frac{2 \times \texttt{precision} \times \texttt{recall}}{\texttt{precision} + \texttt{recall}}$$

where in a classification problem, higher precision shows that the data with predicted label i is more likely to be correct, however, the amount of data from class i but with wrong predicted labels is not guaranteed to be small. On the other hand, high recall demonstrates that the data from class i is more likely to be correctly predicted but it says nothing about the data from different classes $j, j \neq i$ but labelled i. F1 score considers both precision and recall value and is a reasonable measure of testing accuracy.

The statistics for each model are presented in table II and it is expected to see that all models perform best on secreted and worst on cytosolic protein. A follow-up experiment regarding the binary classification of protein sequences yields 95% validation accuracy on Secreted&Mito, 95% on Cyto&Secreted, 87% on Cyto&Mito and 70% on Cyto&Nuclear. These may suggest that Cytosolic and Nuclear protein sequence has less distinct signal peptide so that it is relatively harder to recognise these two while Mitochondrial and Cytosolic protein has more.

(a)	P	R	F1
Cyto	0.663	0.625	0.643
Secr	0.948	0.855	**0.899**
Mito	0.732	0.735	0.733
Nucl	0.663	0.745	0.702

(b)	P	R	F1
Cyto	0.538	0.654	0.590
Secr	0.918	0.903	**0.910**
Mito	0.772	0.647	0.704
Nucl	0.733	0.681	0.706

(c)	P	R	F1
Cyto	0.637	0.604	0.620
Secr	0.921	0.913	**0.917**
Mito	0.681	0.733	0.706
Nucl	0.692	0.711	0.701

(d)	P	R	F1
Cyto	0.586	0.647	0.615
Secr	0.857	0.91	**0.883**
Mito	0.739	0.574	0.646
Nucl	0.684	0.696	0.690

TABLE II: Precision, recall and F1 score for each model.

Taking advantage of attention mechanism, Fig 8-11 (Appendix IX-A) visualise which regions of the sequences are essential to the subcellular location prediction.* Note that sequences with length smaller than 2,000 are zero-padded from the middle so that N and C terminus align. For Cytosolic protein, the attention weights seems to spread across the sequence showing no particularly

*The plots use correct prediction samples during validating at convergence significant region. Nuclear protein has slightly more scattered important-regions located at both N and C terminals. On the contrary, Secreted and Mitochondrial protein have much more clear pattern of attention weights. For Secreted protein, the model mainly focus on the signal peptide located at the N-terminus of the sequence with short and roughly fixed length, which explains its high predictive accuracy. The Mitochondrial protein has relatively larger regions at N-terminus and also a small amount of attention at C-terminus.

Also in Appendix IX-B we have listed out a few example sequences of Secreted and Mitochondrial protein with important regions highlighted. We can observe that on short sequence, the beginning two to four amino acid codes play an essential role while for long sequences the important region is shifted slightly to the right.

C. *Best model*

According to Opitz and Burst [2019], there are two ways to calculated the averaged F1 score so-called *macro F1* for each model:

1) Average of F1:

$$F^{(1)} = \frac{1}{4}(F_{cyto} + F_{secr} + F_{mito} + F_{nucl}) = \frac{1}{4}\sum_{i=1}^{4}\frac{2P_iR_i}{P_i+R_i}$$

2) F1 of average:

$$F^{(2)} = \frac{2\times\hat{P}\times\hat{R}}{\hat{P}+\hat{R}} = 2\frac{(\frac{1}{4}\sum P_i)(\frac{1}{4}\sum R_i)}{(\frac{1}{4}\sum P_i)+(\frac{1}{4}\sum R_i)}$$

We can see in $F^{(1)}$ the precision of each class is multiplied with the recall of the same class and in $F^{(2)}$ it is multiplied with the recall of all other classes. Although the average of F1 is stated to have more robustness, both ways of computation lead to same ranking of the models. The values of $F^{(1)}$ are: (a): 0.7443, (b): 0.7275, (c): 0.736, (d): 0.7085. The values of $F^{(2)}$ are: (a): 0.7457, (b): 0.729, (c): 0.7365, (d): 0.7116. Thus the rank regarding the F1 score is:

$$LSTM > LSTM_{att} > Bi-LSTM > CNN-LSTM$$

However, we choose LSTM with attention over pure LSTM model as attention mechanism can provide a more explicit interpretation for the prediction the model generates.

D. *Cross-validation*

We implement 5-fold cross-validation using LSTM-attention model. Fig 13 (Appendix IX-C) shows the training and validation statistics for each fold of data. The mean validation accuracy at convergence is present as below:

fold-1	fold-2	fold-3	fold-4	fold-5	Mean
71.368	71.494	69.888	72.785	72.231	71.553

the mean confusion matrix is shown as:

True \ Pred	Cyto	Secr	Mito	Nucl	Total
Cyto	**388.8**	15.0	48.6	183.4	635.8
Secr	15.6	**288.6**	8.0	9.8	322
Mito	27.2	5.8	**175.6**	12.6	221.2
Nucl	161.6	6.8	24.6	**452.0**	645
Total	593.2	316.2	256.8	657.8	**1824**

TABLE III: Mean confusion matrix

From which we can compute the corresponding precision, recall and F1 score for each class:

	Cyto	Secr	Mito	Nucl
Precision	0.6554	**0.9127**	0.6838	0.6871
Recall	0.6115	**0.8963**	0.7939	0.7008
F1	0.6327	**0.9044**	0.7347	0.6939

TABLE IV: Mean precision, recall and F1 score for cross-validation

The model achieve high F1 score on Secreted protein and relatively smaller score on Cytosolic protein which match with our previous results.

VI. Prediction

In this section, we train a LSTM-attention model with whole data set and generate predictions on blind test set along with complementary results. The number of training epochs is set to be 20 where validation accuracy plateaus according to Fig 13.

A. *Prediction table* (shown in Fig 14)

The model is quite certain for most of the cases, especially when predicting Secreted protein. There are two cases where the model is less confident between Cytosolic & Mitochondrial (SEQ608) and Mitochondroal & Nuclear (SEQ862). It is also interesting to see that nearly half of the sequences are assuredly predicted as Cytosolic which is previously shown to be the least convinced label among all. This might suggest that some of the sequences predicted as Cytosolic are wrong and deeper analysis is needed.

B. *Attention visualisation*

Unexpectedly, the attention plot (Fig 15) of testing data meets our assumption from previous attention graphs. The sequences with important regions concertrated on N-terminus are mostly predicted as Secreted protein. Table V has listed testing sequences with highlighted important regions.

C. *Embedding visualisation*

We also plot the trained embedding weight which is the high dimension representation of each amino acid code. We use t-distribution stochastic neighbor embedding(tSNE) (Maaten and Hinton [2008]) to reduce the dimension and visualise in 2-d graph (Fig 7). The closer distance shows the similar property in the prediction task.

Fig. 7: Trained embedding layer in two dimension. The letters represent the amino acid code and the distance between two letters demonstrates the similarity.

VII. Discussions

Unlike traditional NLP tasks with extremely large vocabulary size, the protein prediction task has a limited amount of unique amino acid codes, thereby increasing the difficulty in prediction. This means the deep learning models need to form more and deeper feature representations of the data.

In our model setting, an embedding layer capture the similarity and high dimensional distances between amino acid codes; a LSTM model takes the order of sequences into account; attention network looks for the important regions which contribute to the prediction task and a convolutional filter focus on whether the combination of N consecutive codes matters. These text-based-only features might be useful for signal sequence prediction and residue composition analysis, but not sufficient to predict the subcellular location for certain protein as we have seen the results for Cytosolic and Nuclear. It can be the case that these proteins sequences have long and scattered signal peptide reflected from their attention plots (Fig 8, 9). Or other factors such as membrane association and post-translational modification may affect subcellular location as well while being significantly hard to capture from text-based information. Thus, feature engineering techniques can be incorporated to enlarge the expressivity of deep learning models, which can be set as our potential future work.

Looking back at the experiment, there are still a few things worth discussing. The bi-directional LSTM does not improve the performance. The validation plots (Fig 5) shows no significant difference between LSTM and Bi-LSTM, suggesting the model only utilise the forward order. This may indicate that the reverse protein sequence has little effect of its functionality which is biologically reasonable since the chemical structure of a protein will change largely if the order is reversed.

VIII. Conclusion and Future works

In this paper, we have examined different deep learning model variants on sequential protein data. On subcellular location prediction tasks, particularly for Cytosolic, Secreted, Mitochondrial and Nuclear protein, all models give similar validation accuracy at convergence and they all perform the worst on Cytosolic protein and the best on Secreted protein. Attention mechanism enables us to discover that Secreted and Mitochondrial protein have clear signal peptide at N-terminus while the attention for Cytosolic and Nuclear are less obvious. We have also generated a prediction on 20 blind testing set and their corresponding attention visualisation graph.

The potential future works of this report are multiple. Such as a deeper exploration on convolutional layer on protein data and also, as we discussed before, the incorporation of feature engineering technique, like a pre-trained biological embedding (Asgari and Mofrad [2015]). Furthermore, by looking at the probability prediction for each class, we can discover that the model tend to be over-confident which might not be desired in many scenario. One way to provide better probabilistic interpretation is to introduce uncertainty quantification method such as *Bayesian neural network* Blundell et al. [2015].

References

José Juan Almagro Armenteros, Casper Kaae Sønderby, Søren Kaae Sønderby, Henrik Nielsen, and Ole Winther. Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, 2017.

Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11), 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Pierre Dönnes and Annette Höglund. Predicting protein subcellular localization: past, present, and future. *Genomics, proteomics & bioinformatics*, 2(4):209–215, 2004.

Olof Emanuelsson, Søren Brunak, Gunnar Von Heijne, and Henrik Nielsen. Locating proteins in the cell using targetp, signalp and related tools. *Nature protocols*, 2(4):953, 2007.

Quang-Thai Ho, Dinh-Van Phan, Yu-Yen Ou, et al. Using word embedding technique to efficiently represent protein sequences for identifying substrate specificities of transporters. *Analytical biochemistry*, 577:73–81, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (Nov):2579–2605, 2008.

Jun Meng, Zheng Chang, Peng Zhang, Wenhao Shi, and Yushi Luan. lncrna-lstm: Prediction of plant long non-coding rnas using long short-term memory based on p-nts encoding. In *International Conference on Intelligent Computing*, pages 347–357. Springer, 2019.

Robert F Murphy, Michael V Boland, Meel Velliste, et al. Towards a systematics for protein subcellular location: Quantitative description of protein localization patterns and automated analysis of fluorescence microscope images. In *ISMB*, volume 8, pages 251–259, 2000.

Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Thomas Nordahl Petersen, Søren Brunak, Gunnar Von Heijne, and Henrik Nielsen. Signalp 4.0: discriminating signal peptides from transmembrane regions. *Nature methods*, 8(10):785, 2011.

Zhen Shen, Wenzheng Bao, and De-Shuang Huang. Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, 8(1):1–10, 2018.

Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen, and Ole Winther. Convolutional lstm networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*, pages 68–80. Springer, 2015.

Shibiao Wan and Man-Wai Mak. *Machine learning for protein subcellular localization prediction*. Walter de Gruyter GmbH & Co KG, 2015.

Chin-Sheng Yu, Yu-Ching Chen, Chih-Hao Lu, and Jenn-Kang Hwang. Prediction of protein subcellular localization. *Proteins: Structure, Function, and Bioinformatics*, 64(3):643–651, 2006.

Zhiqiang Zeng, Hua Shi, Yun Wu, and Zhiling Hong. Survey of natural language processing techniques in bioinformatics. *Computational and mathematical methods in medicine*, 2015, 2015.

IX. Appendix

A. *Attention visualisation of Cytosolic, Secreted, Mitochondrial and Nuclear protein*

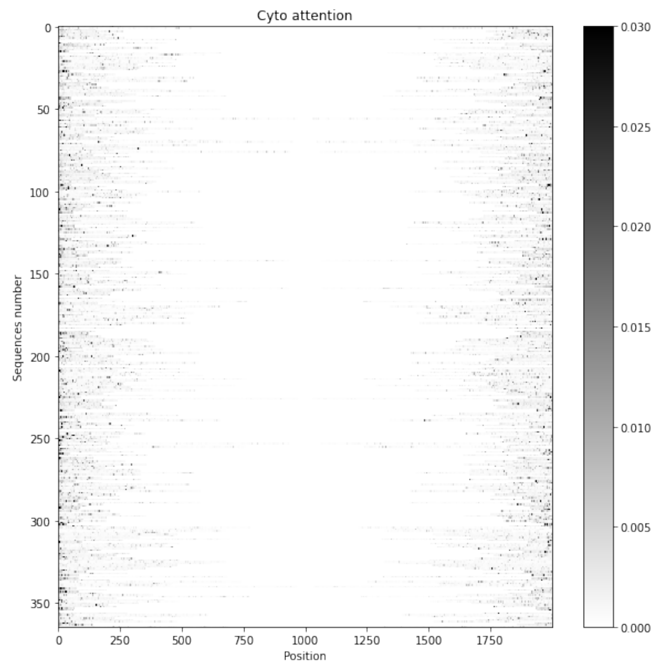


Fig. 8: Attention plot of cytosolic protein

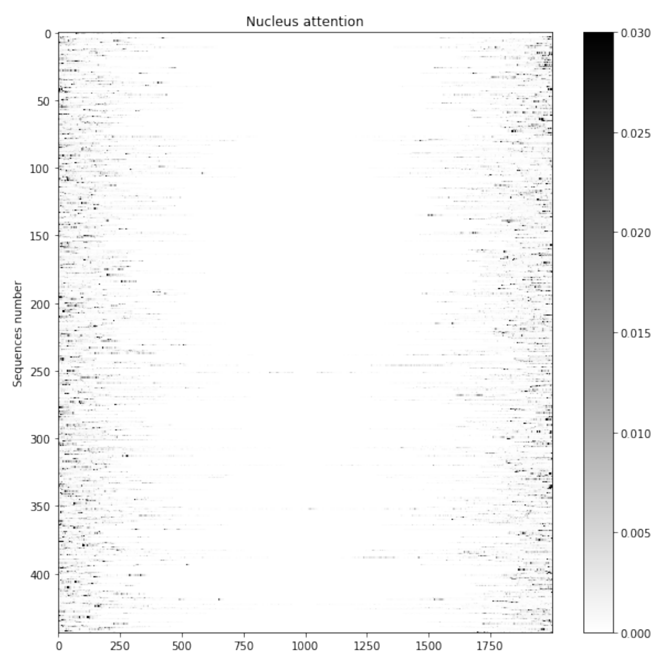


Fig. 9: Attention plot of Nuclear protein

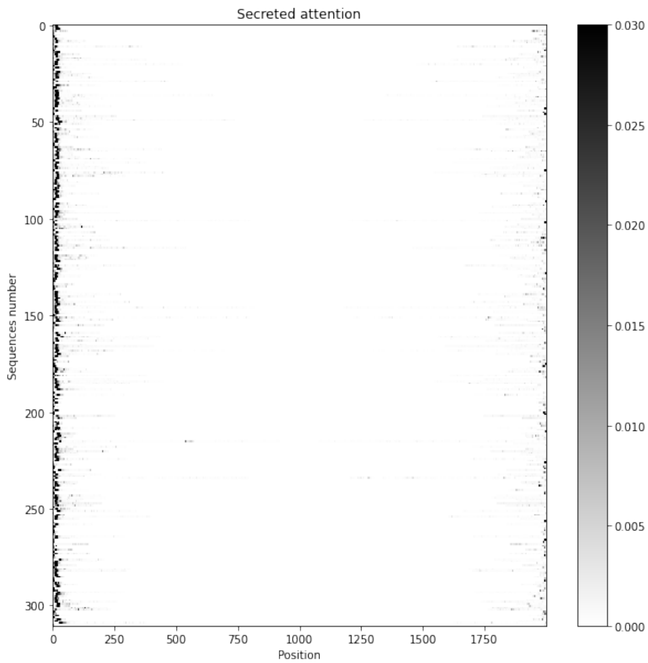


Fig. 10: Attention plot of secreted protein

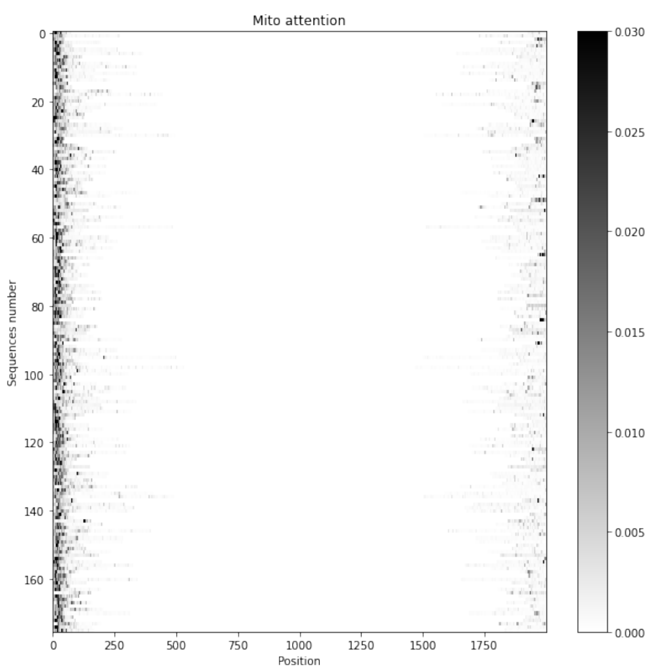


Fig. 11: Attention plot of Mitochondrial protein

B. Detailed important regions

Secreted protein (threshold=0.05):

INWKKIASIGKEVLKAL
IIGHLIKTALGMLGL
GCMKEYCAGQCRGKVSQDYCLKHCKCIPR
MKFTATFLMMAIF**VLMVE**PGECGWGSFFKKAAHVGKH...
AFVKGSAQRVAHGY
FLSLIPKIAGGIASLVKNL
MKHLIVAVVLLS**ALAICTSAE**EEQVNVPFRPEERIGECA...
MKQTIVIVLL**AAVA**MMA**CL**QMVAAEPLPEAAPAPSPLAEAE...
SKRLSNGCFGLKLDRIGAMSGLGCWRLINESK

Mitochondrial protein (threshold=0.015):

MFP**GMFMRKPDKA**EALKQL**R**THVALFGSWVVIIRAAPYVLSYFSD
+SKDELKIDF

MVKVKSKNSVIKLLSTA**ASGYSRYISIKKGAP**LVTQV**R**Y
+DPVVKR**H**VLFKEAKK**RK**VAERKPLDFLRTAK

MAALRPGSRAL**RRLLCRSFSGGGGVRLARERP**TDH**R**DAAS
+SRVSRFCPPRQSCHDWIGPPDKCSNL...

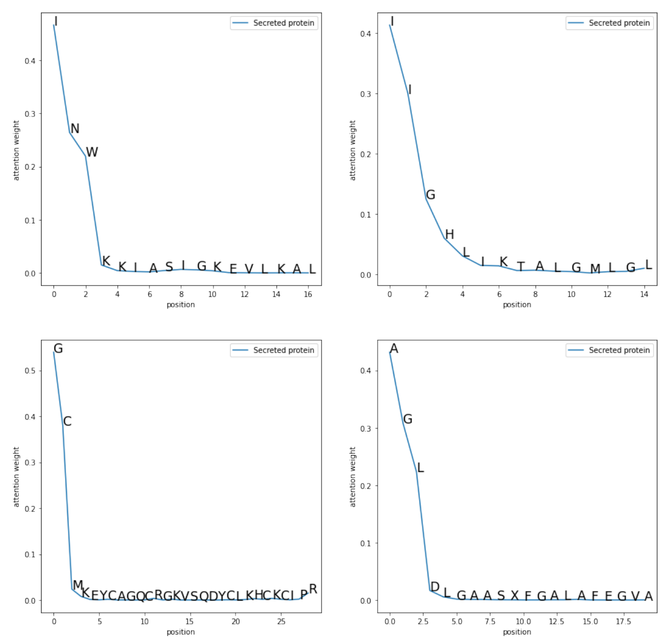


Fig. 12: Attention weight of Secreted proteins

C. Cross-validation plots

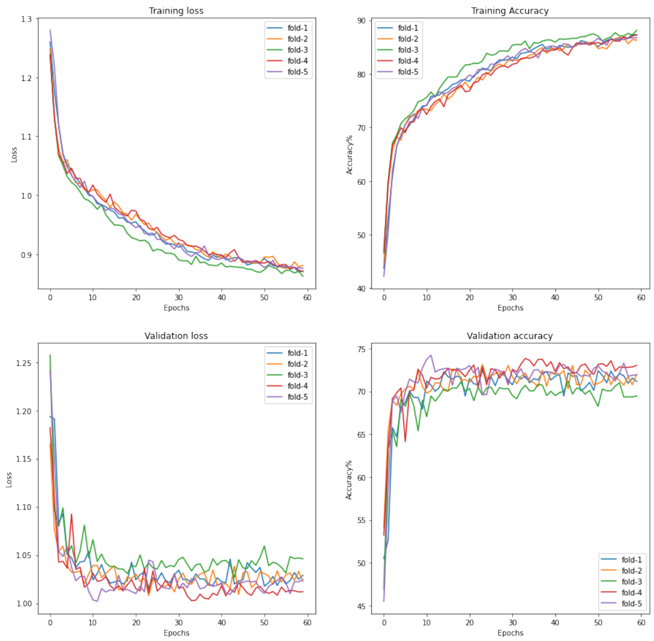


Fig. 13: Training and validation loss and accuracy during 5-fold cross-validation

D. Testing prediction and attention visualisation

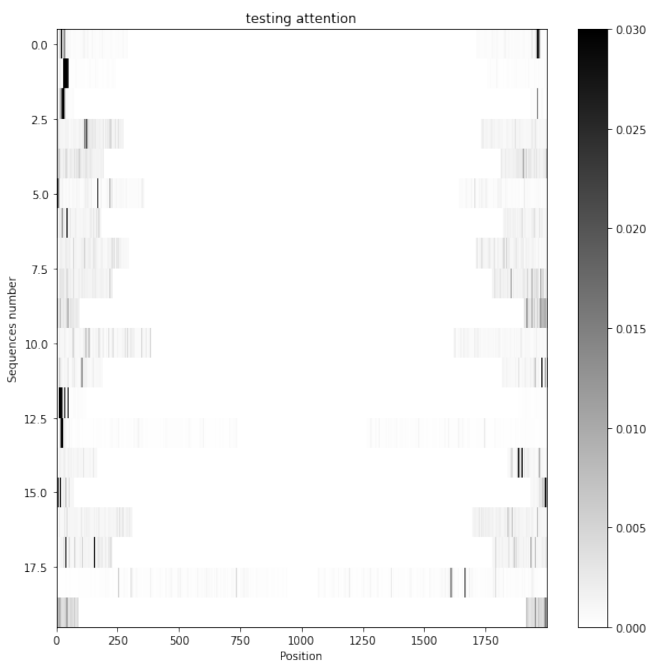


Fig. 15: Testing attention

	cyto	secreted	mito	nucleus	Prediction
SEQ677	0.066	0.934	0.000	0.000	secreted
SEQ231	0.000	1.000	0.000	0.000	secreted
SEQ871	0.000	1.000	0.000	0.000	secreted
SEQ388	0.995	0.000	0.000	0.005	cyto
SEQ122	1.000	0.000	0.000	0.000	cyto
SEQ758	0.000	0.000	0.000	1.000	nucleus
SEQ333	0.001	0.000	0.000	0.999	nucleus
SEQ937	1.000	0.000	0.000	0.000	cyto
SEQ351	1.000	0.000	0.000	0.000	cyto
SEQ202	1.000	0.000	0.000	0.000	cyto
SEQ608	0.760	0.000	0.240	0.000	cyto
SEQ402	0.000	0.000	0.000	1.000	nucleus
SEQ433	0.000	1.000	0.000	0.000	secreted
SEQ821	0.000	1.000	0.000	0.000	secreted
SEQ322	0.000	0.000	0.000	1.000	nucleus
SEQ982	0.000	0.000	0.000	1.000	nucleus
SEQ951	1.000	0.000	0.000	0.000	cyto
SEQ173	1.000	0.000	0.000	0.000	cyto
SEQ862	0.078	0.000	0.318	0.604	nucleus
SEQ224	0.999	0.000	0.001	0.000	cyto

Fig. 14: Testing table

E. *Code briefing*

1) Data processing:

- Read `.fasta` data
- build dictionary: the function takes as input raw amino acid sequences and outputs a dictionary and reverse dictionary containing all tokens (amino acids), encoded amino acid sequences and the sequence length for each protein and the max sequence length across all proteins.
- Save pickle file: for later usage.
- Statistic summary: create length distribution table and histogram

2) Model:

- Library: Pytorch
- __init__: setting up embedding layer, LSTM layer with initial hidden and cell states also claimed as model parameters, a CNN layer with input channel size 1 and output channel size equals to embedding dimension, two linear layers and three attention(also linear) layers.
- attention: takes all hidden states and padding mask as input, assigning zero value to all padding elements and returning context vector together with attention weight.
- forward function: pass encoded input sequence through layers and ouput results with dimension 4 along with attentin weights.

3) Utility function:

- gradient_clamping: clamp the gradient to avoid exploding gradient.
- data_shuffling.
- truncate the data: for sequence larger than 2,000, truncate it.

4) Training and validation:

- set_model: lstm, attention or CNN.
- save_model: checkpointing.
- batcher: mini-batching the data.
- train in one epoch: takes encoded training sequence, labels and original length as input. Generate prediction and compute cross-entropy loss. Perform back-propagation using Adam Kingma and Ba [2014].
- Validation in one epoch: takes validation data, compute validation loss and accuracy, updating attention and confusion matrix.
- run: takes input encoded sequence as inpout, implement k-fold splitting to generate training and valdiation data set. Do training and validation in each epoch, output loss, accuracy and confusion matrix.
- train_full_data: function training models on full input data without splitting.

5) Testing:

- testing_padding: if the input test data is shorter than batch size, perform padding.
- testing: taks encoded testing sequence and length as input, generate prediction and attention weight.
- print_frame: create dataframe of the testing result.

X. Supplementary material

TABLE V: Testing sequences

SEQ677	MESKGASSCRLLFCLL**ISATVFRP**G**L**G**WY**TVN**SAY**GDTIIIPCRLDVPQ.....IVVGLL**LAALVA**G**VVY**W**LY**MKKS KTASKHVNKDLGNMEENKKLEENNHKTEA
SEQ231	MPGPRVWGKYLWRSPHSKGCPGAMWW**LLLWGVLQACPTRGSVLLAQEL**PQQLTSPGYPEPYG....IKGVMNGKN
SEQ871	MDSRICTSFARLMASA**LCVSTLLVTAMPFDLRRGSS**DTDLDLQG....SGSGYNMLMKMQRHG
SEQ388	M**C**SLGLFPPPPPPRGQVTLYEHNNELVTGSSYESPPPDFRGQW........WVKTASGWA**LALCR**W**ASSL**HGSL FPHLSLR**S**EDLIAEFAQ.......PGPPPVLPHSPHSHL
SEQ122	MNPQIRNPME**R**MYR**D**TFYDNFE.....YKCFQITWFV**S**WTPCPDCVAKL.....RVTWFISW**S**PCFSW GCAG......LSGRL**R**AILQNQGN
SEQ758	MV**RK**ST**RRTA**KASEKP...STTPKSS HSPPSTRK**RRG**SVGTTATH...SNGDKAAK**RKG**SFNDTKPIVNV PSIVEIDDPTILDIVEEEPLA**R**NDSFSS....PQFMQNAISAAQKKASKQ
SEQ333	MP**H**SHPALTPEQKKELSDIAHR**I**V**A**PGK GILAADESTGSIAK**RLQ**SIGTENT....PEILPDGDHD**L**KR CQY....ASESLFISNHAY
SEQ937	MAAADGDDSLYPIAVLIDELRNEDVQ...TVEETVVRDKAVESLRAISHEHSPSDLEAHFVPLVK**RL**AGGDW....LTVLSLA
SEQ351	MGEEYKKTHTLVFHTS....NPVELFF EIPR**L**QNANLREALLDISR KSDIKALIIDFFCNAAF EVSTSMNIPTYFDVS GG....FVTHCGW SSVLEALSFGVPMIGWPLY AEQRINRVFMVEEIKVALPLDEEDGFV AMELEK**R**VRE**L**MES VK....FINSVTR
SEQ202	**M**ATPASAPDTRALVAD FVGYKLRQKGYVCGAGPGEGPAADPLHQAMRAAGD EFETRFRRTFSDLAAQLHVTPGSAQQRFTQVSDELF QGGPNWGRLVAFFVFGAA**L**C**A**E SVNKEMEPLVGQVQEWMV**AYL**ETQLADWIH SSGGW**A**EFTALYGDGALEEARRL**RE**G NW**A**S**VR**TV**L**TGA V**AL**G**ALV**TVGAFF**AS**K
SEQ608	MAEAHQ....VSDWWEEYIY**L**RGRGPL MVNSNYYAMDLLYILPTHIQAARAGNAIHAILLYRRKLDR EEIKPIRLLG**S**TIPLC....ENLINFHISSKF SCPETDSHRFG**R**HLKEAMTDIITLFGLSSNSKK
SEQ402	MGIQGLAKLI....DGKPPQLKSGELAK**R**S E**R**RA**E**A**E**KQLQQAQAAG AEQ....LHKEAHQLFLEPEVLDPESVELKWSEPNEEELI KFMCGEKQFSEERIRSGVK**RL**SKSRQGSTQG RLDDFFKVTGSLSSAK**RKE**PEPKGSTK KK**AKT**GAAGKFK**RG**K
SEQ433	MIAFSLLCLA**AVLRQSFGNVD**FNS**E**STRRKK**KQKE**IVDLHNSLR**RRV**SPTASNM.....ASCFCRNKII
SEQ821	MGKNKLLHPSLVLLL**VLLLPTDASVSGKP**QYMVLVP SLLHTETTEKGCVLLSYLNETVTVSASLESVRGNRSLFTDLEAENDVLH CVAF........TGSRSASNMAIVDV KMVSGFIPLKPTVKMLERSNHVSR TEVSSNHVLIYLDKVSNQTLSL FFTVLQDVPVRDLKPAIVKVYDYYETDE FAIAEYNAPCSKDLGNA
SEQ322	MAQLPPKI...NNYNESDEVQSQCKTEPQDGPSANQNSGGS SGNRIHDPK**RVKRILA**NRQSAQ**R**S**R**VR K**L**QYISELERSVTSLQTEVSVLSPRVAFLDHQRLLLNVD NSAIKQ**R**IAALAQDKIFKDAHQEALKREIERLR QVYHQQSLKKMENNVSDQSPADIKPSVEKEQLLNV
SEQ982	MGK**RKKSTRK**PTK**RLV**QKLDT KFNCLFCNHEKSVSCTLDKKN SIGTISCKICGQSFQTRINSL SQPVDVYSDWFDAVEEVNS GRGSDTDDGDEGSDSDYESDSEQ DAKTQNDGEIDSDEEEVD SDEERIGQVK**RGRGALV**DSDDE
SEQ951	MADLPQKVSNLSINNKENGGGGGKS......GTSA NYN**R**GGSS.....HLARD FLDNYIFLSVGRV GSTSENITQRILYVDDMDKKS ALLDLLSAEHKGL TLIFVETKRMADQLTDFLIMQNFKATAIH GDRTQ**A**E RERALSAFKANVA DIVLATAVAARGLDIPNVTHVINYDLPS DIDDYV**H**RIGRTGRAGNT G......NDNEKNGYGNSNASWW
SEQ173	MTTDEGAKNSRGNPAATV AEQGEDVTSKKDRGVLKIVK**R**VG HGEETPMIGDRVYVHYNGKLANGKKFDS SHD**R**NEPFVFSI....GIIR RTK**RRGE**GYSNPNEGARVQ IHLEGRCGGRVFDCR.....IVKEKGTVY FKGGKYVQA**V**IQYGKIVSWLEME YGLSEKESKASESFLLAAFLN**L**AM CYLKL.....KAKEHNERDR**RT**YAN MFKKFAEQDAKEEANKAMSKKTSEGVTNEKLT ASHAVEEEKPEGHV
SEQ862	MNTDQQPYQ.......STG RGFATSRIPFSILYSRFAGSAIYMGARSML MLLFGTVAHWQAPLLWFWASL**SS**L IF**A**PFVFNPHQFAWEDFFLD YRDYIRWLSRGNNQYHRNSWIGYVRMSRA RITGFK**RKLV**GDESEKAAGDA.......LF WLKPSRQIRPPIYSLKQTRLRKRMVKKYCSL YFLVLAIFAGCIIGPAVASAKIHKHIGD SLDGVVHNLFQPINTTNNDTGSQMS TYQSHYYTHTPSLKTWSTIK
SEQ224	**ME**GEELIYHNIINEIL**V**GY IKYYINDISEHELSPYQQQIKKI**L**TYYDECLNK QVTITFSLTSVQEIKTQFTGVVTELFKDLINWGRI CGFIVFSAKMAKYCKDANNHLESTVITTAYNFMK HNLLPW**M**I**S**HGGQEEF LAFSLHSDMYSVIFNIKYFLSKFCNHMFF R**S**CVQ**LLR**NCNLI