Machine Learning of Amino Acid Composition Models for Protein Redesign

Sijia Xiao

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Chemical Engineering

University of Washington

2019

Reading Committee:

François Baneyx

David Beck

Program Authorized to Offer Degree:

Chemical Engineering

University of Washington

Abstract

Machine Learning of Amino Acid Composition Models for Protein Redesign

Sijia Xiao

Chair of the Supervisory Committee:
François Baneyx
David A.C. Beck
Chemical Engineering

Proteins from thermophiles can preserve their basic structures and original functions at high temperature. However, most of the mesophilic proteins are vulnerable to such extreme condition due to their different amino acid composition. Improve the thermostability of thermophilic proteins will reduce the cost of from storage and production in industrial process. Nowadays, machine learning becomes a powerful method for data-intensive computation. This work provides a well-tuned network, called Thermalizer, which applies Recurrent Neural Network (RNN) to encode the mesophilic proteins to thermostable proteins which is predicted to be able to perform expected function in higher temperature. The toolkit also provides workflow from gene and amino acid sequence preprocessing to encoder and decoder construction, model training and evaluation, and translation window. The project is accessible on GitHub: https://github.com/BeckResearchLab/thermalizer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. INTRODUCTION

1.1 THERMOSTABILITY

For a long time, researchers are curious about extremophiles and their living strategy to thrive in the harsh environment. For example, Halophilic Bacteria can maintain a low ionic concentration in their cells and adapt to high salt concentrations.[1] Thermophile, invulnerable to high temperature, is another kind of organisms that researchers are obsessed with. In general, mesophile has optimum growth temperature (OGT) in the range of 15°C to 45°C, while thermophiles can survive at the temperature up to 80°C.[2] Besides, hyperthermophile and psychrophile are two other genera that can survive and propagate in extremely high and low temperature.

Amino acids are the building blocks for protein sequence, which contain all of the information to perform certain functions in metabolic reaction, signaling interaction and structure maintenance.[3] There are few amino acid residues interactions that determine the shape of the protein and stabilize its structure, including disulfide bridge, salt bridge, hydrogen bond, Van der Waals interaction and electrostatic interaction.[4] A number of studies revealed that composition of amino acid is biased in proteome for thermophiles and mesophiles. In previous studies, researchers discovered a short amino acid sequence – Ile, Val, Tyr, Trp, Arg, Glu, Leu (IVYWREL) – to predict the OGT for prokaryotes.[5] Thermophilic proteins prevent misassemblies and stabilize structures by increasing the fraction of positively charged residues, such as Gln and Ile on the surface[6,7] and negative charge at the N-terminus[8]. Several substitutions frequently occur to form more salt bridges within the protein helix.[8] Thermophilic adaptation of proteins also take advantage of their compact hydrophobic core since hydrophobic effect is one of the primary forces to stabilize protein structure at high temperature.[9] Although thermostability is one of the fundamental and well-studied features,

predicting a unknown sequence or modifying a sequence to maintain a function at high temperature is still cumbersome. Take all complicated interactions into consideration and finding the optimal positions for all amino acid sequences is far beyond human's control.

1.2 MACHINE LEARNING

Machine learning, a rapidly growing discipline, utilizes an array of statistical algorithms to construct an automatically problem-solving system through experience learned from the large dataset. Instead of tackling a specific problem, machine learning algorithms provide a universal framework to "learn, search and predict". By taking advantage of it, machine learning also brings fundamental change in other discipline, such as Economy, Social Science, Biology, etc. Over the years, numbers of projects are promoted to gain insight into formation of diseases like cancer by collecting enormous genomic and proteomic data. Thousands of scientists and researchers contribute to the construction of gene bank and protein bank, which provide reliable datasets for statistical analysis.[10]

In terms of data labels, machine learning algorithms can be classified into supervised learning, unsupervised learning and reinforcement learning. While the last two kinds contain training inputs without labels, supervised learning, including linear regression, logistic regression, decision trees, random forest, support vector machine (SVM), neural networks and Bayesian classifiers, makes predictions from input-output pairs.[11] These methods are applied not only to predict the protein folding patterns[12] and extract factors that determine the protein interactions[13], but also to detect aberrant activation in cancers' metabolic pathway[14] and to accelerate drug discovery process[15]. Although traditional statistic models are ubiquitous and powerful in many fields, their efficiency are greatly limited in terms of high-dimensional data in the raw form[16, 17].

Deep learning is an unconventional branch of machine learning family, which stacks multiple non-linear layers and generates prediction by backpropagate gradient to minimize error.[17] Compared with classical machine learning methods, deep learning can discriminate the high-level features in a more efficient way.[18] Typically, there are two types of deep architectures, convolutional neural network (CNN) and recurrent neural networks (RNN). While CNN is designed for multi-dimensional arrays, like image pixels, RNN is useful for one-dimensional sequence such as human language analysis. Protein translation is highly analogous to language processing to a great extent: amino acids can be viewed as the words to construct a sentence and rules, such as amino acid frequency, become the "grammar". Human language is to formulate a legal sentence that indicates a specific meaning, while amino acid processing is to generate a protein that can perform a unique function. On the basis of this idea, RNN is adopted to deal with sequence to sequence learning. RNN takes one word at a time and introduced a hidden state flows to the end which memories previous update.[17] In proteomic and genomic research, RNN is especially powerful because of the linear representation of compositional units, such as amino acids and nucleotides. Researchers release an RNN model – ProLanGO[19] – to predict protein function based on the amino acid sequence along. Also, RNN has been applied in gene regulatory networks construction.[20]

The essential part of the research to improve thermostability is to discriminate the composition of thermophilic protein with its mesophilic counterpart. In previous work, researchers attempt to use Bayes rules, neural network, SVM, decision trees and several more for this purpose, and the best they achieve is through neural network by an accuracy over 90%.[21] In this work, we developed a neural network toolkit, called **Thermalizer**, to estimate positions in mesophilic proteins which can be replaced by other amino acids and improve their thermostability. In general, the work contains

several steps: data preprocessing for thermophilic-mesophilic pairs, model construction including encoder, decoder and translation process and hyperparameters tuning.

Chapter 2. MATERIALS & METHODS

2.1 Data Collection and Preprocessing

2.1.1 *Dataset Collection*

The complete genome of bacteria and archaea in this work was collected from National Center Biotechnology Information (NCBI). Although NCBI assembles few optimal growth temperature (OGT) records, this work also collects OGT from other reliable source. The Bacterial Diversity Metadatabase, Bac*Dive* (https://bacdive.dsmz.de/), provides cultivation information including OGT for 63,669 bacterial and archaeal strains.[22] Thus, Bac*Dive* is the primary platform for OGT collection. Moreover, some organisms in Bac*Dive* have already classified as thermophiles and mesophiles while others don't. For simplicity, we defined that organisms with OGT greater than 50°C classified as thermophiles and organisms with OGT less than 50°C as mesophiles. Any organisms with OGT in between should be ignored in order to have two clearly classified datasets. In total, there are 5675 organisms from NCBI and around 2000 OGT records from Bac*Dive*. Those organisms we cannot acquire the OGT will be deleted from the dataset. Each genome translates into groups of protein sequences.

For the purpose of this research, mesophilic protein will be used as input and thermophilic protein will be used as responses in the training set. In order to guarantee that the model is capturing features that affect its thermostability, other functions of the protein should keep the same. First, we use Basic Local Alignment Search Tool (BLAST) to compare the 16S ribosomal RNA (rRNA) nucleotide similarity among all genomes. 16S rRNA, the 30S small subunit of a prokaryotic ribosome that binds to SD sequence, is inferred as an accurate label to reveal prokaryotes evolution and to identify and classify Bacteria and Archaea.[23] The alignment reduced the original dataset

into 214 mesophilic-thermophilic genome pairs, which have high homology. For each pair, we assemble proteins from the genome and use BLAST to pair protein sequences, which serves as the input (mesophilic) and response (thermophilic) for the training process.

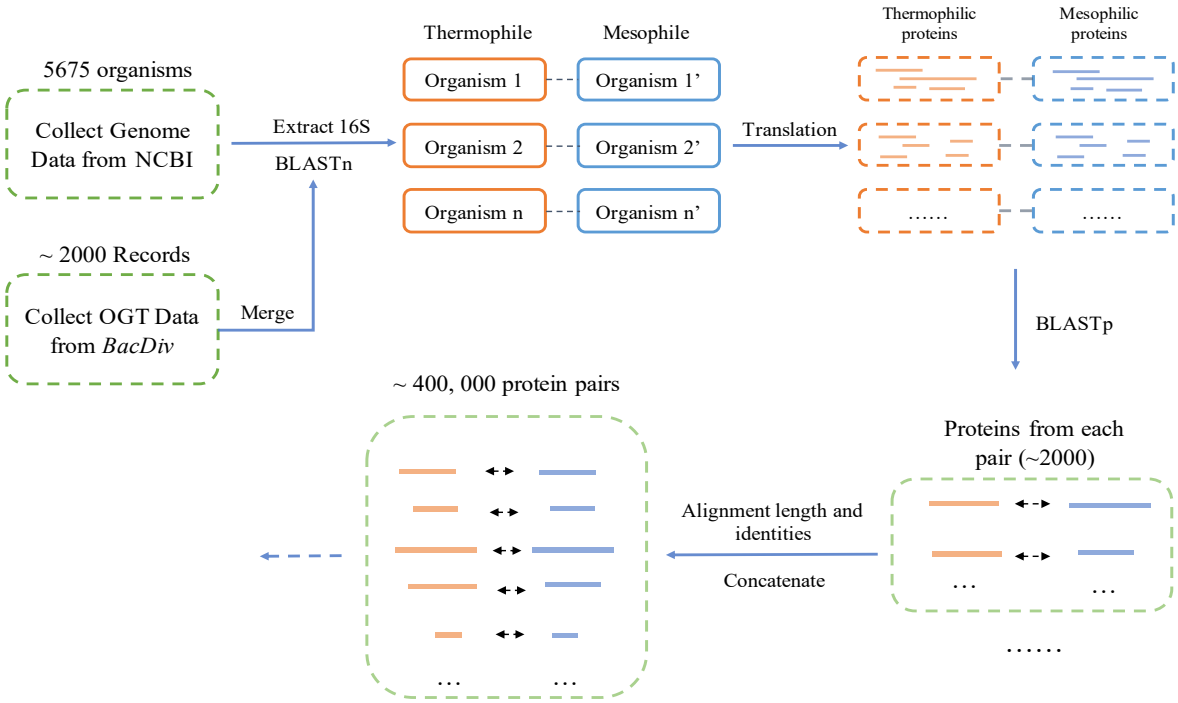


Figure 1. Workflow for Data Preprocessing

2.1.2 *Dataset Partition*

Typically, dataset should be partitioned into training set, validation set and test set, and each set should have input-target pairs. The input and target from training set will be feed to the Encoder and Decoder of the network respectively and get the set of weights representing the rule for prediction. The validation set will be used to evaluate the model as training process continues. The loss value or accuracy from the validation set will be taken as reference to determine the best configuration of hyperparameters and set the early stopping. The test set will not be used to update

the weights, in other word, remain untouched in the training process. It will be used in the final assessment of the model. In this work, training set will take 80% of the data, validation set will be 10% and test set will also be 10%.

2.1.3 *Sequence Preparation*

Before feeding training network, data should be converted into numerical form that computers can manipulate. The natural way to deal with it in classification problem will be one-hot encoding. The vector has the same size as the number of categories. It will be filled by zero, except for the category the target belongs to, which is one. However, in language processing, the vector can be unreliable since some words have similar meaning. For example, “big” and “large” should be classified in the same category, instead of two categories in one-hot vector. We use word embedding in this work. Each character in the protein sequence will be represented by a unique index. The protein sequence should be padded into the same length with a special character added to the end of the sentence. Each index will be then replaced by a random vector of continuous number.

2.2 MODEL CONSTRUCTION

2.2.1 *Encoder and Decoder*

Model construction can be separated into two sections: Encoder and Decode. (Shown in Figure 2) Encoder takes a batch of embedded symbols to a fixed-length vector[24] and trained forward into a final hidden state which consists of the estimated grammar logic of those sequences. The Decoder, on the other hand, accepts the final hidden state of the Encoder as the initial state and predicts a sequence to maximize the probability distribution at each timestep.[24] The efficiency of traditional

RNN are limited because of the exploding or vanishing gradient and gradient clipping. There are two types of RNN are designed to improve the ability of memorizing context from far away: long short-term memory cell (LSTM) and gated recurrent unit cell (GRU). (As shown in Table 1) LSTM consists of three gates (forget, input and output) to control the percentage of current input memorized by the network.[25] It also separates the process to update memory and hidden states. However, the GRU adopts a different strategy where it only has reset and update gates, and it combines the memory and hidden states flow.[26] Although LSTM and GRU have slightly different structure, it remains problematic to answer which one has better performance. Thus, in this work, we will try both LSTM and GRU units and compare them with each other.

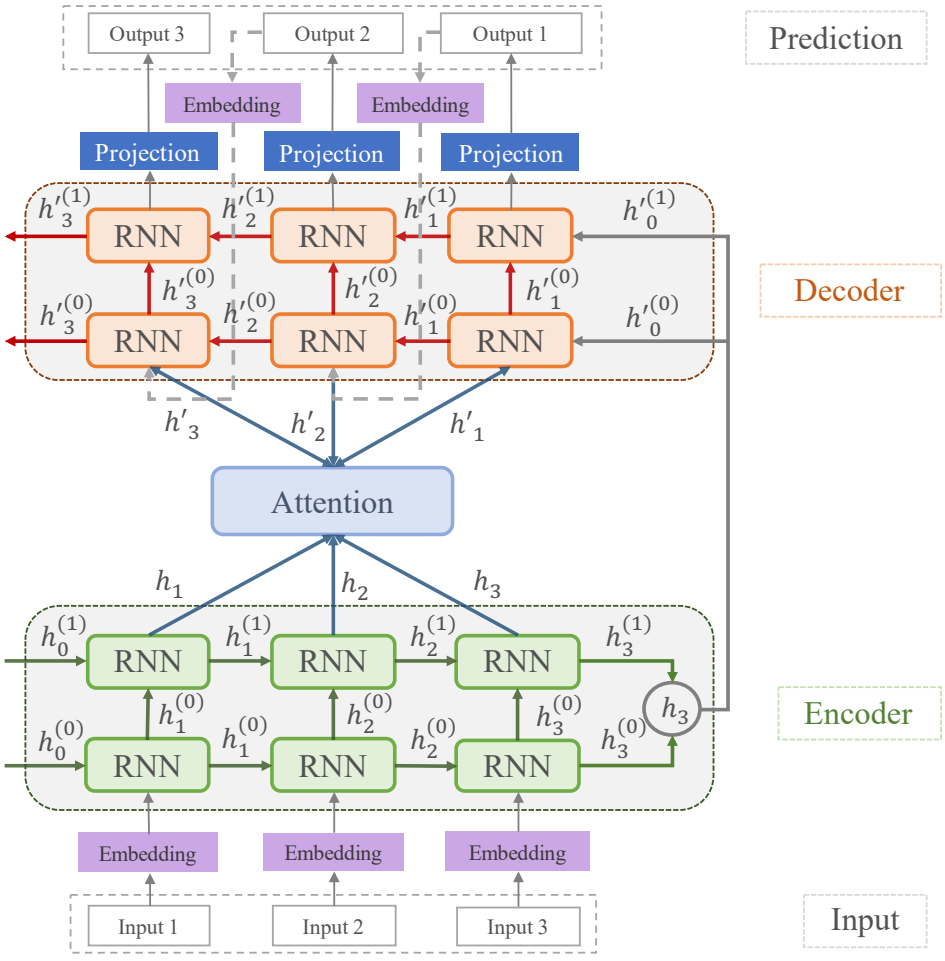


Figure 2. Encoder and Decoder System

Table 1. LSTM and GRU Updates at Each Timestep[25, 26]

LSTM	GRU
General form: $\boldsymbol{h_t = f(W \cdot h_{t-1} + W \cdot x_t)}$	
Forget gate: $\boldsymbol{f_t = \sigma(W_{hf} \cdot h_{t-1} + W_{xf} \cdot x_t)}$ Input gate: $\boldsymbol{i_t = \sigma(W_{hi} \cdot h_{t-1} + W_{xi} \cdot x_t)}$ Output gate: $\boldsymbol{o_t = \sigma(W_{ho} \cdot h_{t-1} + W_{xo} \cdot x_t)}$	Reset gate: $r_t = \sigma(W_{hr} \cdot h_{t-1} + W_{xr} \cdot x_t)$ Update gate: $z_t = \sigma(W_{hz} \cdot h_{t-1} + W_{xz} \cdot x_t)$
Update memory: $\boldsymbol{\tilde{c}_t = \tanh(W_{hf} \cdot h_{t-1} + W_{xf} \cdot x_t)}$ $\boldsymbol{c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t}$ Update hidden state: $\boldsymbol{h_t = o_t \circ \tanh(c_t)}$	Update memory: N/A Update hidden state: $\tilde{h}_t = \tanh(r_t \circ (W_{hn} \cdot h_{t-1}) + W_{xn} \cdot x_t)$ $h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1}$

2.2.2 *Attention Mechanism*

In classical Encoder-Decoder network, the initial state of Decoder is merely from the last state of the Encoder. It is problematic since the Encoder attempts to capture the meaning of the context using the last matrix. To improve it, researchers developed a mechanism called "Attention" which uses an extra weight matrix to map query to output.[27] It can, therefore, pay attention to a certain region that is crucial for Decoder to predict the next word or letter. The attention mechanism in this work applies global attention which has three alternative forms[28] (as shown in Figure 3): dot product (a_t) of overall Encoder hidden states (h_s) and current Decoder hidden states (h_t), cross product (c_t) of h_s and a_t, and concatenation of h_t and c_t. And we will try all three forms and pick the best one for our model.

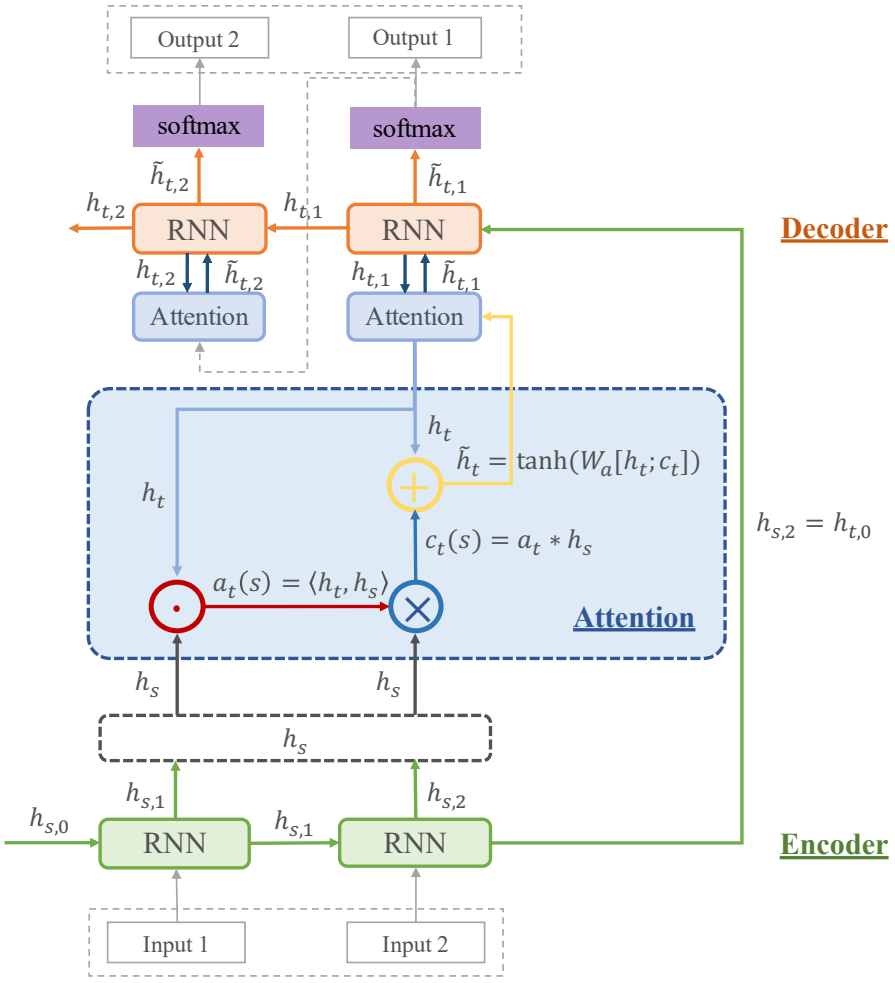


Figure 3. Attention Mechanism

2.3 MODEL TRAINING

2.3.1 *Compiling*

The model will initially be trained on CPU with configuration - Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz. The training will be shifted to Tesla P100 GPU which has compute capability 6.0. To compile in the GPU nodes, we need to move variables and parameters into GPU. In this work, we will use built-in packages from Pytorch, which provides a very easy and convenient way to transfer data from CPU to GPU or vice versa.

2.3.2 *Batching*

The model will train on a total of 401,885 unique mesophilic-thermophilic sequence pairs with 144,836 unique mesophilic proteins and 381,838 unique thermophilic proteins. The maximal (minimal) length in Encoder and Decoder are 14,039 (37) and 11,787 (14) respectively. Therefore, some of the thermophilic proteins can pair with the same mesophilic protein. In RNN model, the number of timesteps has to be equal to the sequence length. Therefore, the dimension will be tremendous if padding all sequences to the maximum sequence length. In this work, we cut down the protein sequence length to be 200.

2.3.3 *Hyperparameter Tuning*

The performance of a deep learning model can be improved by increasing the size of training dataset, constructing a more efficient architecture and setting the optimal hyperparameters. Unlike the previous two strategies, hyperparameter tuning is empirical and is cumbersome to train. The most intuitive way to find the best hyperparameter setting is grid search which exhaustively searches all possible hyperparameter combinations and pick the one with highest accuracy. This method is practical only if few hyperparameters are in the pool. Randomized search,[29] on the other hand, provides an efficient way to search in a large hyperparameter grid. However, it does not necessarily lead to the optimal setting. In this work, we adopt the strategy in between: change the hyperparameter in one kind and use the best one to adjust the other hyperparameters. Also, hyperparameters affect the model by changing the dimension of matrix that has been used to store the learning process such as weights. In neural networks, the change of dimension is often the most confusing part and contributes to a huge number of bugs. Therefore, we also provide a dimension overview for the model and reveal how it changes through training process.

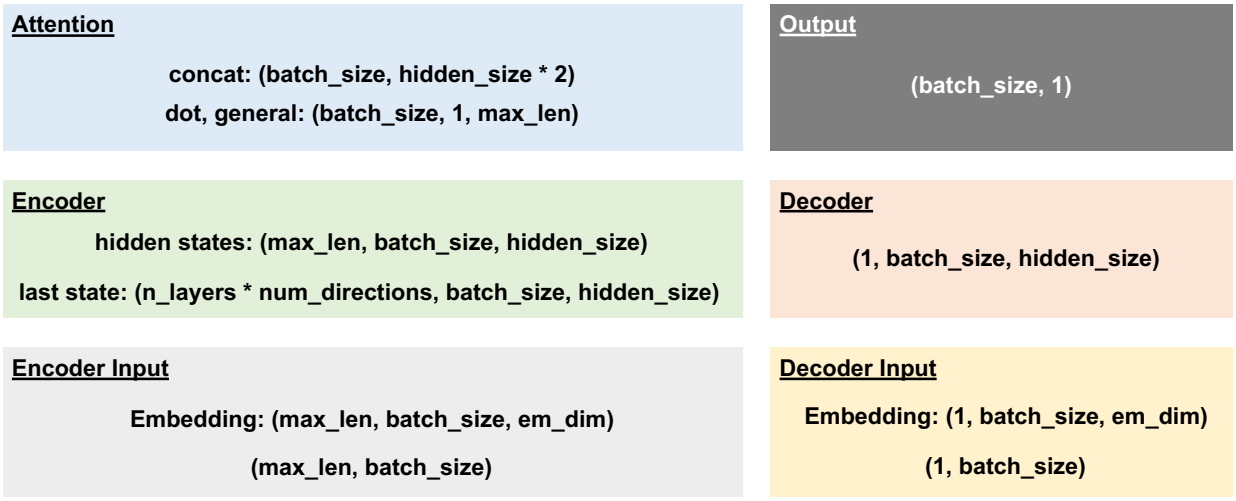


Figure 4. Dimension Overview of the Model

Maximum sequence length (max_len): the sequence that is shorter than the maximum sequence will be padded with zero. We set the max_len to be 200 for hyperparameter tuning.

Batch Size (batch_size): batch_size controls the number of samples feeding into the networks for each iteration, where$Iteration = number\ of dataset/batch_size$. Dividing datasets into batches can not only improve the quality of gradient calculation but also more efficient than calculate one by one for multiple times.[30] In our work, we will try 64, 128, 256 samples for each batch.

Embedding Size (em_dim): word embedding is to represent words with a dense vector, which is powerful to catch the words' similarity using a low-dimension matrix.[31] Embedding size is determined by the vocabulary size and hidden dimension, which is not really a hyperparameter.

Number of Layers (n_layer): number of layers is determined by how complex the classification problem is.[32] In this work, the layers of encoder and decoder will set to be equal and the candidate layer number will be 2, 3 and 4.

Hidden Unit (hidden_size): the size of hidden units controls the nonlinearity or complexity of each layer by introducing activation function. Possible choices are 300, 400 and 500.

Dropout: dropout is a tool to reduce overfitting, which randomly removes some of hidden units from each layer.[33] The dropout rate will be taken in the range of [0.1, 0.3, 0.5].

Learning Rate: The essential part of neural network is the optimizer, in which learning rate controls the amount of parameter it updates during training process. If learning rate is too small, the converge will be too slow and computational expensive. In some cases, smaller learning rate will lead to a suboptimal solution. However, if learning rate is too high, the gradient descent will step over the minimum point and also result in a local minimum point. Therefore, we adjust learning rate between 0.001, 0.0005 and 0.0001.

2.3.4 *Evaluation Function*

Model learns from the probability distribution through backpropagation.[26] The sequence to sequence model is basically a classification problem, for which people often use cross-entropy loss and negative log-likelihood loss. However, researchers found that the negative log-likelihood loss performs better than cross-entropy loss for high dimensional dataset.[34] In this work, we will use negative log-likelihood to evaluate the model. Since we padded all sequence length to be the same, the loss will be overestimated if we take into account the padded part also. Therefore, we will use a vector to store the original sequence length (mask), and only use original length to calculate the loss.

$$NLL = -\frac{1}{N}\sum_{i \in mask} \log\left(\frac{\exp(z_i)}{\sum \exp\,(z_j)}\right)$$

Chapter 3. RESULT & DISCUSSION

3.1 AMINO ACID SEQUENCE ANALYSIS

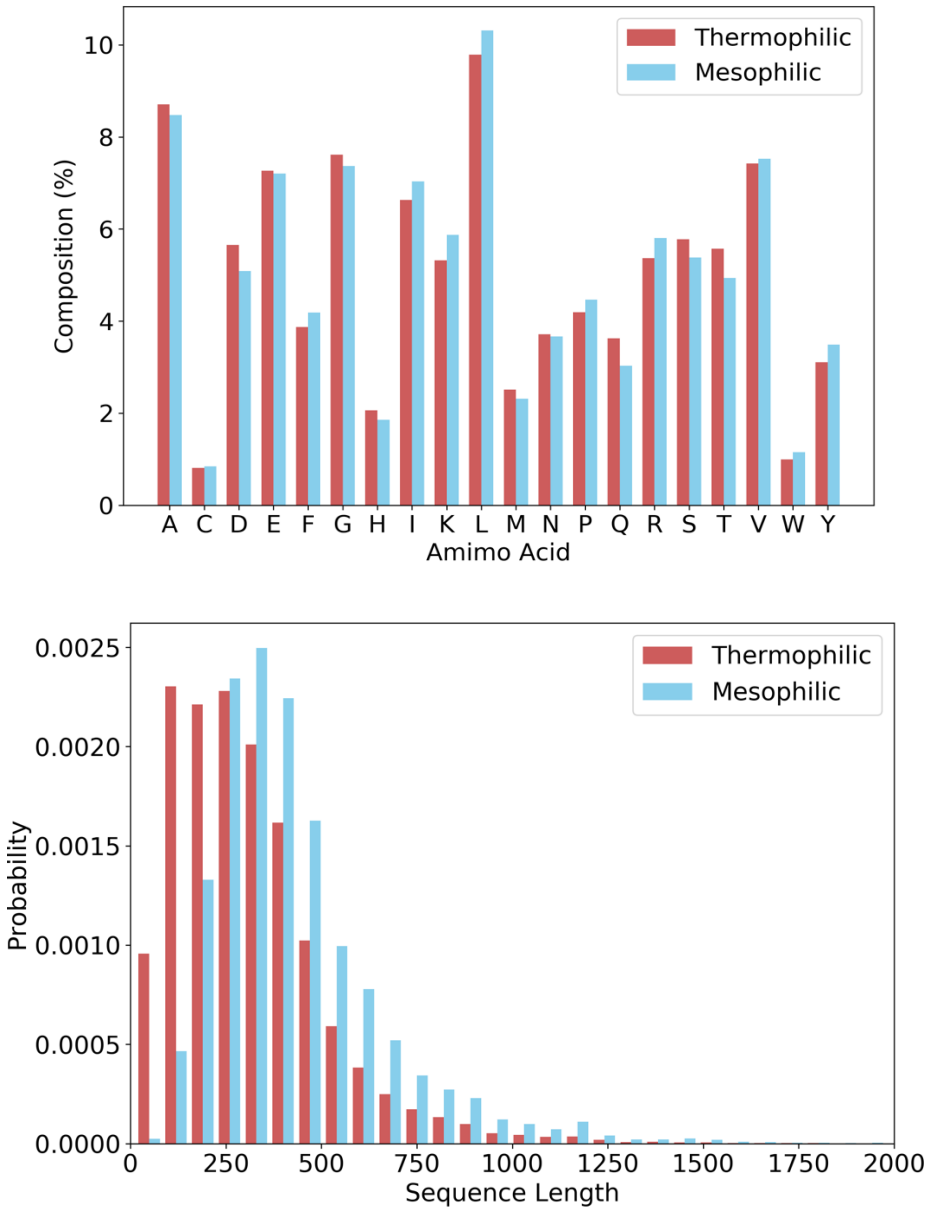


Figure 5. Sequence Analysis

Note: A~Y are one letter abbreviation for amino acids. X represents the unknown amino acid.

As the first step, we analyze the amino acid composition across all thermophilic proteins and mesophilic proteins (as shown in figure 4A). In general, these two kinds of proteins have very similar amino acid composition. However, thermophilic proteins have slightly higher Ala (A), Asp (D), Gly (G), His (H), Met (M), Gln (Q), Ser (S) and Thr (T) than mesophilic proteins. Previous research found that acidic residues such as D, K, N, A and T contribute to the OGT variance of mesophiles and thermophiles.[35] Also, increasing hydrophobic residues like A, L, M and V, and positive charged residue like H will improve thermostability of proteins.[9] As illustrated in Figure 4B, the average thermophilic proteins tend to be shorter than mesophilic ones, which is probably because proteins from bacteria are generally longer[36], however, most of thermophiles are from archaea. Although the amino acid composition reveals some elementary rules to make thermophilic proteins, the interactions of these amino acids are too complicated to be concluded in the composition analysis. Therefore, we need machine learning that can learn from the data with less human interference and predict in a less descriptive way.

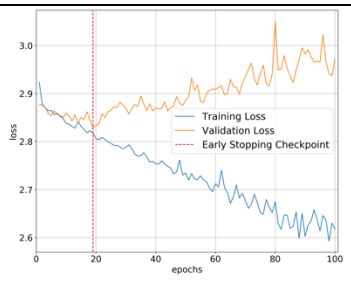
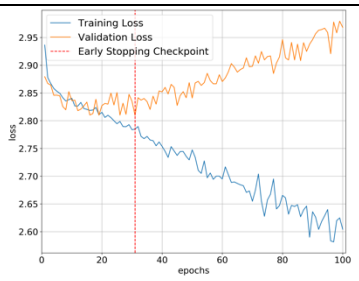
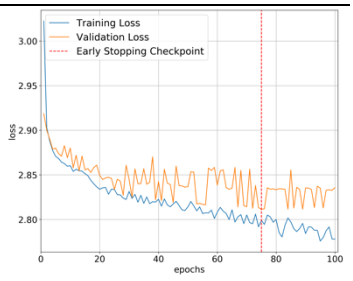
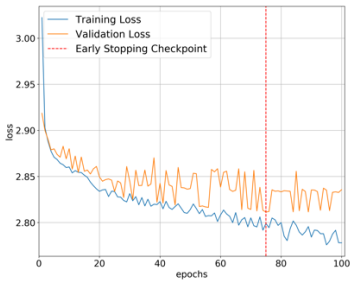
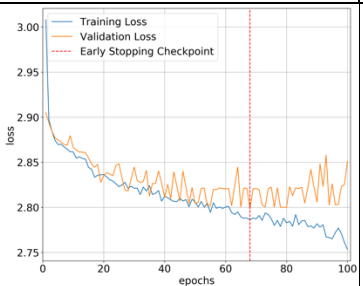
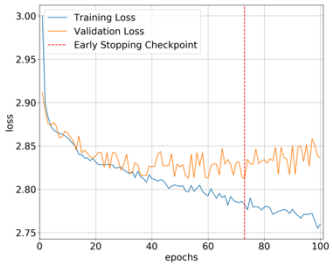
3.2 HYPERPARAMETER TUNING

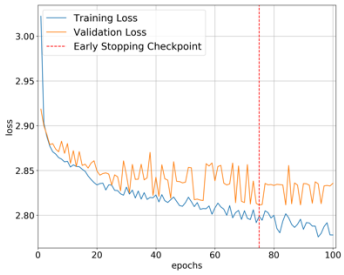
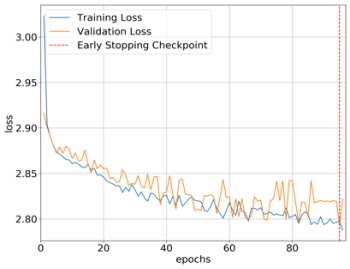
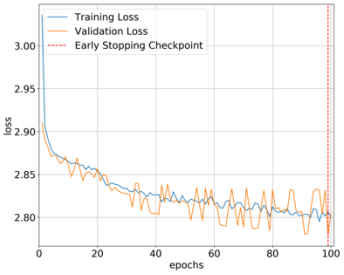
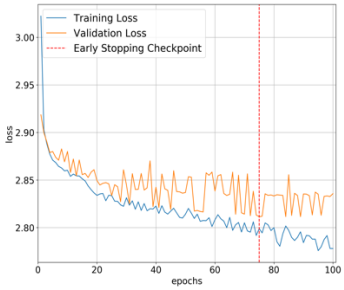
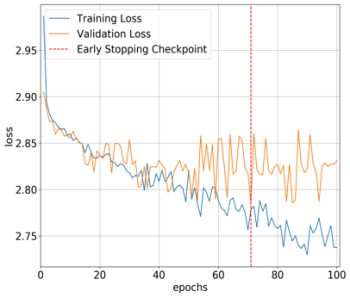
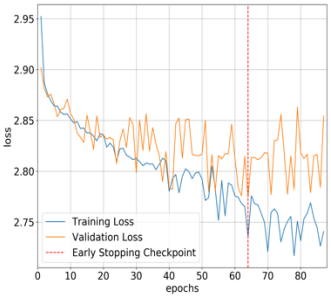
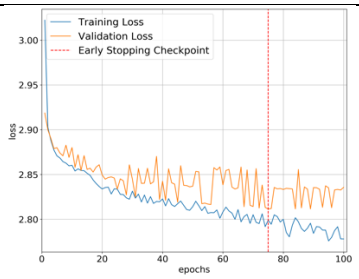
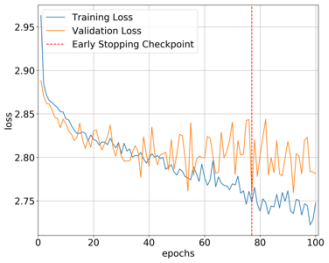
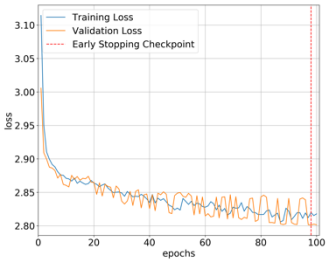
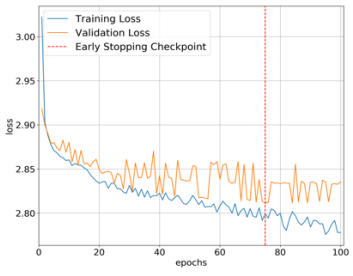
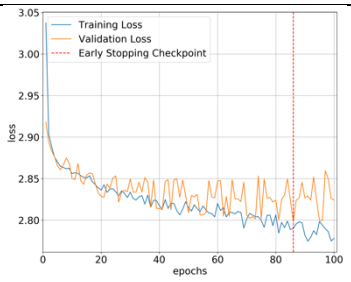
We trained the model with default setting (2 layers, 0.1 dropout, 0.001 learning rate, 128 batch size, 300 hidden size) in 100 epochs. The protein lengths are limited under 200 amino acids and the size of dataset is 3,600. As shown in Table 1, we plot learning curve for different settings to observe the dynamic performance of each model. In general, all training curves have similar shape since the model is adjusting the weights to minimize the loss function, negative log likelihood in this work. Therefore, the loss value continuously decreases while training, which explains why the training loss in all plots are dropping steadily. However, validation loss often drops at first and increases afterwards due to overfitting.

3.2.1 *Learning Rate*

In default setting, the validation loss drops at first 20 epochs and starts to increase steadily afterwards. It exceeds the original loss value after few epochs which suggests an early overfitting. The inference result in this setting does not perform well, where few characters repetitively appear continuously in the sequence. The most important hyperparameter to greatly improve the performance, in this case, is to decrease the learning rate.[37] Learning rate determines how fast the model is converging. As shown in the table 2, the model performs poorly with 0.001 and 0.0005 learning rate. The reason is that the gradient descent so fast that it actually misses the optimal point and gets stuck into a suboptimal solution. However, decreasing the learning rate can make the model converge slowly and reach the possible global minimum. However, if the learning rate is too small, not only will the training process become slow, it will also be trapped into a solution that is not optimal.

Table 2. Result for hyperparameter tuning

	Default Setting	Setting 1	Setting 2
Learning rate	 learning_rate = 0.001	 learning_rate = 0.0005	 **learning_rate = 0.0001**
Number of Layers	 **n_layers = 2**	 n_layers = 3	 n_layers = 4

Dropout	 **dropout = 0.1**	 dropout = 0.3	 dropout = 0.5
Hidden size	 **hidden_dim = 300**	 hidden_dim = 400	 hidden_dim = 500
Batch size	 **batch_size = 128**	 batch_size = 64	 batch_size = 256
Attention Mech.	 **general**	 dot	CUDA out of Memory concat

Note: First column is the default setting and settings in Bold are optimal.

3.2.2 *Hidden Layer*

We tuned the number of hidden layers with best setting of 0.0001 learning rate. Hidden units, such as GRU cells, is designed to calculate weighted sum and then pass it to a non-linear

activation function. In the sequence model, the Encoder and Decoder requires separated hidden layers. So, when we set the number of layers to be 2, there are 2 hidden layers for Encoder and another 2 for Decoder which is 4 hidden layers in total. Compare the loss curve of layers of 2 with curve of 3 and 4, we notice that the validation loss of former decreases and remain relatively stable. However, validation loss of the last two decreases in at first and increases thereafter, which are represented as overfitting. Since greater number of layers bears a risk of running into a local minimum,[37] we use 2 layers to optimize the model.

3.2.3 *Dropout*

Dropout rate, a major regularization method to improve deep learning networks, randomly drops a fraction of hidden units in training process and recovers them during validation.[38] Notice that inflection is ubiquitous in all current figures, which shows that the model is overfitting. As we add more dropout, validation loss is approaching closer to the training loss. However, dropout rate of 0.1 performs better when we sample inference on test set. It is possible that the dataset we are training are too small.

3.2.4 *Hidden Units*

Apart from learning rate, the size of hidden units is another important hyperparameter needed to be chosen carefully. Unlike learning rate which may result in the suboptimal solution and much worse performance, increasing the hidden size can actually prevent the model relying much on other regularization methods such as dropout.[39] As we increase the number of hidden units (hidden_dim), both of training loss and validation loss become more fluctuating. The noisy learning curve suggests that the mini batch cannot accurately represent the overall performance of the model under current learning rate. Although the model with 400 hidden units is noisier than

that with 300 units, the performance of these two settings are comparable. However, greater hidden size requires more computational space and parallelizing time.[40] In our work, memory required by the model with 500 hidden units exceeds the maximum computing capacity of Nvidia CUDA. Therefore, the setting with 300 hidden units is adopted for other hyperparameter tuning.

3.2.5 *Batch Size*

As we increase the number of batch size, the validation loss curve becomes less noisy. Previous research found that large batch size often results in overfitting while small batch size has better performance in generalization.[41] It is consistent with our result that the fluctuated validation loss is very close to training loss with 256 samples per mini-batch. In our inference test, the prediction of the setting is much worse than that with 128 samples in a batch. However, if the batch size is 64, the curve is too noisy which indicates that the model has been generalized too much so that it did not learn a lot from the inputs.

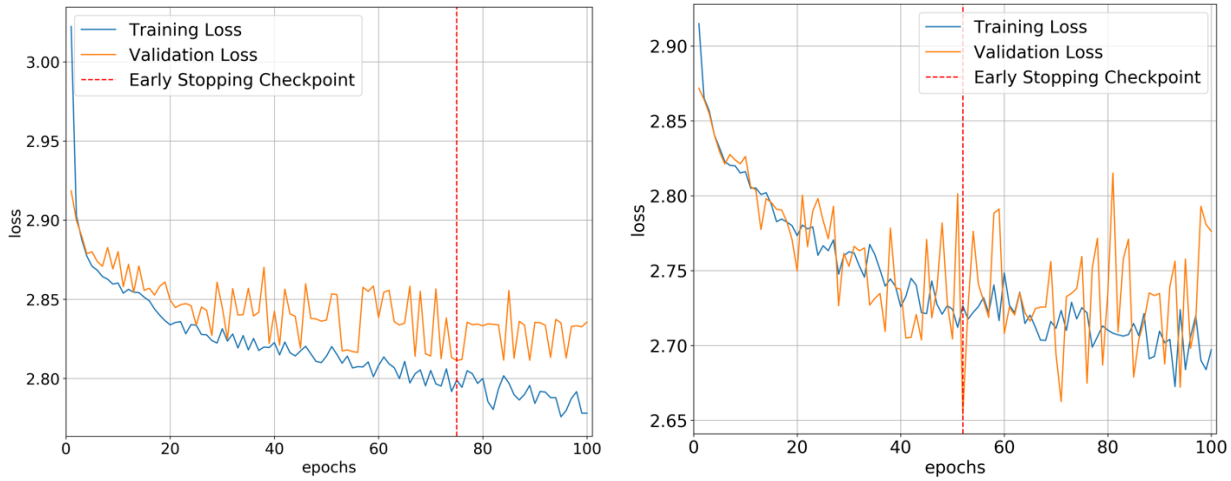
3.2.6 *Attention Mechanism*

We tried the dot product (dot), cross product (general) and concat product (concat) in our setting. The general method is slightly better than dot product in the later testing, which can be confirmed from the figure that the general method becomes more stable over the time than dot product does. However, we failed to train using the concat product, which requires much more internal memory in GPU. When we implement the dot method in PyTorch, we simply sum across two hidden states, which is the lightest one among these three methods. In general mechanism, however, the hidden states of the Encoder will flow through a linear layer which contains an additional parameter with dimension (hidden size by hidden size) that tracks the attention energy

in the mechanism. Unlike the previous two, concat product requires much more memory since the computation will double the weights matrix. In this case, general method is the best we can choose.

3.3 Performance with Large Dataset

In order to generalize the model, the dataset should contain sufficient observations.[42] Thus, we test the performance of the model with a larger dataset: 15,000 samples with maximal 200 amino acids. The result is shown in Figure 6. As we increase the size of the dataset, the fluctuation of validation curve increases as well, which indicates that the descent is overshooting the local minima under current hyperparameter setting. Also, we notice that the average loss for larger dataset is smaller than that for smaller dataset. We assume that the larger dataset should have a better performance. However, the loss curve cannot provide enough information to estimate the model's ability of prediction.



size: 3600 size: 15,000

Figure 6. Learning curve with different size of training set

Therefore, we sampled one mesophilic protein and show its thermophilic predictions through training epochs (shown in Table 3). We observe that the model's prediction in first 200 epochs

consists of a lot of meaningless sequence fractions. As we are training for more epochs, the length of the predicted sequence gets closer to the target's sequence length. The sequence does not contain the repetitive fractions in the prediction anymore after 300 epochs. To get a better understanding for the prediction, we use local alignment tool, BLAST, to compare the target sequence (query) with the predicted sequence (subject) and the original input (As shown in Figure 7). The positive sign means that the two amino acids have similar physico-chemical properties. From the figure, we can see that the prediction has only 38% identities and 64% positives with target sequence. However, it shares the most similarity with input sequence. In this case, the model does not perform well to predict thermophilic proteins based on mesophilic sequence. Although the model captures some conservative part, it predicts poorly on the key amino acid residues that may improve thermostability of mesophilic proteins.

Table 3. The change of inference over time

	Sequence
Input	MGIKIIATNKKAYHDYFIDEVYEAGIALKGSEVKSIREGKVNLKEAFCRIKNGEVFIHNMNIS PYEFASQEAVDPVRMRKLLLHREEIAKLSRKIDEKGFSLVPTKIYFKNSRAKLELGVARGKK LYDKRETLKRKQHDRETARAIREHHK
Epoch 100	MKIKIIATNKKKAYHDYYIDEVYEAGIALKGEEVKKIKEGKVNLKELLEELLKEILEIIEIIIE IEIELLLLLLLLLLLLLLLLLLLLLLKKKLLKKKKKKKKKKKKKKKKKKKKKKKKRRRR RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Epoch 200	MKIIATTNKKAYYYYIDEVYEAGIALKGKEVKSIKEGKVKLKEALKELEEIEEIEIEIEIEEEE EEEEEEEELLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLKKKKKKKKKKKKKKKR RRRRRRRRRRRRRRRREEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
Epoch 300	MKIIATNTKKAYHYYIDEVYEAGIALKGSEVKSIREGKVNLKEAFKKGEVLIHNINISPYEFA SKEAIDPVRLRKLLLHREIAKLLKHREIAKLSRKLLNKLDEKGFSLVPTKIYFKNSRAKLLEI GIRGRGLYDRRRKKIELRRKEELERRREIRERKRKNEIEER
Target	MGRQVKVVTDNRRARHDYFIEETYEAGIALTGTEVKSLRNGRANIKDSYARVENGELILH DMHISPYEQGNRFNHEPRRPRRLLMHRYEIQRLYGKVREKGLTLIPLKVYFNERGLAKVE LALVKGKRLYDKREDIAARDAQREIARALRERQKV

A

Range 1: 1 to 137 Graphics Next Match Previous Match

Score	Expect	Method	Identities	Positives	Gaps
191 bits(484)	5e-68	Compositional matrix adjust.	109/142(77%)	120/142(84%)	19/142(13%)

```
Query  3    IKIIATN-KKAYHDYFIDEVYEAGIALKGSEVKSIREGKVNLKEAFCRIKNGEVFIHNMN  61
            +KIIATN KKAYH Y+IDEVYEAGIALKGSEVKSIREGKVNLKEAF   K GEV IHN+N
Sbjct  1    MKIIATNTKKAYH-YYIDEVYEAGIALKGSEVKSIREGKVNLKEAF---KKGEVLIHNIN  56

Query  62   ISPYEFASQEAVDPVRMRKLLLHRE--------EIAKLSR----KIDEKGFSLVPTKIYF  109
            ISPYEFAS+EA+DPVR+RKLLLHRE        EIAKLSR    K+DEKGFSLVPTKIYF
Sbjct  57   ISPYEFASKEAIDPVRLRKLLLHREIAKLLKHREIAKLSRKLLNKLDEKGFSLVPTKIYF  116

Query  110  KNSRAK-LELGVARGKKLYDKR  130
            KNSRAK LE+G+ RG+ LYD+R
Sbjct  117  KNSRAKLLEIGI-RGRGLYDRR  137
```

B

Range 1: 1 to 137 Graphics Next Match Previous Match

Score	Expect	Method	Identities	Positives	Gaps
102 bits(253)	5e-33	Compositional matrix adjust.	54/142(38%)	92/142(64%)	18/142(12%)

```
Query  5    VKVVTDNRRARHDYFIEETYEAGIALTGTEVKSLRNGRANIKDSYARVENGELILHDMHI  64
            +K++  N +  + Y+I+E YEAGIAL G+EVKS+R G+ N+K+++ +   GE+++H+++I
Sbjct  1    MKIIATNTKKAYHYYIDEVYEAGIALKGSEVKSIREGKVNLKEAFKK---GEVLIHNINI  57

Query  65   SPYEQGNRFNHEPRRPRRLLMHRYEI-------------QRLYGKVREKGLTLIPLKVYF  111
            SPYE  ++   +P R R+LL+HR EI              ++L  K+ EKG +L+P K+YF
Sbjct  58   SPYEFASKEAIDPVRLRKLLLHR-EIAKLLKHREIAKLSRKLLNKLDEKGFSLVPTKIYF  116

Query  112  NERGLAKVELALVKGKRLYDKR  133
                 +E+  ++G+ LYD+R
Sbjct  117  KNSRAKLLEIG-IRGRGLYDRR  137
```

C

Range 1: 5 to 154 Graphics Next Match Previous Match

Score	Expect	Method	Identities	Positives	Gaps
162 bits(410)	6e-57	Compositional matrix adjust.	77/151(51%)	113/151(74%)	3/151(1%)

```
Query  3    IKIIATNKKAYHDYFIDEVYEAGIALKGSEVKSIREGKVNLKEAFCRIKNGEVFIHNMNI  62
            +K++  N++A HDYFI+E YEAGIAL G+EVKS+R G+ N+K+++ R++NGE+ +H+M+I
Sbjct  5    VKVVTDNRRARHDYFIEETYEAGIALTGTEVKSLRNGRANIKDSYARVENGELILHDMHI  64

Query  63   SPYEFASQEAVDPVRMRKLLLHREEIAKLSRKIDEKGFSLVPTKIYFKNSR--AKLELGV  120
            SPYE  ++   +P R R+LL+HR EI +L  K+ EKG +L+P K+YF N R  AK+EL +
Sbjct  65   SPYEQGNRFNHEPRRPRRLLMHRYEIQRLYGKVREKGLTLIPLKVYF-NERGLAKVELAL  123

Query  121  ARGKKLYDKRETLKRKQHDRETARAIREHHK  151
             +GK+LYDKRE +  +   RE ARA+RE  K
Sbjct  124  VKGKRLYDKREDIAARDAQREIARALRERQK  154
```

Figure 7. BLAST Result for one sample in 300 epochs

A. query: mesophilic input, subject: thermophilic prediction

B. query: expected thermophilic target, subject: thermophilic prediction

C. query: mesophilic input, subject: expected thermophilic target

3.4 Computation Time Analysis

We trained the model with the best hyperparameter setting in both CPU and GPU. Average computing time for 3036 samples with max length of 200 in GPU is 19.21 s and average time in CPU is 431.92 s for each epoch. It indicates that GPU is running approximately 22.5 times faster than CPU. In the dataset, the maximal amino acid sequence length is 14,039 and minimum

sequence length is 37. As we pad all sequence to the same length, the dimension of the input becomes too large, which is expensive in computation. Therefore, we need to balance between computing time and the maximum length.

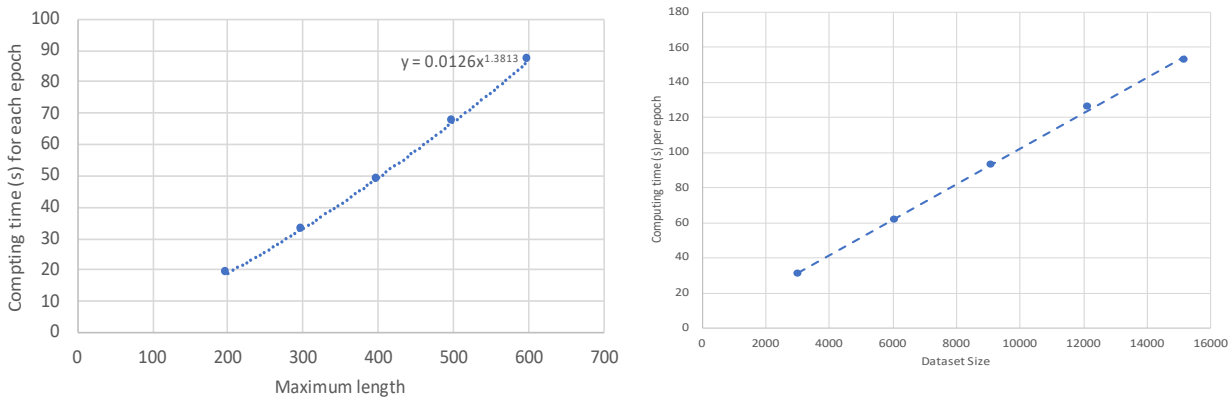


Figure 8. Computational time analysis

Left (A): Increase the max length and keep other parameters unchanged

Right (B): Increase the size of training set with the same hyperparameters.

We test the change of computational time per epoch constricted to optimal hyperparameter setting with data size unchanged as max sequence length increases (Shown in Figure 8A). In general, the computing time increases in polynomial with degree of 1.38. We also test the change of time over dataset size (shown in Figure 8B). Unlike the formal trial, the time cost of computation increases linearly as dataset gets larger. It is because the size of training set only affects the number of batches for each iteration: double the training set will only double the number of mini-batches per epoch. The time for each batch remains the same as before. However, when we change the max length, the dimension of input, hidden states and weights will all be different. Thus, the computational time for each batch will be remarkably different.

Chapter 4. CONCLUSION

4.1 SUMMARY

In this work, we build the Encoder-Decoder with attention mechanism neural network system to modify the mesophilic proteins so that their thermostability can be improved. We let the system to determine the key amino acids that may affect its thermostability and the system can output the target sequence that we might be interested. During the process, we introduced a data preprocessing workflow that can be promising to deal with similar problem. After build the model in PyTorch, we tuned the hyperparameter so that the model can work in its best mode. We tuned the hyperparameter using a small fraction of samples and figure out the best hyperparameter setting is 0.0001 for learning rate, 0.1 for dropout, 2 for hidden layers, 300 for hidden units, 128 for batch size, general for attention. We make predictions using the best setting with greater dataset and analyze its inference result over time. It suggests that as time goes, the prediction gets closer to the target sequence. Later, we did the computational time analysis. We transfer the model from CPU to GPU and improve the calculation efficiency by 22.5 time. Also, if we increase max length and other parameters remain the same, the computational time will increase in polynomial with the degree of 1.4. If we increase the size of training set with hyperparameters unchanged, the computational time will increase linearly.

4.2 FUTURE WORK

So far, we are training with just 5% of the dataset. In order to generate a better result, it is plausible to feed in more samples. Also, table 3 indicates that the model is improving over epochs, which suggests that we need to spend more time training. When tuning hyperparameters, we are using

forward algorithm in which we choose the best setting for one parameter and use this to adjust other hyperparameters. However, this method does not necessarily result in the optimal one. To guarantee the optimal setting, it's better to adopt grid search or random search[29]. Although the default optimizer Adam[43] is sufficiently good in many cases, we can replace it with other optimizers such as RMSprop.

We can also modify the architecture. Researchers found that RNN model can incorporated with the pyramid structure from convolutional neural network (CNN) which reduces hidden units, layer by layer.[44] In this way, we can accelerate the computation without sacrificing the performance.[45] We can also try LSTM cells instead of GRU cells in the model. In the training process, we are padding zeros to the tail in order to maintain the fixed-length batches. However, we can also add gaps during alignment. In this case, the difference between two sequences becomes clearer.

BIBLIOGRAPHY

[1] Ventosa, A et al. “Biology of moderately halophilic aerobic bacteria.” *Microbiology and molecular biology reviews: MMBR* vol. 62,2 (1998): 504-44.

[2] Zhou, X-X., et al. "Differences in amino acids composition and coupling patterns between mesophilic and thermophilic proteins." *Amino acids* 34.1 (2008): 25-33.

[3] Marcotte, Edward M., et al. "Detecting protein function and protein-protein interactions from genome sequences." *Science* 285.5428 (1999): 751-753.

[4] Petsko, Gregory A., and Dagmar Ringe. *Protein structure and function*. New Science Press, 2004.

[5] Zeldovich, Konstantin B., Igor N. Berezovsky, and Eugene I. Shakhnovich. "Protein and DNA sequence determinants of thermophilic adaptation." *PLoS computational biology* 3.1 (2007): e5.

[6] Ma, Bin-Guang, Alexander Goncearenco, and Igor N. Berezovsky. "Thermophilic adaptation of protein complexes inferred from proteomic homology modeling." *Structure* 18.7 (2010): 819-828.

[7] Mizuguchi, K., M. Sele, and Maria Vittoria Cubellis. "Environment specific substitution tables for thermophilic proteins." *BMC bioinformatics* 8.1 (2007): S15.

[8] Chakravarty, Suvobrata, and Raghavan Varadarajan. "Elucidation of factors responsible for enhanced thermal stability of proteins: a structural genomics-based study." *Biochemistry* 41.25 (2002): 8152-8161.

[9] Reed, Christopher J., et al. "Protein adaptations in archaeal extremophiles." *Archaea* 2013 (2013).

[10] Camacho, Diogo M., et al. "Next-generation machine learning for biological networks." *Cell* 173.7 (2018): 1581-1592.

[11] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." *Science* 349.6245 (2015): 255-260.

[12] Cheng, Jianlin, and Pierre Baldi. "A machine learning information retrieval approach to protein fold recognition." *Bioinformatics* 22.12 (2006): 1456-1463.

[13] Zhou, Chang, et al. "Multi-scale encoding of amino acid sequences for predicting protein interactions using gradient boosting decision tree." *PloS one* 12.8 (2017): e0181426.

[14] Way, Gregory P., et al. "Machine learning detects pan-cancer ras pathway activation in the cancer genome atlas." *Cell reports* 23.1 (2018): 172-180.

[15] Ekins, Sean, et al. "Machine learning models and pathway genome data base for Trypanosoma cruzi drug discovery." *PLoS neglected tropical diseases* 9.6 (2015): e0003878.

[16] Zhang, Lu, et al. "From machine learning to deep learning: progress in machine intelligence for rational drug discovery." *Drug discovery today* 22.11 (2017): 1680-1685.

[17] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436.

[18] Angermueller, Christof, et al. "Deep learning for computational biology." *Molecular systems biology* 12.7 (2016): 878.

[19] Cao, Renzhi, et al. "ProLanGO: protein function prediction using neural machine translation based on a recurrent neural network." *Molecules* 22.10 (2017): 1732.

[20] Raza, Khalid, and Mansaf Alam. "Recurrent neural network-based hybrid model for reconstructing gene regulatory network." *Computational biology and chemistry* 64 (2016): 322-334.

[21] Gromiha, M. Michael, and M. Xavier Suresh. "Discrimination of mesophilic and thermophilic proteins using machine learning algorithms." *Proteins: Structure, Function, and Bioinformatics* 70.4 (2008): 1274-1279.

[22] Reimer, Lorenz Christian, et al. "Bac Dive in 2019: bacterial phenotypic data for High-throughput biodiversity analysis." *Nucleic acids research* 47.D1 (2018): D631-D636.

[23] DeSantis, Todd Z., et al. "Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB." *Appl. Environ. Microbiol*. 72.7 (2006): 5069-5072.

[24] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv*:1406.1078 (2014).

[25] Cheng, Jianpeng, Li Dong, and Mirella Lapata. "Long short-term memory-networks for machine reading." *arXiv preprint arXiv*:1601.06733 (2016).

[26] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv*:1412.3555 (2014).

[27] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

[28] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

[29] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

[30] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv*:1502.03167 (2015).

[31] Levy, Omer, and Yoav Goldberg. "Dependency-based word embeddings." *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).* Vol. 2. 2014.

[32] Hinton, Geoffrey E. "Learning multiple layers of representation." *Trends in cognitive sciences* 11.10 (2007): 428-434.

[33] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[34] Zhu, Donglai, et al. "Negative Log Likelihood Ratio Loss for Deep Neural Network Classification." *arXiv preprint arXiv*:1804.10690 (2018).

[35] Fukuchi, Satoshi, et al. "Unique amino acid composition of proteins in halophilic bacteria." *Journal of molecular biology* 327.2 (2003): 347-357.

[36] Brocchieri, Luciano, and Samuel Karlin. "Protein length in eukaryotic and prokaryotic proteomes." *Nucleic acids research* 33.10 (2005): 3390-3400.

[37] Panchal, Gaurang, et al. "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers." *International Journal of Computer Theory and Engineering* 3.2 (2011): 332-337.

[38] Pham, Vu, et al. "Dropout improves recurrent neural networks for handwriting recognition." *2014 14th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2014.

[39] Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." *Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 2012. 437-478.

[40] Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3. Feb (2003): 1137-1155.

[41] Masters, Dominic, and Carlo Luschi. "Revisiting small batch training for deep neural networks." *arXiv preprint arXiv*:1804.07612 (2018).

[42] Alwosheel, Ahmad, Sander van Cranenburgh, and Caspar G. Chorus. "Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis." *Journal of choice modelling* 28 (2018): 167-182.

[43] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv*:1412.6980(2014).

[44] Stollenga, Marijn F., et al. "Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation." *Advances in neural information processing systems*. 2015.

[45] Chan, William, et al. "Listen, attend and spell." *arXiv preprint arXiv*:1508.01211 (2015).