Protein Secondary Structure Prediction Using Densely Connected Convolutional Networks

Hongrui Zhang and Tieming Geng
Computer Science and Engineering Department, University of South Carolina, Columbia, SC
{hongrui, tgeng}@email.sc.edu

Abstract

Protein secondary structure prediction is a breakthrough problem for analyzing protein function in the bioinformatics field. And the recently developed deep learning technologies have already the proof success in computer vision tasks. In this project, we propose a Densely Connected Neural Network that predicted the protein secondary structure based on both the local information from the amino acid sequences and the extract information from PSSM matrix. We further tune our model with different parameters such as the number of the epoch, batch size, and feature size. The result of our model proved its effectiveness by achieving state-of-art performance, i.e., 71.55% Q8 accuracy on public CB 6133 protein data bank.

Introduction

Proteins are the key substance that corresponded to a variety of different kind of functions within organisms, including catalyzing metabolic reactions, DNA replication, respond to stimuli, and transporting molecules from one location to another. Through years of research, scientists found those functions are highly related to the structure of protein itself. During protein synthesis process, amino acid residues are put together into a polypeptide chain on the ribosome by the covalent bond between two individual amino acid residues. This kind of sequence structure also called nucleotide sequence is referred to the secondary structure of a protein. Over 200 million protein sequences have been collected in Genbank, while only about 100,000 of them have their structure is detected and stored in Protein Data Bank. The process of determining such structure is believed to be highly costed (around $100,000 per protein), thus bring the prediction of such structure the only solution to analysis the rest undetermined protein.

Related Work

Prediction of secondary structure with machine learning technologies has a rich history. The use of neural network and recently developed deep learning algorithm has already produced a significant result that raises the accuracy to about 70%. In this paper, we recurrent two state-of-art machine learning approach, and then we developed our own model with the most recently published deep learning technique called Densely connected neural network. By comparing the result within these models, our DenseNet model has the most accurate result and beats machine learning result.

The first method we explore is traditional two layers convolutional neural network with a filters numbers of 16 and 32. And instead of using a square filter to scan one image, we use a sliding window of size 3 amino acids to scan the whole sequence. Beside this, we using ReLU as our activation function, and cross-entropy as the loss function. The result from their paper is about 0.79 on the CB6133 dataset for the Q8 task. The result from our re-implementation only has an accuracy below 20%.

The other model[8] we explored is called MuFold. The idea of this model is having an inception block that concatenates the output from three different convolutional layers with a different filter size of 3, and 5, and 7, then using the concatenate output as the input for the next inception block. Also, MuFold has a bidirectional LSTM RNN model before fully connected layer that handles all the output from previous concatenate features map. The author reported a training set accuracy of 73.4% and a validation set accuracy of 69.5% both on the CB6133 dataset with Q8 classification task.

Data

The benchmark dataset we used is publicly from Zhou & Troyanskaya:[1] CullPDB. This dataset is consisting of protein sequences and secondary structure labels downloaded from the Protein Data Bank archive (PDB).[2] Each row of data stands for one protein represented by a sequence of amino acids of 700 residues. According to the data description from Zhou & Troyanskaya, each residue has 57 features in which the first 22 was formatted as one-hot encoding of 21 kinds of amino acids identity and one extra "NoSeq", then the following 9 features are the secondary structure labels, also processed with one-hot encoding with one extra digit of "NoSeq", the remainders contains terminals, solvent accessibility and sequence profile.

In training, first 22 digits of amino acids identities and 9 digits of secondary structure labels along with 20 digits of Position-Specific Scoring Matrices (PSSM) generated with PSI-BLAST[3], the introduction of PSSM will increase the prediction accuracy[4]. This entire CullPDB dataset has

6128 proteins while some sequences are longer than 700 amino acids that they are split into two overlapping sequences, so the row number of CullPDB is actually 6133. Here the shape of the data we employed is 6133*700*51.

Methods

A new convolutional network called DenseNet[5], similar to ResNet[6], employed skip connection which has the ability to reduce the concern of gradient vanishing and makes the training of deeper network easier. Differently, DenseNet adopts more skip connection, there is skip connection between any two layers inside one dense block, therefore the structure of the network becomes more dense and the backpropagation of gradient becomes more efficient. In one 5-layer dense block, there are 4 "bn-relu-conv" layers and one transition layer. In order to concatenate all the feature maps from the previous layer, the size of feature maps should maintain unchanged. The transition layer which located between two dense blocks consists of "conv" layer and "pooling" layer, because the size of feature maps don't change, DenseNet needs this transition layer to do the downsampling.

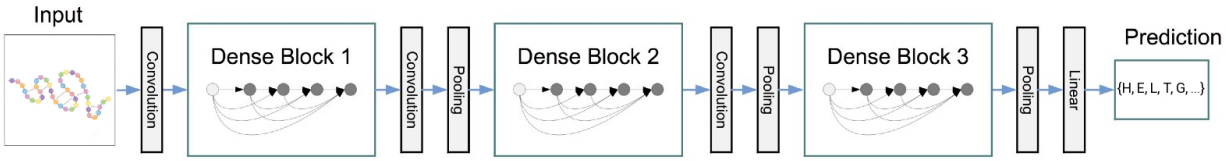

Figure.1 Structure of DenseNet

We used PyTorch ADAM optimizer[7] with 4 dense blocks contains 6, 12, 24, 16 layers inside respectively. During training, the batch size we used is 1 and the initial feature size is 512. The table below shows the change of output size in different dense blocks and transition layers. The size of final classification layer is a one-hot encoding of predicted secondary structure label, we compare this output with given labels to compute the accuracy after applying argmax on both of them.

Table 1. Parameters used in network and output feature size

Layers	**Output Size**	**DenseNet**
Convolution	512*700	3 conv, stride 1, padding 1
Pooling	512*700	3 max pool, stride 1, padding 1
Dense Block (1)	1280*700	(1 conv, 3conv) * 6
Transition Layer (1)	1280*700	1 conv

	640*700	3 avg pool, stide 1, padding 1
Dense Block (2)	2176*700	(1 conv, 3conv) * 12
Transition Layer (2)	2176*700	1 conv
	1088*700	3 avg pool, stide 1, padding 1
Dense Block (3)	4160*700	(1 conv, 3conv) * 24
Transition Layer (3)	4160*700	1 conv
	2080*700	3 avg pool, stide 1, padding 1
Dense Block (4)	4128*700	(1 conv, 3conv) * 16
Classification Layer	9*700	Hidden layer fully-connected

Criterion and optimizer

L1Loss (eq 1.) is a criterion that measures the mean absolute value of the element-wise difference between input x and target y. Thus L1Loss is best to describe the difference between two sequences has the same length which is our case.

$$loss(x, y) = \sum \left| x_i - y_i \right| / n \qquad \text{eq.1}$$

The optimizer algorithm we used is Adam. And Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.

Distance Function

The return of argmax is 700 integer number ranges from 0 to 8, since this simple type and limited range, we selected Edit Distance[9] to calculate the distance between the predictions and labels. Edit distance is one method of quantifying how dissimilar two strings are by counting the minimum number of operations required to transform one into the other one.

Results

We run our model with both single epoch and five epoch with or without PSSM matrix as part of the input features. The results for both training set and validation set are presented in the following table.

Table 2. Training accuracy and validation accuracy without PSSM matrix

	Training accuracy	Validation accuracy
1 epoch	59.73%	62.24%
5 epoch	74.20%	71.55%

We split the 6133 protein sequence into two subset, the first 5800 protein sequences are treated as training dataset, the rest 333 protein sequences are used as validation dataset.

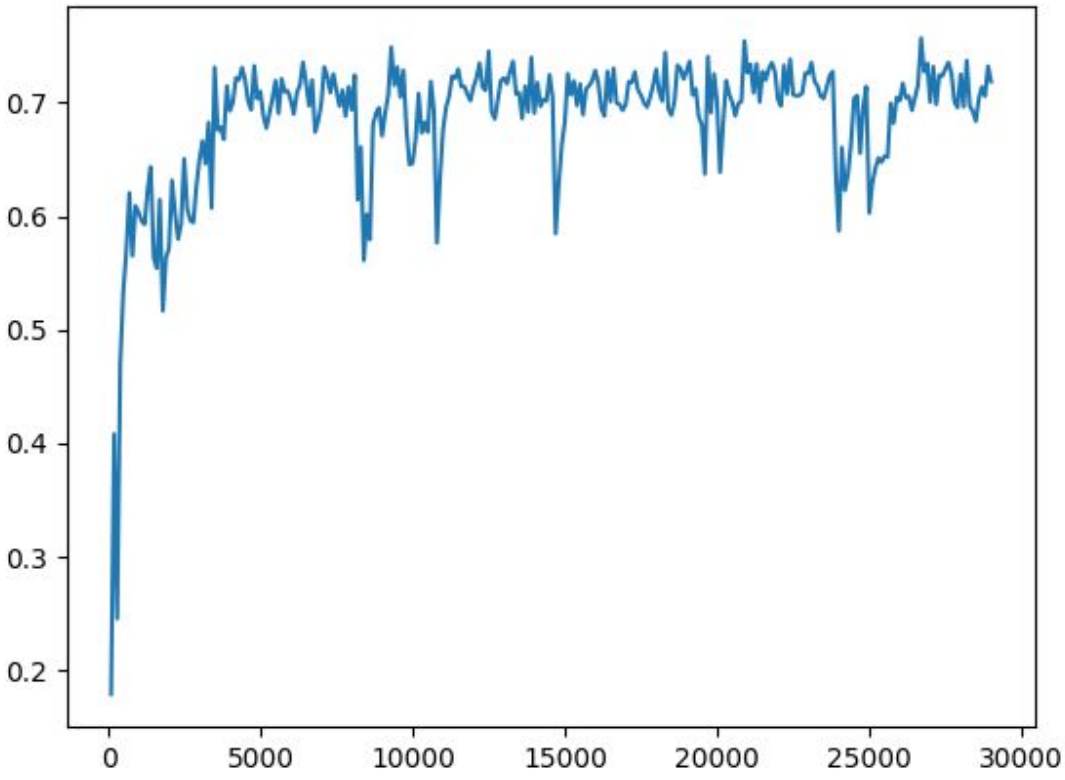

Figure 2. Accuracy result for the first 29000 training with only amino acid information

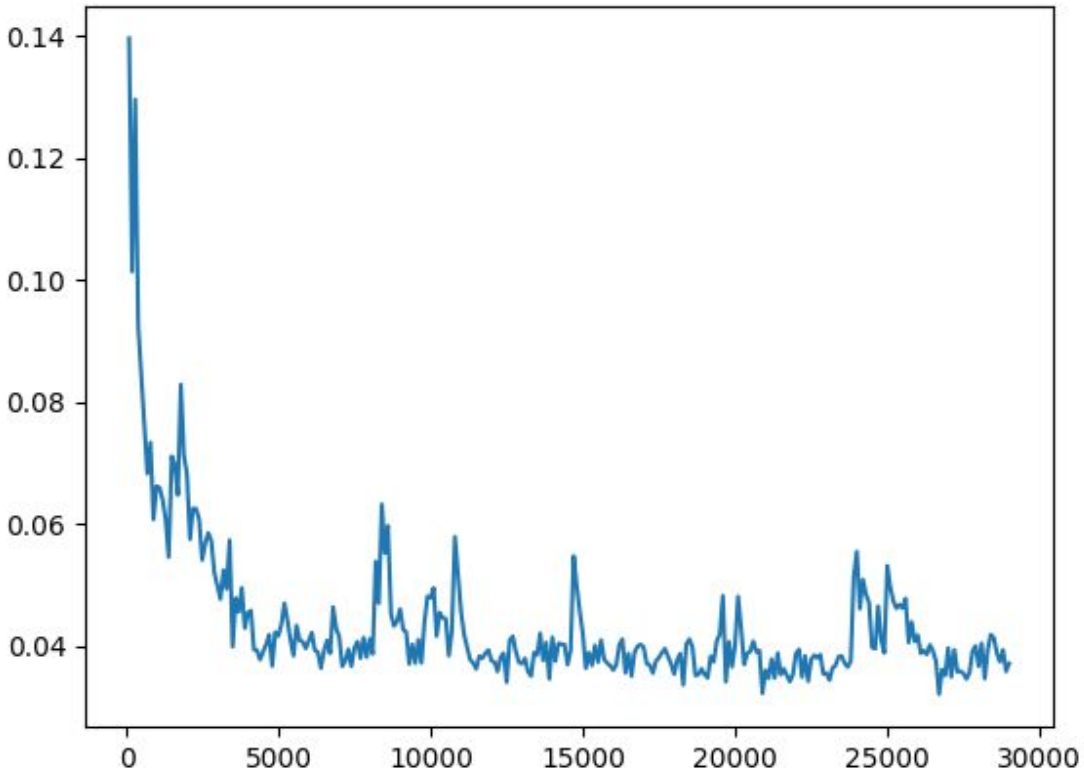

Figure 3. Loss result for the first 29000 training with only amino acid information

The fluctuation present in these figure is due to the model is restart its learning process for another epoch. Such irregulation will vanish as the model march further. These graph also proof that five epoch with a total number of 29000 training process is enough to prove the effectiveness of our model, compared with our other model with 20 epoch run totally.

Discussion

Features

The features parameters for our final model is set to have an initial feature size of 512 with a growth rate of 128. So After the whole training cycle, the total number of parameters before fully connected layer will be 2889600. With each of these parameters multiplies with one single hidden layer neural, the need of computation power is at large. Though this large number of feature significantly improve our model's accuracy compared with our first test model with an initial feature size of 16

and a growth rate of 32, we did not further increase the size of the feature due to lack of computation power and memory.

Performance

At the beginning of the training, the program was run by an Intel i7 6700K CPU which didn't bring satisfying performance, then we switched to CUDA platform of a NVIDIA GTX 1070 with 8GB memory. CUDA could provide almost 13 times of performance of CPU.

Table 3. Comparison of efficiency on different platform

	Intel i7 6700K	**NVIDIA GTX 1070**
15 seconds	10 proteins	133 proteins
1000 proteins	24 minutes 46 seconds	1 minute 50 seconds

The entire 5800*700*21 training data with batch size equals to 1 occupied around 2GB GPU memory.

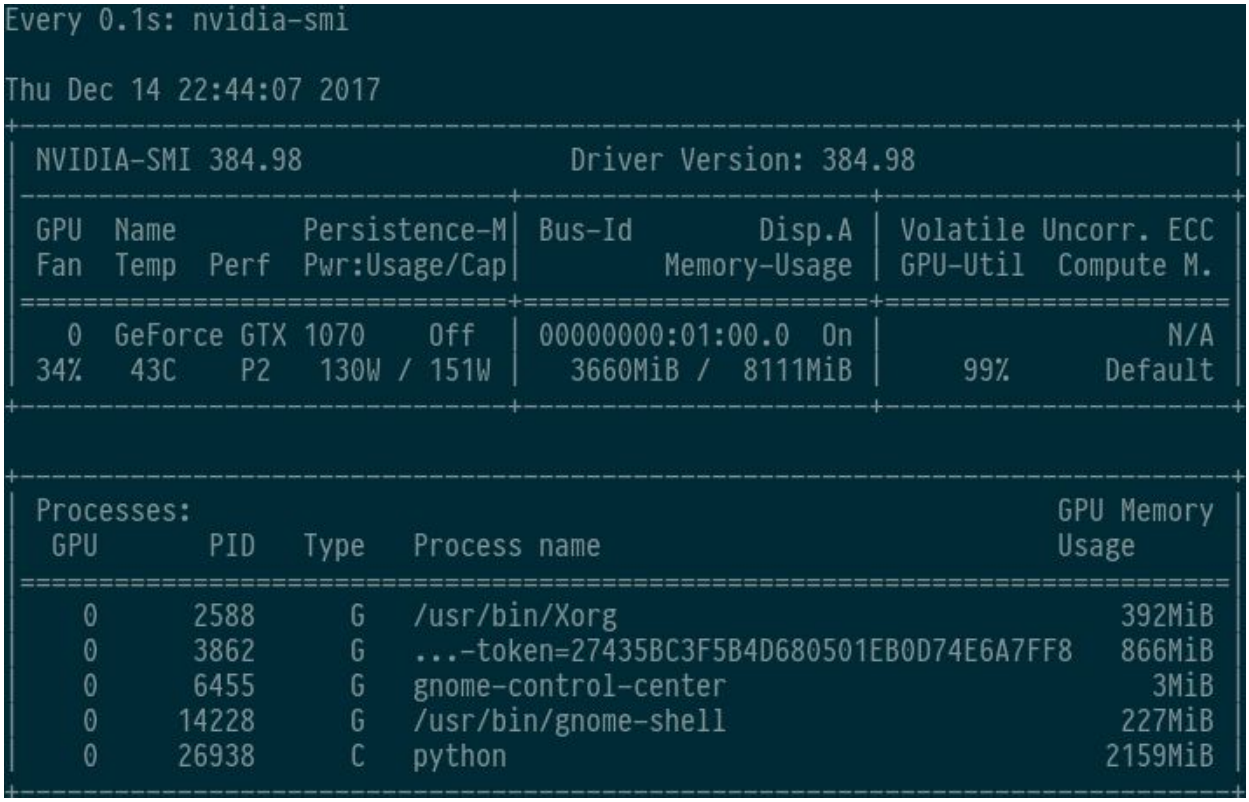

Figure 4. GPU memory usage

Due to the time and environment, the biggest epoch round we have is 20 epochs, from 5 epochs to 20 epochs, no significant improvement was observed but an improvement of accuracy still exists, so in the future, we may try with much larger epoch such as 500.

Batch Size

We tried several different batch size from 1 to 4 (the size of GPU memory limits bigger batch size), the results turn out that smallest batch size which means 1 could provide best accuracy. Unlike image data, protein sequence is 1-dimensional array of features, bigger batch size would build longer sequence which may seriously impacts the learning efficiency of the network model.

References

1. Zhou, J., & Troyanskaya, O. G. (2014, March). Deep Supervised and Convolutional Generative Stochastic Network for Protein Secondary Structure Prediction. In ICML (pp. 745-753).
2. Berman, H.M., Henrick, K., & Nakamura, H. (2003). Announcing the worldwide Protein Data Bank. Nature Structural Biology 10(12): 980.
3. Bhagwat M, Aravind L. PSI-BLAST Tutorial. In: Bergman NH, editor. Comparative Genomics: Volumes 1 and 2. Totowa (NJ): Humana Press; 2007. Chapter 10. Available from: https://www.ncbi.nlm.nih.gov/books/NBK2590/
4. Dehzangi, A., Paliwal, K., Lyons, J., Sharma, A., & Sattar, A. (2013, June). Exploring potential discriminatory information embedded in pssm to enhance protein structural class prediction accuracy. In IAPR International Conference on Pattern Recognition in Bioinformatics (pp. 208-219). Springer, Berlin, Heidelberg.
5. Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2016). Densely connected convolutional networks. arXiv preprint arXiv:1608.06993.
6. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
7. Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
8. Shih-Cheng Huang, et.al,. Recurrent Neural Networks for Protein Secondary Structure Prediction
9. Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady. 10 (8): 707–710.