Protein Sequence Function Classification

Sean Whalen, Kirsten Winter, Boyang Zhang

I. INTRODUCTION

Proteins are widely studied using numerous highly sophisticated equipment in laboratories and through computational approaches. One important problem is the functional classification of proteins, using only structural information of these molecules. There has been success in numerical functional classification using high level structures of protein (from primary to quaternary). For this project, we aim to explore the attainability of classifying protein functions using only primary structure, i.e. the amino acid sequences. The code can be found at https://github.com/bzhang610/ProteinSeq

II. DATASET AND PREPROCESSING

The data we use is from the Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB). The database has over 400,000 protein sequences labeled with over 3000 types of functions. Most of the functions, however, have only a few sequences. Since the database is open for uploads, there are various uploads different in features we are not interested in but with same amino acid sequences. For this project, we remove the duplicate sequences and focus on 10 common classes of the 3000, while having a total of 38901 sequences.

III. CLASSIFICATION MODELS

To classify protein sequences based on their functions, we constructed models that have shown success in sequential data classification.

A. Hidden Markov Model

Hidden Markov Modeling is an approach to using a Markov chain–a discrete time stochastic process that only keeps track of the current and previous states–with hidden states to calculate the probability of an observed sequence. HMM is defined by having a set of hidden states which behaves as a Markov chain, with each state emitting an observable symbol. HMM can be used to generate a sequence or to calculate the probability of an observed sequence. Different types of protein sequences are similar to DNA sequences in the sense that both can be modeled as chains of amino acids; however, that's the main extent of their similarities. Protein chains can have up to 25 different types of amino acids in an observable sequence, while DNA has 4. Being able to model the protein sequences, without knowing the potential codon sequences and patterns, is much more difficult for proteins than DNA.

In this project, the sequence of observable symbols is the sequence of amino acids, which forms a stochastic process. The hidden chain is treated as the effective codon sequence of the protein. The transition matrix for each protein (x): **A**x = a[i][j] is the probability of observing the current amino acid, given the previous acid–represented by indexes j and i, respectively; each protein also has an initial distribution likelihood vector.

B. Deep Learning Approaches

Deep learning has been successful in numerous fields in recent years. For this project, we will focus on models that have been especially successful in natural language processing, since we can interpret protein sequences as sentences with amino acids as vocabularies. Since most deep learning models requires same input size, after the preprocessing procedures mentioned above, the sequence are also trimmed or padded to a length of 256.

1) Long-Short Term Memory (LSTM): LSTM is a recurrent neural network structure. When sequential data is fed into the recurrent neural network (RNN), RNN not only takes current input but as well as inputs from previous locations in the sequence of the same layer. This allows the network to utilize the sequential nature of the data. However, a typical difficulty RNN encounters is the vanishing gradient problem. Since the RNN layers take input across the sequence, when the model tries to update weights based on the loss, all weights change in response to a change in loss. Thus the network is unable to calculate the gradient and therefore fail to continue reduce the loss.

The long short-term memory model takes the input the same way as RNN, but it also has a memory cell, forget gate and output gate. This allows LSTM units to select inputs to be discarded, stored (memorized for later use) or for calculating output, and subsequently solve the gradient vanishing problem. For analyzing language, an n-gram word embedding is usually applied to the input before fed into the LSTM networks. This procedure uses a moving window of length n to analyze the word combination along the sequence. We then have a vocabulary comprise of all these unique n-length words combinations. For this project, we apply something similar with 1-dimensional convolution layer (512 filters with kernel size 3). We can interpret the number of filters as maximum vocabulary

2) Convolution Neural Network (CNN): CNN is widely used for pattern recognition with image and audio data. It has the benefit of using small number of parameters to learn complex features and also being spatial invariant, that it is able to detect certain feature across all locations in the input data.

Similar to LSTM model, the first hidden layer after the input layer is a CONV1d layer working as an embedding layer. Instead of treating each amino acid as separate input like in LSTM model, the CNN model adds another CONV1D layer after the embedding layer and try to learn reoccurring features for a given function type.

IV. Results

In this section, we will evaluate the performance of each model.

A. HMM

The calculation of the likelihood an observed sequence belonging to a protein group, x, is measured as the sum of the probability of the initial amino acid of protein group x and the probability of each transition–for protein group x–in the sequence. The highest value of the calculations indicates the observed sequence's group; the highest 3 scores are kept track to further show the effectiveness of the algorithm. Each sequence in every classification of protein is calculated and assigned the it's top-3 classifications, which are compared to each sequence's actual classification.

The best iteration of the algorithm works for ten sequences and uses all sequences to calculate the transition matrices; this isn't the ideal method for ”training” the matrices, but it yields the best transition rates and therefore the most accurate classifications. The overall results for this method were not as anticipated, yielding a classification accuracy of 32.56 percent. The Top-3 classification accuracy–where the correct classification is the top-3 most likely classification of the given sequence–gives a better result at 65.25 percent, but still not optimal. For all sequences, having the correct classification outside the top-3 most likely at 34.74 percent is particularly poor. Given the lack of complexity of the HMM and probability classification algorithm, great results would be difficult to achieve; a more complex HMM or probability algorithm would likely improve the performance.

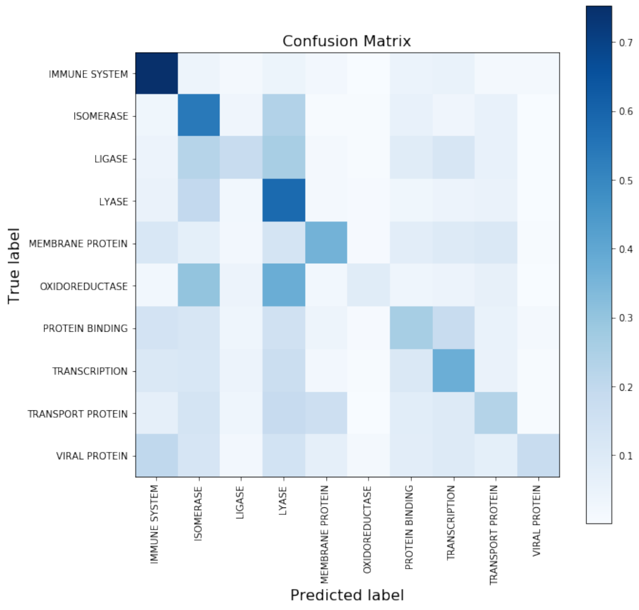

Fig. 1. Confusion Matrix: HMM

Producing a confusion matrix from the results gives more context into how the incorrect results came about. Of all ten classes, seven had the highest count of classifications to itself, while the other three classes had one or more higher populated count than itself; in other words, three classes had incredibly poor accuracy. The highest classification accuracy was for immune system proteins at 75.27 percent, which is fantastic compared to the average. The worst three (ligase, oxidoreductase, and viral proteins) each had classification accuracy of 18.08 percent, 8.80 percent, and 17.81 percent, respectively. These three outliers also had their highest classification count of a different protein; this is meaningful because it indicates each of the three proteins likely have either similar rates for all transitions or similar transition rates to other proteins. Having more training data than the given set or using a more complex algorithm could improve these results, further improving the overall accuracy.

From a performance driven perspective, the code runs relatively quickly, with the entire HMM program completing in under 20 seconds. From a complexity perspective, each sequence is addressed twice, once for building the transition matrix and once for calculating the most likely classification. The extent to which every acid of a sequence is addressed isn't matched on a complexity base, so you could approximate this portion of the project to be $\mathbf{O}(2N)$. From a storage perspective, this part of the project isn't too complex. Other than a python dictionary with a 2-list for the transition rates and a 1-dimensional dictionary of lists, there aren't any significant data structures.

We decided to use 10 of the 13 proteins available on the database for the HMM to get the best results, removing the 3 proteins that yielded the worst classification accuracy and decreased the average correct classification significantly. Specifically, the accuracy of the most likely classification being correct decreased by nearly half, and the top-3 accuracy decreased by over a third. For similar reasons, the full collection of data was used to compute the transition rates for each protein class. We began by attempting to optimize on a small percentage of the sequences for each class (5-10 percent), and then on increasingly larger sized sequences, but we came to a conclusion that given the full set for training classification should be as strong as possible based on the algorithm. In TABLE I, the accuracy for each set is listed as to compare the different methods:

	Test Accuracy	Test Top-3 Accuracy
Full Dataset, 10 Seq.	32.56%	65.26%
Full Dataset, 13 Seq.	18.75%	40.50%
10% Dataset, 10 Seq.	25.80%	52.37%
10% Dataset, 13 Seq.	14.44%	31.82%

TABLE I
HMM Classification Performance for Different Models

As indicated in above, the 10-sequence models held up better in general, as did the full data set for calculating transition rates. Ideally, we would have used even fewer sequences, but the challenge of this project was to show how efficiently this algorithm could run on this data set with all the proteins. Of course, 10 is a better, more round number than 13, so that was the primary reason we selected 10 protein classes; however, to further increase the top classification accuracy, we might decide to use such a relatively simple algorithm to classify a smaller set of protein classes.

B. LSTM

The LSTM model that yielded the best result has 4 layers. This first layer is 1-dimensional convolution layer with 512 filters of kernel size 3. As mentioned above, this CONV1D layer acts as a embedding layer for LSTM inputs. After the CONV1D layer is a LSTM layer with 256 units, and then a fully connected layer of 128 units before the last output layers of 10 units, representing the 10 classes. We use dropouts (0.6) between each layer to avoid overtraining. The CONV1D and fully connected layer use Relu activation function and LSTM uses tanh. The output layer uses softmax activation for classification. We use Adam optimizer for stochastic gradient descent and categorical cross-entropy loss to achieve multi-class classification.

The loss and accuracy history is shown in fig 2, 3. Over the span of 50 epochs, both training and validation loss gradually decreases. The validation loss and accuracy reaches plateau near the end of the training while training loss keeps decreasing. The structure and other hyper parameters were tuned to avoid overtraining while increasing validation accuracy.

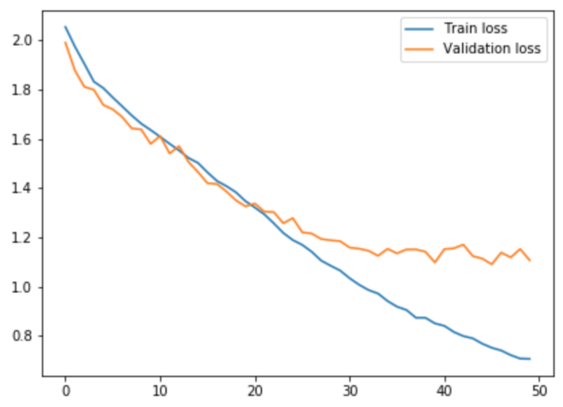

Fig. 2. LSTM Loss

From the confusion matrix and classification performance figures shown in fig 4, 5, we can observe a few interesting features of this model. The model's performance mostly depends on the support size. We can observe that the two classes with most samples, Hydrolase and Transferase, perform generally better than classes with fewer samples. The model also has the tendency to misclassify sequences as these two mentioned classes. Classes with fewer samples tend to have worse performance, for example, signaling protein and isomerase both have sub-par F1 score. However, although immune system's sample size is not large, it performs the best with the LSTM model. This is likely due to a strong

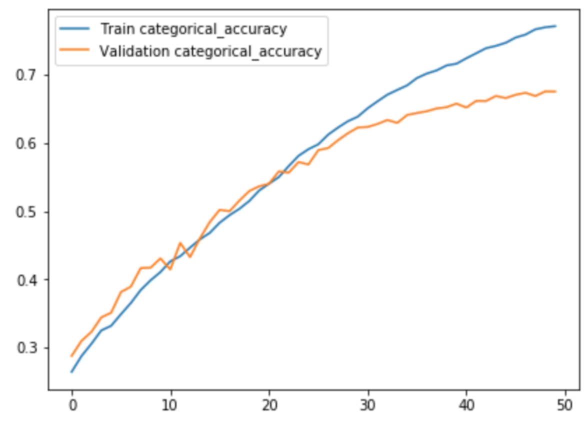

Fig. 3. LSTM Accuracy

similarity between these sequences' amino acids composition.

```
                  precision    recall  f1-score   support

       HYDROLASE       0.71      0.78      0.75      2210
   IMMUNE SYSTEM       0.88      0.76      0.82       495
       ISOMERASE       0.60      0.39      0.47       308
           LYASE       0.61      0.49      0.55       442
  OXIDOREDUCTASE       0.75      0.75      0.75      1243
SIGNALING PROTEIN      0.40      0.33      0.36       377
   TRANSCRIPTION       0.50      0.56      0.53       438
     TRANSFERASE       0.66      0.74      0.70      1603
TRANSPORT PROTEIN      0.63      0.46      0.53       372
   VIRAL PROTEIN       0.61      0.48      0.54       293

       micro avg       0.68      0.68      0.68      7781
       macro avg       0.63      0.58      0.60      7781
    weighted avg       0.67      0.68      0.67      7781
```

Fig. 4. LSTM Classification Performance

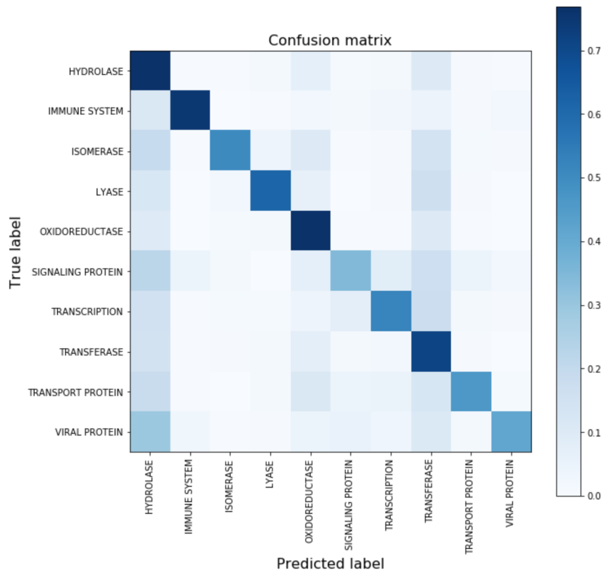

Fig. 5. LSTM Confusion Matrix

C. CNN

The CNN model that yields the best result has 2 Conv1D layers with 128 filters of kernel size 3 and 64 filters of kernel size 2. Both CONV1D layer is followed by a maxpooling layer (pool size 2) to reduce size. The model has one fully connected layers with 64 units and one with 10 units as output layer. The convolution layers and the first fully connected layer use Relu activation funtion. The output layer uses softmax activation for classification. Similar to the LSTM model, the CNN model uses Adam optimizer and categorical cross-entropy loss.

The loss and accuracy history is shown in fig 6, 7. As we can see, the model is overtrained. However, this is the result of tuning hyper parameters to achieve the best validation accuracy. Compare to the LSTM model, CNN has few parameters to learn and the model also takes fewer epoch to reach the plateau. The accuracy performance, however, is similar to that of LSTM. Notice that the higher training accuracy is due to more total epoches trained.

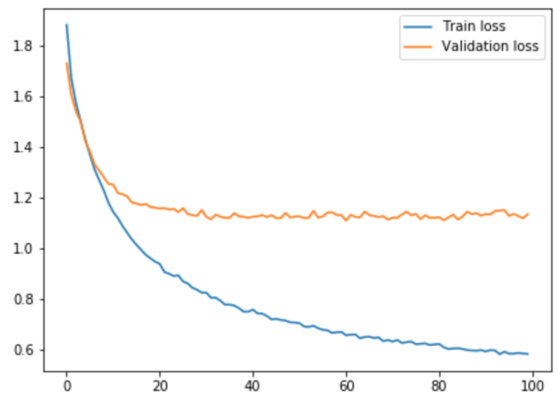

Fig. 6. CNN Loss

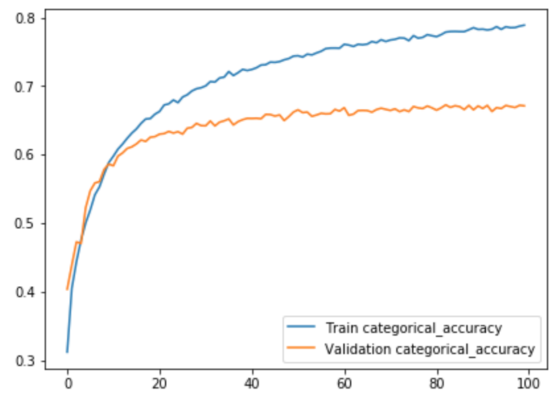

Fig. 7. CNN Accuracy

The confusion matrix and classification performance for the CNN model is shown in fig 8, 9. It exhibits similar characteristics to the LSTM model, where classes with more samples and class immune system performs well, while classes with fewer samples have less than ideal performance. Signaling protein still performs the worst. This reassured that the poor classification for this class is model independent.

```
                  precision    recall  f1-score   support

        HYDROLASE      0.66      0.77      0.71      2114
    IMMUNE SYSTEM      0.90      0.75      0.82       536
        ISOMERASE      0.74      0.50      0.60       317
            LYASE      0.74      0.62      0.67       479
   OXIDOREDUCTASE      0.68      0.77      0.72      1208
SIGNALING PROTEIN      0.50      0.34      0.41       377
    TRANSCRIPTION      0.61      0.52      0.56       441
      TRANSFERASE      0.62      0.71      0.67      1632
TRANSPORT PROTEIN      0.71      0.46      0.56       367
    VIRAL PROTEIN      0.78      0.41      0.54       310

        micro avg      0.67      0.67      0.67      7781
        macro avg      0.69      0.59      0.63      7781
     weighted avg      0.68      0.67      0.67      7781
```

Fig. 8. CNN Classification Performance

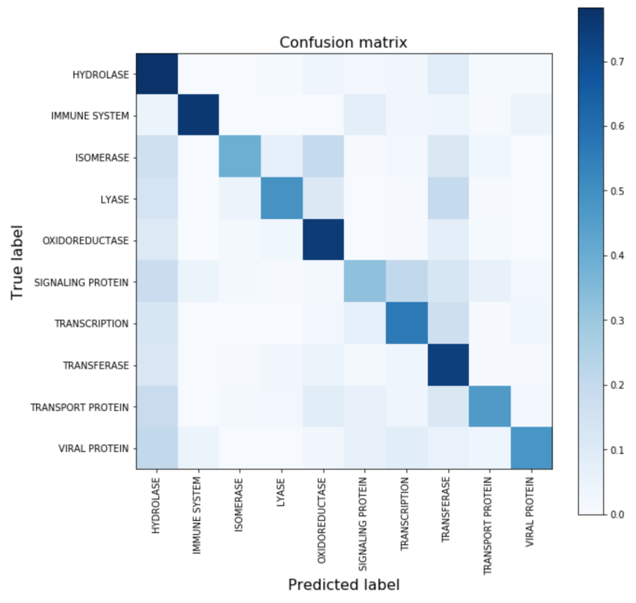

Fig. 9. CNN Confusion Matrix

D. Comparison

In TABLE II we can compare the performance of different models. Overall, the deep learning models can classify protein functions using only the amino acid sequences better than HMM model. This is a result of the more complex structures in both deep learning models than that of the hidden Markov model.

Between the two deep learning models, the classification accuracy is similar for both top-1 and top-3 results. However, the CNN model has fewer parameters and faster to train. This implies that it is more efficient to learn the patterns of segments of amino acid sequences than to analyze the entire sequence as a whole.

V. CONCLUSIONS

It is difficult to classify protein function only based on the amino acid sequences. Even with the best performing

	Training Accuracy	Test Accuracy	Test Top-3 Accuracy
HMM	100%	32.6%	65.3%
LSTM	86.1%	67.5%	85.4%
CNN	96.3%	67.1%	85.4%

TABLE II

CLASSIFICATION PERFORMANCE FOR EACH MODEL

models, we are only able to achieve an accuracy around 67%. However, with the deep learning models, we can narrow down the choice of functions for a given sequence to 3 with around 85% confidence. Using only the amino acid sequences, we do not have enough information for classification. In order to achieve better classification results, we need to extract more information from the proteins.

REFERENCES

[1] Liu X. Deep recurrent neural network for protein function prediction from sequence. arXiv preprint arXiv:1701.08318. 2017 Jan 28.

[2] Mirarab, Siavash. Hidden Markov Models Lecture Slides. ECE 208. 2019 Apr 8.

[3] Szalkai B, Grolmusz V. Near perfect protein multi-label classification with deep neural networks. Methods. 2018 Jan 1;132:50-6.