Incremental & Semi-Supervised Learning for Functional Analysis of Protein Sequences

1st Mali Halac
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
mh3636@drexel.edu

2nd Bahrad Sokhansanj
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
bahrad@molhealtheng.com

3rd William L. Trimble
Argonne National Laboratory
University of Chicago
Chicago, IL, USA
wltrimbl@uchicago.edu

4th Thomas Coard
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
thomas.coard@gmail.com

5th Norman C. Sabin Jr.
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
sabinn49@students.rowan.edu

6th Emrecan Ozdogan
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
ozdoga67@rowan.edu

7th Robi Polikar
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
polikar@rowan.edu

8th Gail L. Rosen
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
glr26@drexel.edu

2021 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-7281-9048-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSCI50451.2021.9659958

***Abstract*—Current approaches for the functional annotation of proteins rely on training a classifier based on a fixed reference database. As more genes are sequenced, the size of the reference database grows and classifiers are retrained with the old and some new data. Considering the ever-increasing number of (meta-)genomic data, repeating this process is computationally expensive. An alternative is to update the classifier continuously based on a stream of data. Thus, in this study, we propose an incremental and semi-supervised learning approach to train a classifier for the functional analysis of protein sequences. Our method proves to have a low computational cost while maintaining high accuracy in predicting protein functions.**

***Index Terms*—Incremental clustering, semi-supervised learning, functional annotation, protein sequence**

I. Introduction

The functional analysis of proteins can involve costly, time-consuming, and difficult experiments. As such, it is helpful to use computational techniques to extend our knowledge about previously studied and characterized proteins to develop hypotheses for, and thus narrow and focus the experimental work required to understand relatively less studied or newly sequenced proteins. Using computation to predict the functional roles of proteins found in genomic samples may be crucial in disease diagnosis, development of novel drugs, and a deeper understanding of the evolutionary relationships among proteins [1], [2].

There exists many computational approaches to predict a protein's function. One of the most commonly used methods relies on the protein sequence homology – statistically significant sequence similarity. Sequence homology based methods are informative about the evolutionary relationship of proteins. Homologous proteins that were separated by a speciation event are called orthologs, which are known to share a common function. Thus, it is possible to infer a protein's function by aligning it to its ortholog [3].

This work was supported in part by NSF grants #1919691 and #1936791.

One of the recent problems in genomics studies is the large computational cost of conducting such analyses. Traditional approaches to detect homology rely on performing a pairwise alignment against an entire reference database, a computationally expensive method. Moreover, the recent boom in the number of proteins sequenced requires an incremental approach that would continually update the classifier without reprocessing the entire database. The MG-RAST metagenomics analysis server, for example, has 467,237 metagenomes containing 1,979 billion DNA sequences [4], which is translated to possible reading frames for protein alignment. In this study, we demonstrate the application of an incremental and semi-supervised learning algorithm for the functional analysis of protein sequences.

II. Background

A. Sequence Alignment

Proteins are composed of polymer chains of up to 22 different kinds of amino acids, which are encoded by an organisms's DNA. The permutations of the amino acids that form the chain, the protein "sequence," provide the structure and function of the protein. As a consequence, a protein's sequence includes information about the evolution of the organism, as well as the functional roles of the protein.

Sequence alignment is a method that is used to compare different protein sequences to discover implicit facts about the functional role or common evolutionary descent of proteins [5]. The two most prevalent types of sequence alignment are: pairwise and multiple. Pairwise sequence alignment compares two sequences to perform a similarity search in a database. Multiple sequence alignment, on the other hand, compares three or more sequences to infer homology between the input sequences.

In this study, we use Diamond [6], [7] to perform pairwise sequence alignment and MUSCLE [8] for multiple sequence alignment.

B. Protein Function Prediction

In order to predict a protein's function, we first have to understand what a "function" is. The term function, by itself, is vaguely descriptive. Thus, there are many studies trying to delineate the term. The Gene Ontology Database, for example, defines protein function in a hierarchical way in three distinct parts: molecular function, cellular component, and biological process [9], [10]. One major challenge in using Gene Ontology is to decide which level of the hierarchy to operate in. Another functional construct that can be used for function prediction is the Clusters of Orthologous Genes (COG) database where proteins are clustered according to their orthologous relationships [11]. Orthologous proteins are known to frequently implement the same function in different organisms. Thus, each orthologous group in COG database has a functional annotation indicating the predicted function of the proteins in that COG [12]. The current COG database includes 4,877 orthologous groups and 3,213,196 proteins [13]. In this study, we use COGs as a reference to determine protein function based on sequence homology.

C. Incremental Learning for Genomic and Genetic Annotations

Fueled by recent advances in technology, the cost of DNA sequencing is reducing at a rate that is faster than the increase in computational power resulting in sequencing many novel organisms and genomes that subsequently need to be analyzed [14], [15]. The conventional methods to align these sequences rely on methods that train on a fixed reference database. Every time more reference and genomic sequences are added, the entire alignment must be rerun from scratch. However, considering the ever increasing number of reference sequences and genomic data, rerunning this approach is expensive due to the cost of computational power [16]. An alternative solution is to update the classifier on new data only, thus avoiding the computational cost to reprocess the entire database [17].

III. Methods

All the experiments were run on Drexel University's high performance computing cluster, Picotte. The reported CPU times are for an Intel Xeon Platinum 8268. The CPU times and memory usage are measured using Slurm Workload Manager's "seff" command.

A. Protein Data Set

The Clusters of Orthologous Genes (COG) database is a useful tool for the functional annotation of protein sequences. In the COG database each protein is assigned to an orthologous group which represents a general function. An example of such a group is COG0008 which includes glutamyl- or glutaminyl-tRNA synthethase proteins. In the case of multi domain proteins, however, the COG approach assigns only the functional domains to orthologous groups to prevent misleading annotations for proteins [18]–[20].

The data sets used for this study are based on the Swiss-Prot and TrEMBL databases both of which are subset of the UniProt database. The Swiss-Prot is a result of manual annotations, whereas TrEMBL consists of automatically annotated proteins [21].

For this study, we formed two different data sets to test our algorithm. The first data set is based on the COG proteins that are part of the Swiss-Prot database. In order to ensure that approximately the same percentage of samples of each target class are present in both the training and validation sets, we performed a stratified split. As a result, the training set contains 60,297 protein sequences with 63,146 functional domains from 3,204 COGs; whereas the validation set contains 57,450 protein sequences.

In order to minimize the error in the labels of proteins used for training for larger data sets, the second data set is formed by employing COG proteins that are part of the Swiss-Prot database as the training set and COG proteins that are part of the TrEMBL database as the validation set. Using the manually curated Swiss-Prot sequences as our training set allows us to have a more reliable starting point for our model. The training set contains 117,747 protein sequences with 122,202 functional domains from 3,858 COGs, whereas the validation set contains 2,938,741 protein sequences. Furthermore, all the protein sequences for which a COG was not defined in the training was excluded from the validation sets for both the first and the second data sets.

B. Centroid Detection for COGs

In order to develop an algorithm that is fast and accurate, we devised an efficient way of representing COGs. Our algorithm is designed to detect multiple centroids for each COG. These centroids are later used to predict the functions for novel proteins.

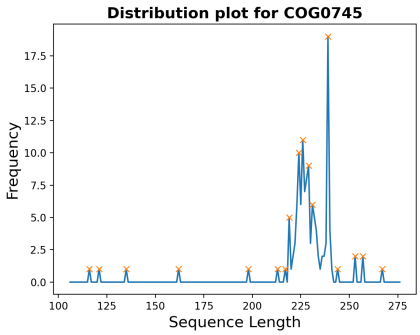

Fig. 1. **The distribution plot for COG0745 showing the local maxima.** Consensus groups are formed between each local maxima. Each protein sequence is assigned to a consensus group according to its sequence length.

Our algorithm relies on the computation of the consensus sequence to detect the centroids. The consensus sequence may be defined as the order of most frequently occurring amino acid in an alignment. The centroid detection is performed in 5 steps:

1) The distribution of sequence lengths is computed for each COG in the reference database (Fig. 1).

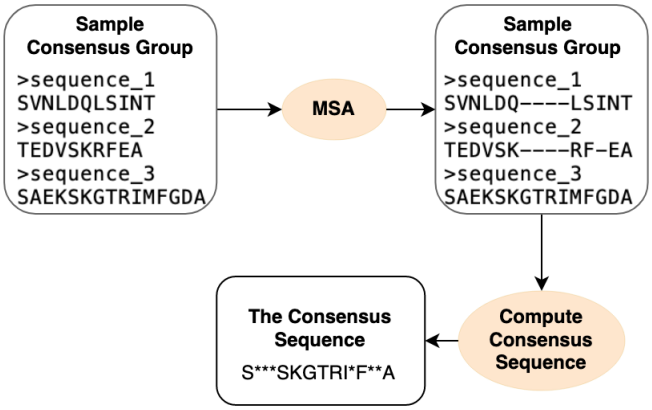

Fig. 2. **Computation of the consensus sequence.** The consensus groups may be defined as subgroups for each COG. Each consensus group has multiple sequences in them. A multiple sequence alignment is performed for each subgroup to compute the consensus sequences. The consensus sequence is a concise representation of the sequences in the subgroups.

2) The local maxima of each distribution is detected (Fig. 1).
3) Consensus groups are formed between each local maxima. Protein sequences are assigned to consensus groups according to their sequence lengths (Fig. 2).
4) A multiple sequence alignment (MSA) is performed for each consensus group using MUSCLE (Fig. 2).
5) The consensus sequence of the multiple alignment is computed using 0.5 threshold. In other words, an amino acid has to be found at a specific position in at least 50% of the sequences to be annotated in the consensus sequence. Otherwise, the position is noted with an asterisk (*) (Fig. 2).

The process of obtaining the initial consensus sequences is not parallelized since the size of our training set is relatively small. For larger training sets, this process can be parallelized by executing the above steps simultaneously for each COG (Fig. 3 Part A). For the training set of the second data set, it took 1 hour and 28 minutes of CPU time to prepare the initial consensus sequences.

C. Algorithm

Once we obtain the initial consensus sequences, we feed them to the main algorithm along with the validation sequences. The algorithm first performs a pairwise sequence alignment using Diamond in sensitive mode with two different parameters entered by the user: sequence identity and query coverage. The sequence identity represents the percentage of identical amino acid residues that were aligned against each other in the local alignment. A threshold of 40% for the sequence identity means the pairwise alignment will only return the alignments having at least 40% sequence similarity. The query coverage, on the other hand, informs the user about the percentage of the query sequence that overlaps with the target sequence. Moreover, the pairwise alignment is run using the ”–max-target-seqs 1” option so that it only returns the alignment with the highest score. The output of the pairwise

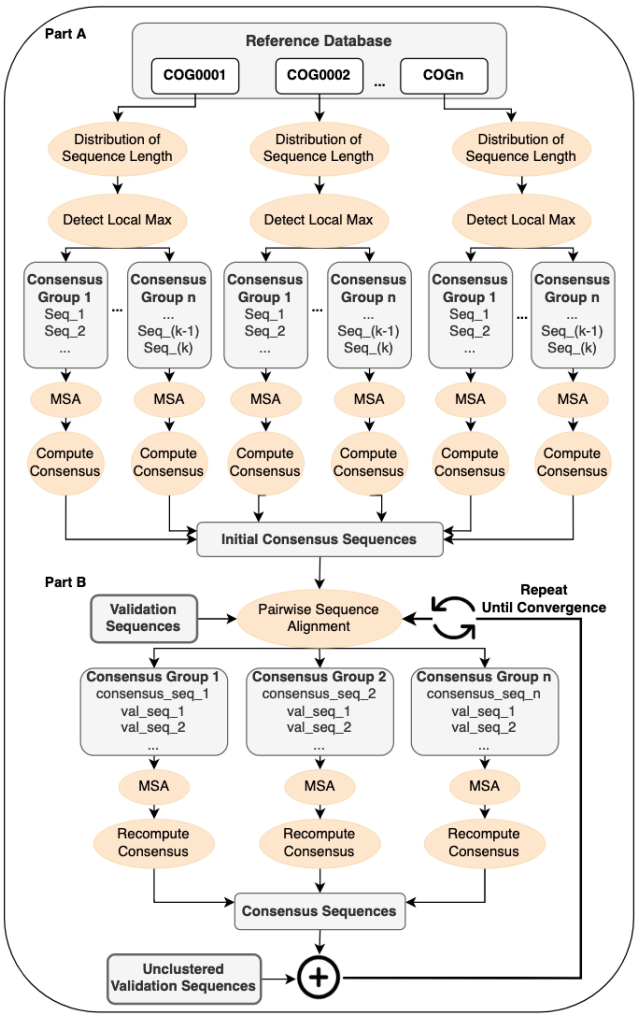

Fig. 3. **The algorithm. Part A: Centroid Detection for COGs** The distribution of sequence lengths are computed for each COG in the reference database. Then, the the local maxima of the distributions are detected. Consensus groups are formed between each local maxima and the protein sequences are assigned to consensus groups according to their sequence lengths. Then, a multiple sequence alignment (MSA) is performed and the consensus sequences are computed for each consensus group. Part A is performed only once to get the initial consensus sequences. **Part B: Main Algorithm.** When a batch of data is fed into the algorithm, a pairwise sequence alignment is performed against the initial consensus sequences. The Diamond protein aligner is used for the pairwise sequence alignment in sensitive mode with 2 different parameters entered by the user: sequence identity and query coverage. A threshold of 40% for sequence identity will return the alignments having at least 40% sequence similarity. The query coverage, on the other hand, informs the user about the percentage of the query sequence that overlaps with the target sequence. Moreover, the pairwise alignment is run using the ”–max-target-seqs 1” option so that it only returns the alignment with the highest score. The consensus sequences are grouped with the validation sequences they aligned to and form the consensus groups. Then, a multiple sequence alignment (MSA) is performed for each consensus group and their consensus sequence is recomputed. Once the consensus sequences are updated, they are fed into the algorithm again along with the unclustered validation sequences. The algorithm runs steps in Part B until it cannot cluster any more sequences for the current batch.

alignment informs us about the predicted function indicating which validation sequence was aligned against a certain consensus sequence (Fig. 4). Then, each consensus sequence is

Query	Target	Seq_Identity	Length
AAM01497_1	COG0001_group_0	48.2	415
AAM02290_1	COG0002_group_7	51.3	345
AAM01238_1	COG0005_group_3	50.2	255
AAM02113_1	COG0013_group_38	44.1	950

Fig. 4. **Output of the pairwise sequence alignment.** "Query" column shows the protein ids for the validation sequence. "Target" column indicates to which consensus sequence the query is aligned to. The third column indicates the sequence identity: the percentage of identical amino acid residues that were aligned against each other in the local alignment [22]. The fourth column shows the total length of the local alignment. We can infer the function of a query sequence by looking at its pairwise alignment. In this figure the protein "AAM01497_1" is aligned to "COG0001_group_0" thus its function is predicted as the general function for sequences in COG0001: "Glutamate-1-semialdehyde aminotransferase".

grouped together with validation sequences it's aligned to. For each group formed, a multiple sequence alignment is performed and their consensus sequence is recomputed (Fig. 3 Part B). The updated consensus sequences are used in the next iterations to cluster any unclustered validation sequences. This way the algorithm learns from both the initial training and the newly clustered sequences, hence behaving as a semi-supervised classifier.

Our algorithm relies on a similarity search against the consensus sequences. The number of validation sequences that gets clustered depends on the consensus sequences' representative power. Similarly, the parameters used for the pairwise sequence alignment may also affect the number of validation sequences that gets clustered. When the new consensus sequences are computed, they are fed into the algorithm again along with the unclustered validation sequences to maximize the number of predictions for a batch of data. The algorithm repeats the steps illustrated in Fig. 3 Part B until it cannot cluster any more sequences for the current batch.

$$Accuracy : \frac{Correct\_Predictions}{Total\_Predictions} \tag{1}$$

$$Precision : \frac{True\_Positive}{True\_Positive + False\_Positive} \tag{2}$$

$$Sensitivity : \frac{True\_Positive}{True\_Positive + False\_Negative} \tag{3}$$

$$F1\_Score : 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity} \tag{4}$$

The accuracy, precision, sensitivity, and F1 scores reported in the results section are obtained using the standard equations (Equations 1, 2, 3, 4).

IV. Results and Discussion

A. Performance Comparison

We first evaluated the performance of our method with the first data set, which was formed by splitting the COG proteins that are part of the Swiss-Prot database into training and validation sets. The validation set is fed into the algorithm

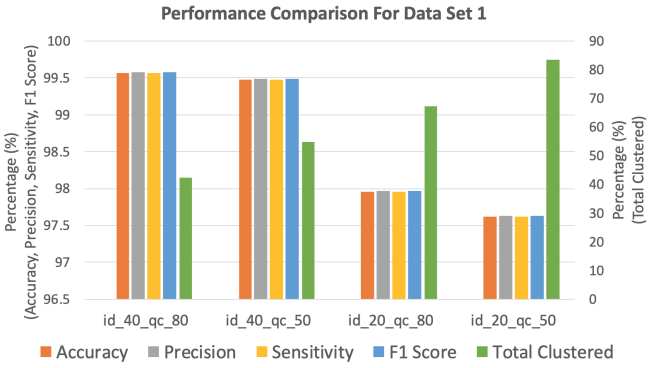

Fig. 5. **Performance Comparison For Data Set 1.** Both y-axes shows the percentage but in different scales. The left axis is for accuracy, precision, sensitivity, and F1 score (performance measures), whereas the right axis is for the total number of clustered sequences. "id_40_qc_80" stands for 40% sequence identity and 80% query coverage.

in one batch and the model's performance is measured for different parameters explained in Part C of the Methods section. The accuracy, precision, sensitivity, and F1 scores are recorded for each different parameter (Fig. 5). The lowest accuracy of 97.62% is observed for 20% sequence identity and 50% query coverage. It is observed from Fig. 5 that, while loosening the parameters leads to a slight decrease in the accuracy, precision, sensitivity, and F1 scores; it results in a significant increase in the total number of clustered sequences.

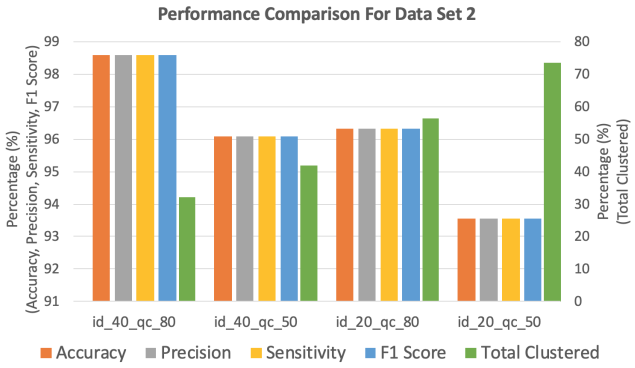

Fig. 6. **Performance Comparison For Data Set 2.** Both y-axes shows the percentage but in different scales. The left axis is for accuracy, precision, sensitivity, and F1 score (performance measures), whereas the right axis is for the total number of clustered sequences. "id_40_qc_80" stands for 40% sequence identity and 80% query coverage. It is observed that the performance measures decrease when loosening the parameters for the pairwise sequence alignment. The total number of clustered sequences, on the other hand, increases when loosening the thresholds.

In order to further evaluate the performance of our method with larger data sets, we proceeded with the second data set. The algorithm is initially trained with 117,747 COG proteins that are part of the Swiss-Prot. Then, 2,938,741 COG proteins that are part of the TrEMBL are fed into the algorithm in one batch. The accuracy, precision, sensitivity, and F1 scores are recorded for each parameter (Fig. 6). The best accuracy (98.58%), precision (98.59%), sensitivity (98.58%),

and F1 score (98.59%) are observed when the parameters for the pairwise alignment were strictest: 40% sequence identity and 80% query coverage. The lowest of these measures are observed with 20% sequence identity and 50% query coverage: 93.55% accuracy, 93.55% precision, 93.55% sensitivity, and 93.55% F1 score. While loosening the thresholds decreases the overall performance, it increases the total number of clustered sequences. Given this trade-off between the performance and the total clustered, it is up to the end user to decide which parameters to use.

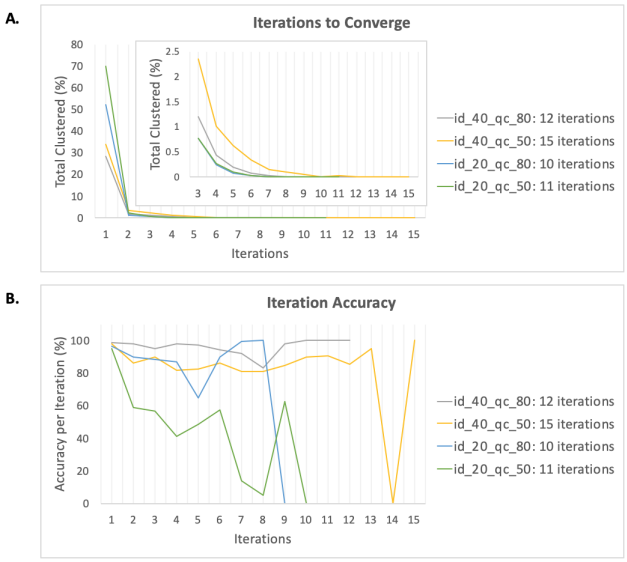

Fig. 7. **Performance Comparison. Part A: Iterations to Converge.** The number of iterations it takes for the algorithm to converge when run with different parameters for the pairwise sequence alignment. The y-axis shows the additional percentage of sequences that were clustered in each iteration. The inner graph shows the number of iterations vs total clustered starting from the third iteration. Most of the sequences get clustered at the first step. The cumulative percentage of clustered sequences at the first step and at the last step are as follows: 28.47%/32.19% for id_40_qc_80, 33.83%/41.87% for id_40_qc_50, 53.89%/56.48% for id_20_qc_80, and 70.10%/73.43% for id_20_qc_50. **Part B: Iteration Accuracy** The accuracy of the predictions for each iteration. Highest accuracies are observed at the first iteration.

It is mentioned in the methods section that the algorithm repeats steps in Fig. 3 Part B until it cannot cluster any more sequences. This is to maximize the number of predictions for a given batch of data. Fig. 7 Part A shows the additional percentage of clustered sequences in each iteration, whereas the Part B shows their clustering accuracy. The percentage of sequences clustered in each iteration is calculated as:

$$\frac{\#\_of\_Sequences\_Clustered\_in\_n^{th}\_iteration}{Total\_\#\_of\_Sequences} * 100 \quad (5)$$

It can be observed from Fig. 7 that each parameter takes different amount of iterations to converge. Most of the sequences are clustered at the first iteration. When the algorithm is run at 40% sequence identity and 80% query coverage, 28.47% of the sequences are get clustered at the first iteration. By the end of the last iteration (12), the percentage of clustered sequences increases to 32.19%. Moreover, the accuracies are the highest at the first iteration.

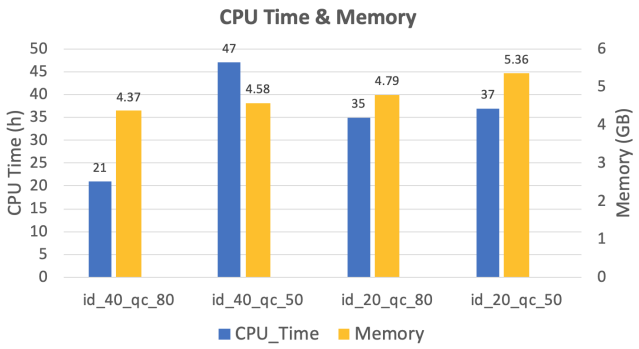

Fig. 8. **CPU Time & Memory Required for Convergence: Offline Usage.** The figure illustrates the memory usage and the CPU time required to run the algorithm until convergence. The minimum usage occurs at 40% sequence identity and 80% query coverage. The majority of memory usage is induced by the pairwise alignment, whereas the majority of the CPU time is used by the multiple sequence alignment and the recomputation of the consensus sequence for the next iteration (Fig. 3 Part B).

We further evaluated the performance of our algorithm in terms of computational usage. Fig. 8 shows the memory usage and the CPU time required for the algorithm to converge. The run with 40% sequence identity and 80% query coverage has the minimum usage and the maximum accuracy. It is important to note that the computational usage illustrated in Fig. 8 is for all the iterations. The part that consumes most of the memory is the pairwise sequence alignment, whereas most of the CPU time is spent during multiple sequence alignment and the computation of the consensus sequence for the next iteration (Fig. 3 Part B). Thus, we offer our program for two different uses: online and offline. Offline analysis employs the same algorithm presented in Fig. 3 Part B. For offline applications, users may decide to run the algorithm either until convergence or for a given number of iterations. This is the case where the goal is the continual learning of protein functions.

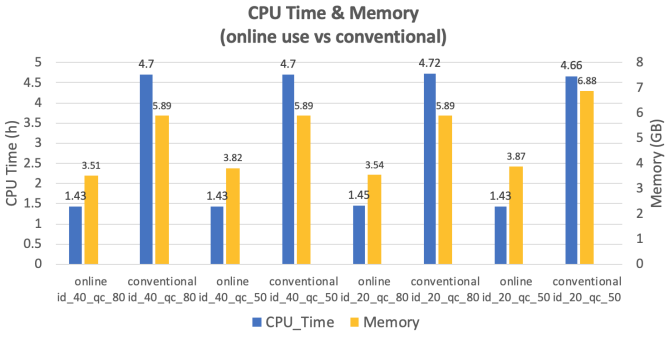

Fig. 9. **CPU Time & Memory: Online vs Conventional.** "id_40_qc_80" stands for 40% sequence identity and 80% query coverage. Conventional methods to determine protein functions rely on the pairwise alignment of the query sequences against an entire set of reference sequences. The use of the consensus sequences as a reference database reduces the search space for the alignment resulting in much less memory and CPU usage. It can also be observed from both figure 8 and 9 that the CPU and memory usage proves to be lower for the online analysis compared to the offline usage.

For users that desire a fast and reliable way of determining

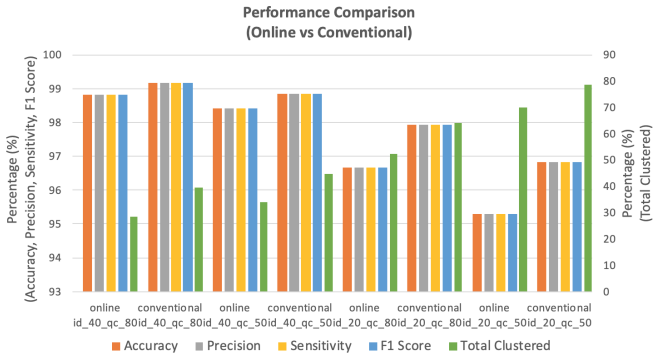

Fig. 10. **Performance Comparison: Online vs Conventional.** "id_40_qc_80" stands for 40% sequence identity and 80% query coverage. Our online analysis method offers similar accuracy, precision, sensitivity, and f1 score to the conventional methods which rely on the pairwise alignment of the query sequences against an entire set of reference sequences. In terms of total number of clustered sequences, however, the conventional method outperforms the online analysis by 8% difference.

the functions for their proteins, we offer an online analysis. When the goal of the user is to get the functional annotations for their proteins, online analysis offers a faster alternative to the conventional methods which rely on performing a pairwise alignment against an entire reference database (Fig. 9). Our method —the online analysis— offers a significant speed-up while showing a similar performance to conventional methods in terms of accuracy, precision, sensitivity, and f1 score (Figs. 9, 10). The online analysis proves to be less demanding for computational power as well. It has a lower memory usage and requires 3 times less CPU time (Fig. 9). The reduction in CPU time and memory comes from the compact representation of the protein sequences in the reference database. The use of the consensus sequences for the online analysis reduces the search space for the pairwise alignment, thus resulting in computational savings. The only drawback, however, is that it clusters around 8% less sequences compared with the conventional methods (Fig. 10).

Compared to the offline usage, the online analysis offers significant speed-up by avoiding many iterations, multiple sequence alignment, and the recomputation of the consensus sequence. However, it's important to note that the online analysis is not incremental, whereas the offline usage offers continual learning of protein functions thanks to its incremental nature. Compared to the offline application, the online analysis is up to 32 times faster in terms of CPU hours. The memory usage is also lower in the online analysis (Figs. 8 and 9).

B. Simulation of Incremental Learning

To simulate a real-world scenario for incremental learning, we split the validation sequences in 5 batches and fed the algorithm one batch at a time. Each batch contains 587,000 protein sequences (2,935,000 protein sequences in total). When a batch of data is introduced, the algorithm clusters protein sequences until it reaches convergence (Fig. 3 Part B). Any sequence that was not clustered at the end of the last iteration

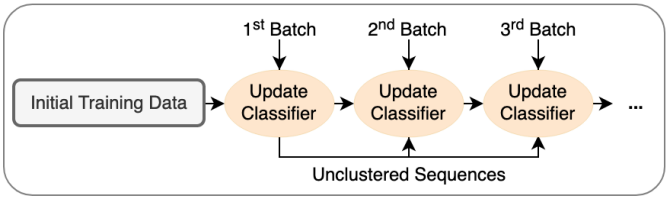

Fig. 11. **Incremental Learning of Protein Sequences.** Since our algorithm is specifically designed for incremental learning, it can continually predict protein functions as more genomic data becomes available. When the algorithm is run with a batch of data, it clusters all the sequences that were aligned against a consensus sequence (Fig. 3 Part B). Any sequence that was not clustered is reintroduced to the algorithm with the next batch. Since the algorithm updates the centroids each time a batch of data is introduced, unclustered sequences from previous batches may get clustered with the next batches.

is merged with the next batch of data and reintroduced to the algorithm (Fig. 11). For that reason, the total size of each batch is larger than its original size— Batch 1: 587,000; Batch 2: 985,184; Batch 3: 1,380,382; Batch 4: 1,745,938; Batch 5: 2,135,513. The incremental learning scenario is run with the best performing parameters: 40% sequence identity and 80% query coverage (Figs. 6, 8).

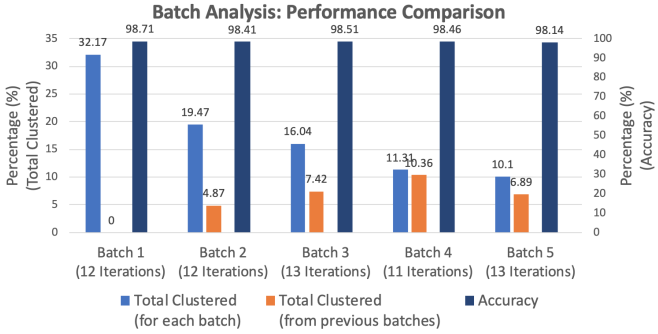

Fig. 12. **Performance Comparison for Each Batch.** Results for incremental scenarios run with 40% sequence identity and 80% query coverage. Both y-axes show the percentage but in different scales. The left axis is for total clustered for each batch and total clustered from previous batches. The right axis is for the accuracy. The number of iterations required for each batch is shown. The accuracy remains at 98% through all batches. As more data is poured in, the consensus sequences are modified based on the new data and thus the algorithm is able to predict functions for proteins who are not clustered at the previous batch. At the end of the fifth batch, a total of 1,015,163 sequences are clustered out of 2,935,000 sequences. That number accounts for 34.59% of the sequences fed into the algorithm. It can be inferred from Fig. 6 that the percentage of the sequences that are clustered remains similar in incremental and non-incremental versions.

The results for the simulation of incremental learning show that our algorithm is able to continually predict protein functions with a high accuracy. It can be observed from Fig. 12 that the accuracy remains at 98% through all batches. This shows that even though an increasing number of sequences are used to modify the consensus sequence, the modifications of the consensus are meaningful (Fig. 2) since the changes in the consensus result in more predictions without having a negative impact on the accuracy.

It is observed from Fig. 13 that the maximum memory usage of 2.64 GB occurs at the fifth batch. Incremental learning

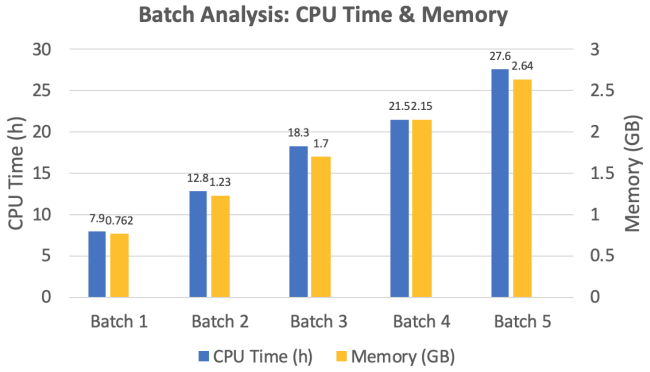

Fig. 13. **CPU Time & Memory for Each Batch.** As the batch size grows, the computational usage increases. The maximum memory used by the incremental version is 1.65 times less than the non-incremental version (Fig. 8). The CPU time, however, is much longer for the incremental version due to the multiple iterations performed for each batch. The incremental version took a total 61 iterations while the non-incremental version took only 12. As the number of iterations gets bigger, the algorithm spends more time on the multiple sequence alignment and the recomputation of consensus sequences—the parts that consume most of the CPU time.

of protein sequences offers 1.65 times less memory usage compared with the non-incremental version (Fig. 8). The total CPU time, on the other hand, is higher than the non-incremental version. This is due to the multiple iterations performed for each batch. While the incremental version took a total of 61 iterations, the non-incremental version took only 12 iterations (Figs. 7, 13). As the number of iterations gets bigger, the algorithm spends more time performing multiple sequence alignment and computing the consensus sequences—the parts that consume most of the CPU time (Fig. 3 Part B).

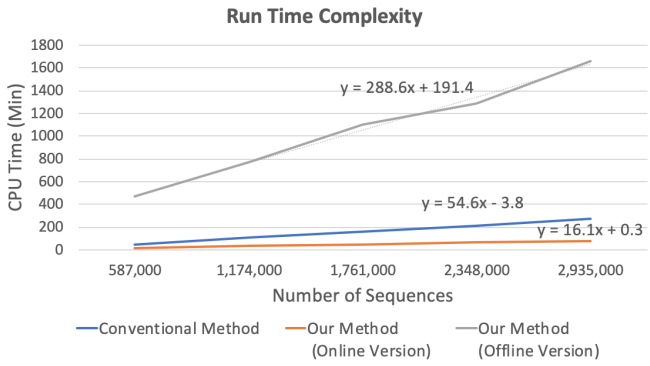

Fig. 14. **Run Time Complexity.** The comparison of the run time complexities. The conventional method represents the case where the query sequences are aligned against an entire set of reference database. The online version of our method reduces the search space for the alignment by compressing the sequences in the reference database into a set of consensus sequences. Thus, the online version has the least run time complexity. The offline version, our implementation of incremental learning, has the highest run time complexity.

The Fig. 14 illustrates the run time scales of different methods. The online version achieves a lower time complexity compared to both the conventional method and the offline version by performing only the pairwise alignment of query sequences against a set of consensus sequences. While the conventional methods rely on the alignment of the query sequences against an entire reference database, the online version achieves a similar accuracy (Fig. 10) by using a compact representation of protein groups. Given the lower run time complexity and similar performance of online version (Fig. 10), it may be an alternative to the conventional methods for large protein data sets in terms of reduced memory usage, and reduced CPU time. The offline version of our method, on the other hand, proves to have the highest run time complexity. Due to the semi-supervised nature of our algorithm, the consensus sequences are updated based on the newly classified protein sequences in each iteration. Performing multiple sequence alignment for each updated consensus groups and recomputing the consensus sequences result in a higher run time complexity for offline version (Fig. 3 Part B).

V. Conclusion and Future Work

The transformation of the reference database into a set of consensus sequences reduces the computational usage required for the pairwise alignment. Thus, for online analysis it offers a faster alternative to the conventional methods.

For offline applications, the incremental learning method proves to have a high accuracy, while requiring a longer CPU time. Decreasing the number of iterations for each batch will result in computational savings. Considering that most of the sequences get clustered at the first iteration with an accuracy above 95% (Fig. 7), we may modify our algorithm in the future studies so that it stops after the first iteration.

The run time complexity of our algorithm is further constrained by the software used for multiple sequence alignment (MSA)—MUSCLE. Considering that the multiple sequence alignment (MSA) and the recomputation of the consensus sequences are the most time consuming parts, we may replace MUSCLE with faster aligners such as FAMSA [23]. Decreasing the number of consensus groups for each COG may also result in a faster execution of the algorithm by consequently reducing the number of multiple sequence alignments (MSA) required. Once these performance issues are addressed, the next step will be novelty detection. In order to generalize our method, it's crucial for it to be able to dynamically create novel groups for functionally similar proteins for which a functional group is not defined in the training set.

VI. Acknowledgements

This work was supported by hardware provided by Drexel University's Research Computing Facility (URCF).

References

[1] D. J. Selkoe, "Alzheimer's Disease: Genes, Proteins, and Therapy," Physiological Reviews, vol. 81, no. 2, pp. 741–766, 2001, doi: 10.1152/physrev.2001.81.2.741.

[2] S. Wu, "Progress and Concept for COVID-19 Vaccine Development," Biotechnology journal, vol. 15, no. 6, p. e2000147–n/a, 2020, doi: 10.1002/biot.202000147.

[3] R. D. Emes, "Inferring Function from Homology," Bioinformatics. Methods in Molecular Biology™, vol. 453, 2008, doi: 10.1007/978-1-60327-429-6_6

[4] MG-RAST: metagenomics analysis server. [Online]. Available: https://www.mg-rast.org/index.html?stay=1. [Accessed: 13-Aug-2021].

[5] S. F. Altschul and M. Pop, "Sequence Alignment," Handbook of Discrete and Combinatorial Mathematics, 2nd edition, Boca Raton (FL): CRC Press/Taylor & Francis, 2017, Chapter 20.1. Available: https://www.ncbi.nlm.nih.gov/books/NBK464187/. [Accessed: 15-Aug-2021].
[6] B. Buchfink, C. Xie, and D. H. Huson, "Fast and sensitive protein alignment using DIAMOND," Nature methods, vol. 12, no. 1, pp. 59–60, 2015, doi: 10.1038/nmeth.3176.
[7] B. Buchfink, K. Reuter, and H.-G. Drost, "Sensitive protein alignments at tree-of-life scale using DIAMOND," Nature methods, vol. 18, no. 4, pp. 366–368, 2021, doi: 10.1038/s41592-021-01101-x.
[8] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," Nucleic acids research, vol. 32, no. 5, pp. 1792–1797, 2004, doi: 10.1093/nar/gkh340.
[9] A. P. Davis et al., "Gene Ontology: tool for the unification of biology," Nature genetics, vol. 25, no. 1, pp. 25–29, 2000, doi: 10.1038/75556.
[10] "The Gene Ontology resource: enriching a GOld mine," Nucleic acids research, vol. 49, no. D1, pp. D325–D334, 2021, doi: 10.1093/nar/gkaa1113.
[11] R. L. Tatusov, E. V. Koonin, and D. J. Lipman, "A Genomic Perspective on Protein Families," Science, vol. 278, no. 5338, pp. 631–637, 1997, doi: 10.1126/science.278.5338.631.
[12] T. Gabaldón and E. V. Koonin, "Functional and evolutionary implications of gene orthology," Nature reviews. Genetics, vol. 14, no. 5, pp. 360–366, 2013, doi: 10.1038/nrg3456.
[13] M. Y. Galperin, Y. I. Wolf, K. S. Makarova, R. Vera Alvarez, D. Landsman, and E. V. Koonin, "COG database update: focus on microbial diversity, model organisms, and widespread pathogens," Nucleic acids research, vol. 49, no. D1, pp. D274–D281, 2021, doi: 10.1093/nar/gkaa1018.
[14] K. Wetterstrand, "DNA Sequencing Costs: Data", Genome.gov. [Online]. Available: https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data. [Accessed August 6, 2021].
[15] C. M. Cullen et al., "Emerging Priorities for Microbiome Research," Frontiers in microbiology, vol. 11, pp. 136–136, 2020, doi: 10.3389/fmicb.2020.00136.
[16] C. Loeffler, A. Karlsberg, L. S. Martin, E. Eskin, D. Koslicki, and S. Mangul, "Improving the usability and comprehensiveness of microbial databases," BMC biology, vol. 18, no. 1, pp. 37–37, 2020, doi: 10.1186/s12915-020-0756-z.
[17] Z. Zhao, A. Cristian, and G. Rosen, "Keeping up with the genomes: efficient learning of our increasing knowledge of the tree of life," BMC bioinformatics, vol. 21, no. 1, pp. 1–412, 2020, doi: 10.1186/s12859-020-03744-7.
[18] M. Y. Galperin, D. M. Kristensen, K. S. Makarova, Y. I. Wolf, and E. V. Koonin, "Microbial genome analysis: the COG approach," Briefings in bioinformatics, vol. 20, no. 4, pp. 1063–1070, 2019, doi: 10.1093/bib/bbx117.
[19] R. L. Tatusov, M. Y. Galperin, D. A. Natale, and E. V. Koonin, "The COG database: a tool for genome-scale analysis of protein functions and evolution," Nucleic acids research, vol. 28, no. 1, pp. 33–36, 2000, doi: 10.1093/Nar/28.1.33.
[20] R. L. Tatusov, E. V. Koonin, and D. J. Lipman, "A Genomic Perspective on Protein Families," Science, vol. 278, no. 5338, pp. 631–637, 1997, doi: 10.1126/science.278.5338.631.
[21] A. Bairoch and R. Apweiler, "The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999," Nucleic acids research, vol. 27, no. 1, pp. 49–54, 1999, doi: 10.1093/nar/27.1.49.
[22] B. Buchfink, "1. Tutorial - Diamond Wiki," GitHub. [Online]. Available: https://github.com/bbuchfink/diamond/wiki/1.-Tutorial.
[23] S. Deorowicz, A. Debudaj-Grabysz, and A. Gudyś, "FAMSA: Fast and accurate multiple sequence alignment of huge protein families," Scientific reports, vol. 6, no. 1, pp. 33964–33964, 2016, doi: 10.1038/srep33964.