Project report: Protein function prediction through Neural Networks

Fathy Shalaby[*]
Institute for Machine Learning
Johannes Kepler University Linz
Linz, A-4040
fathy.mshalaby@gmail.com

Abstract

The recent discovery of high-throughput sequencing technologies has resulted in a significant increase in the number of protein sequences available. The traditional methods used for protein annotations scale poorly and have a high cost. With the recent advances in the area of Computational Biology, many annotation approaches have been suggested to fill the gap. The annotation of protein sequences is typically performed through the usage of Gene Ontology (GO), which contains over 40000 classes. Furthermore, protein sequences have multiple functions, making the annotation of these sequences a large-scale, multi-scale, multi-label problem. In this project, I applied a multi-layered one-dimensional Convolutional Neural Network (CNN) with a max-pooling layer and a softmax output to train a model that is able to perform protein function prediction. The method outputs the corresponding labels in the structure of the GO hierarchy, which contains all dependencies between the GO classes to annotate the sequence. The model was evaluated by balanced accuracy and area under the receiver operating characteristic curve (AUC ROC). The model scored an average balanced accuracy of 0.6 and an average area AUC ROC of 0.575, a maximum of 0.762, and a minimum of 0.464.

Implementation: The code source code is available at
https://github.com/fathyshalaby/BioinformaticsProject.git

1 Introduction

The recent advances in sequencing technology have resulted in a rapid increase in the number of protein and genetic sequences. With the continuous advances in this area and more sequences of more organisms becoming available, it is expected that sequencing data will increase further. Even though the knowledge of protein sequences is useful for many applications in the area of Biology, such as evolutionary biology and phylogenetics, which requires knowledge about the protein function (Lee et al., 2017). In this case, to annotate a protein sequence requires much time and resources.

One new prominent method to solve the stated problem is through computational predictions. One of the methods which have shown excellent results recently are Machine Learning methods more specifically, Neural Networks.

In this report, I explain and discuss the attempt to use machine learning to predict protein function by annotating protein sequences through computational predictions. Firstly, the various data sources used for this project will be discussed. In subsection 3.1, the chosen deep learning architecture used in the project (Convolutional Neural Network) is introduced. Explain the mathematical basis of the

[*]BSc Student at Institute for Machine Learning, 5th semester

method through a visual example of the method. Next, I discuss the training procedure in subsection 3.2, the model, the loss function, and the optimizer. Finally, I discuss the findings, compare this project to others, and discuss any future work that has resulted from this in sections 4 and 5.

2 Datasets

For this project, I use the Gene-Ontology (GO), downloaded on 18 February 2018[1] in OBO format (Gene & Consortium, 2000). GO has an acyclic graph structure[2] GO has three major domains, biological processes[3], molecular functions[4], and cellular component[5], each containing 30817, 12133 and 4407 classes, respectively. Due to the limited computing resources and time, I will only focus on the biological processes domain as they contain the most classes (Gene Ontology Consortium, 2004) as visualized in Figure. 5 in Appendix A.

Uniprot References Clusters (UniRef) with 50 percent sequence identity downloaded on 18 February 2018[6] was used. In the data set, 15306 sequences correspond to relevant GO-IDs. The maximum length of sequences was 34500, a median of 415, and a mean of 516 amino acids, as shown in Figure. 6 in Appendix A.

2.1 Data representation

The input of the model is the amino acid (AA) sequence of a protein. Each protein is a character sequence composed of 25 unique AA codes (which include five ambiguous amino acids). Inputs are represented through a matrix where the columns correspond to the amino acid and the rows to the position of the amino acid in the protein sequence, also known as a feature vector. However, due to the model's nature of working with fixed input shapes, shorter inputs are zero-padded to have the same length of 35000 to include all sequences in the data set.

3 Methods

3.1 Neural Network Architecture

Convolutional Neural Networks (CNN) are a class of deep neural network which is a biologically inspired Neural Network. CNN has been proven effective in computer vision tasks such as image classification, due to its ability to extract high-level patterns.CNN typically contains an input layer, output layer, and multiple hidden layers that generally consist of convolutional layers, RELU activation function, pooling layers, fully connected layer, and normalization layer. The convolution operation on CNN usually is applied to the input layer to extract features. The process is through the exploitation of local correlation by enforcing local connections between neurons in adjacent layers. Each region of the input is connected to a neuron in the hidden layer which are then connected to the output layer. Therefore, when applying multiple convolutional filters, the model can learn multiple features and provide multiple facets of the data (O'Shea & Nash, 2015). During this project, a three-layer one-dimensional convolution with a max-pooling layer is applied over the protein sequences. The reason for using CNN model is due to patterns in the protein sequences. Furthermore, this architecture works with images; it should, in theory, perform well on multi-class classification tasks.

Below I show a small example of the steps of the model in the form of a 1-D Convolution, where X is the input, W the kernel. Which the output is then feed into a max-pooling layer[7]is applied. :

[1]http://purl.obolibrary.org/obo/go/go-basic.obo

[2]A finite directed graph with no directed cycles

[3]The 'biological programs' accomplished by multiple molecular activities

[4]Molecular-level activities performed by gene products

[5]The locations relative to cellular structures in which a gene product performs a function.

[6]ftp://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref50/uniref50.fasta.gz

[7]A downsampling strategy used in Convolutional Neural Networks, where the largest value in a predetermined window is chosen to represent the specific part of the input. This allows the network to learn more efficiently due to the simplified representations.

$$\mathbf{f} = (1 \quad 3 \quad 3 \quad 0 \quad 1 \quad 2)$$
$$\mathbf{h} = (2 \quad 0 \quad 1)$$
$$\mathbf{f} * \mathbf{h} = (1 \quad 3 \quad 3 \quad 0 \quad 1 \quad 2) * (2 \quad 0 \quad 1)$$
$$= ((2+0+3) \quad (6+0+0) \quad (6+0+1) \quad (1+0+1))$$
$$= (5 \quad 6 \quad 7 \quad 2) \xrightarrow{max-pooling} (7)$$

The Convolution step can be mathematically represented as the following.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \tag{1}$$

where f is the input image and h is the kernel which is used to modify the input.The indexes of rows and columns of the result matrix are marked with m and n respectively. A Table with the model structure and parameters used can be found under Appendix A subsection 6.4.

3.2 Training procedure

First, annotations are propagated using the GO ontology structure and randomly split proteins into a training set (70%), testing set (15%), validation set (15%) containing 8950, 2986, and 2987 sequences respectively. Due to computational limitations and the small number of annotations for particular GO classes, GO classes were ranked by their number of annotations and GO classes which had a minimum of 1000 annotations were chosen as evident Figure.1.

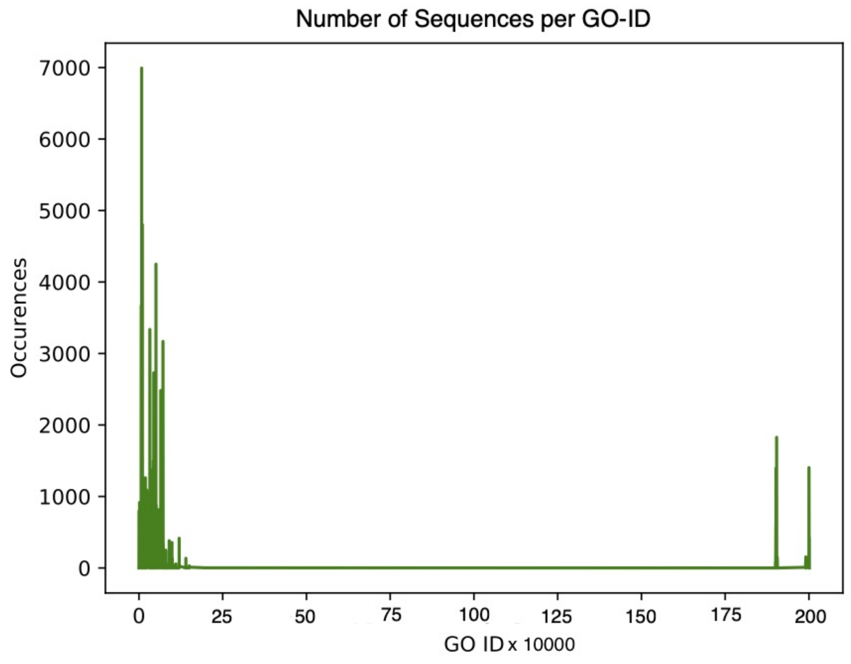


Figure 1: An imbalanced class distribution is evident by the two clusters in range 0-20 and 180-200, due to the hierarchical nature of the classes.

DDuring training, binary cross-entropy logistic loss was used, which is a combination of sigmoid and binary cross-entropy, with a learning rate of 0.001. This rate is one of the recommended learning rates when using ADAM optimizer (Ding et al., 2015). Furthermore, to minimize the chance of overfitting, a weight decay of 1e-5 was used. The loss can be described mathematically as:

$$\ell(x,y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1-y_n) \cdot \log(1-\sigma(x_n))\right] \tag{2}$$

where N is the batch size. This is used for measuring the error of a reconstruction in for example an auto-encoder. Note that the labels need to be between 0 and 1. It's possible to trade off recall and precision by adding weights to positive examples. In this case the loss can be described as:

$$\begin{gathered}\ell_c(x, y) = L_c = \{l_{1,c}, \dots, l_{N,c}\}^\top, \\ l_{n,c} = -w_{n,c}\left[p_c y_{n,c} \cdot \log \sigma(x_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - \sigma(x_{n,c}))\right]\end{gathered} \tag{3}$$

where c is the class number (c > 1 for multi-label binary classification,c = 1 for single-label binary classification),n is the number of the sample in the batch,w_n is the weight,p_c is the weight of the positive answer for the class c and where p_n is the positive weight of class n. For example, if a dataset contains 100 positive and 300 negative examples of a single class, then the positional weight[8] for the class should be equal to $\frac{300}{100} = 3$. The loss would act as if the dataset contains $3 \times 100 = 300$ positive examples.

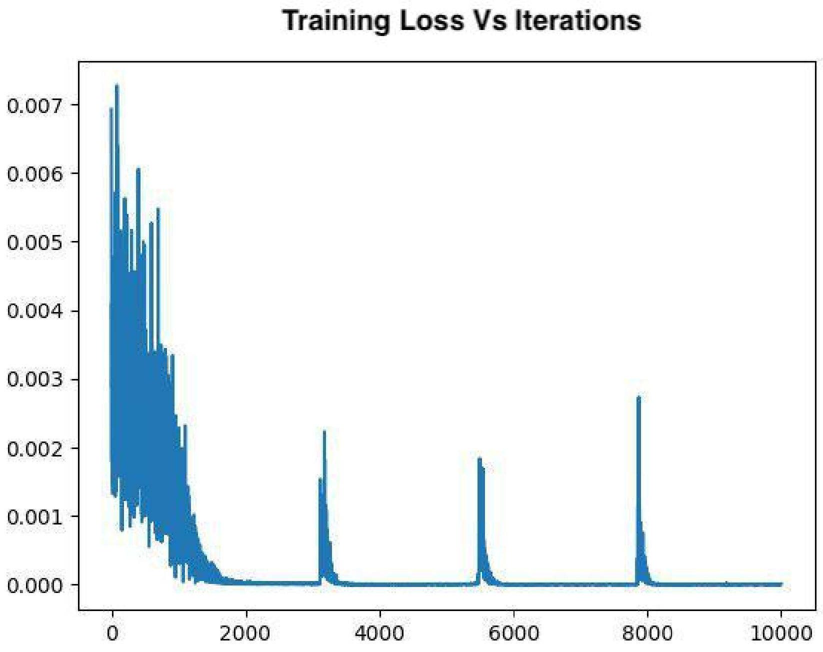


Figure 2: Loss on training set with the learning rate = 0.001, L2 weight decay =0.00001 for 10000 updates over all classes which took approximately 6 hours on a i7 7th generation CPU. X-Axis = =Iteration, Y-Axis = Loss/10000

[8]The weight assigned to each class, commonly also seen as the probability of the class in the dataset.

4 Results

For evaluating the model, two main evaluation metrics were used. Firstly balanced accuracy was used to evaluate the accuracy of the model on data sets imbalance in positive and negative samples for some of the classes as discussed in section 4. The model scored an average balanced accuracy score of 0.5. The second metric was area under the receiver operating characteristic curve (ROC AUC) score was measured with an average score of 57.5% as illustrated in Figure. 3. I can see that

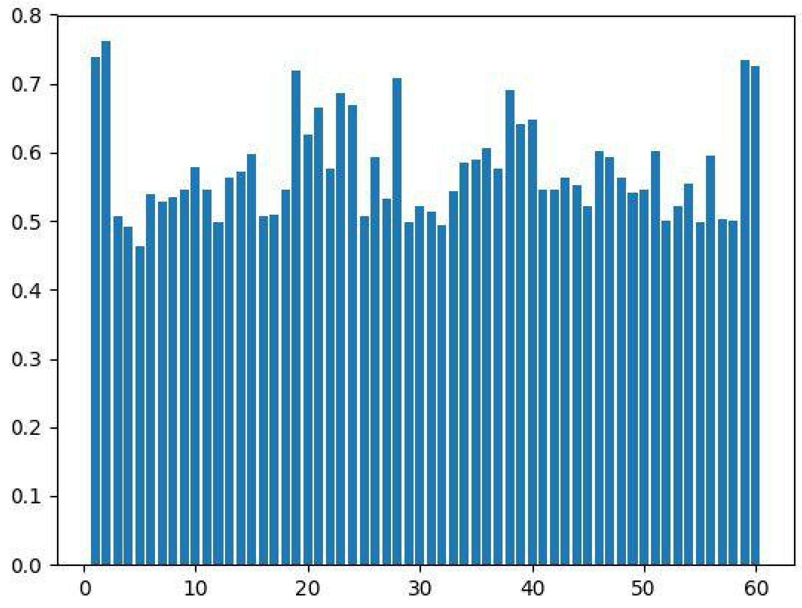


Figure 3: ROC AUC scores (y-axis) distribution based on the different classes (x-axis) in the test set. mean = 0.575 min = 0.464 (class 5), max = 0.762 (class 2)

from the 60 classes, and there are few classes that show high ROC AUC scores (better than random (ROC AUC > 0.5)) and some of which show low scores (random or worse (ROC AUC =< 0.5)), which may be due to some class imbalance in the dataset. When running the model on the test set which is used to evaluate my model on it's ability to predict on unknown data, in Figure. 4 (Left). The model achieves an average ROC AUC score of 0.567. Figure. 4 (right).

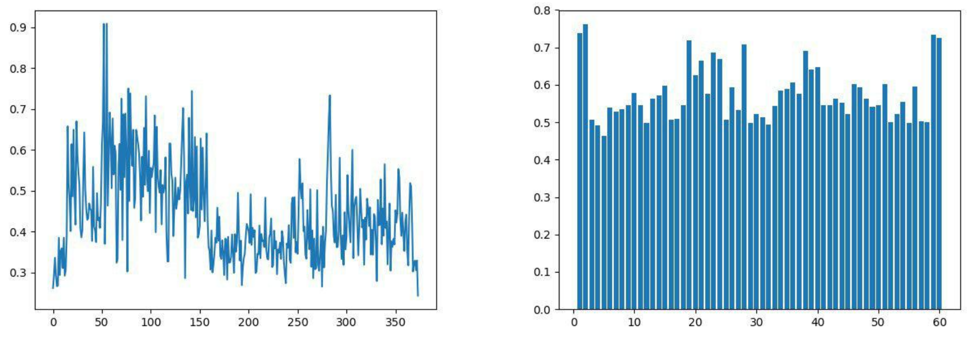


Figure 4: Left: Loss on the validation set for every sequence in the validation set. Right: ROC AUC scores (y-axis) distribution based on the different classes (x-axis) in the validation set. mean = 0.567, min = 0.475 (Class 5), max = 0.742 (class 2)

5 Discussion

During this project, I created a classifier which displayed a low training-loss and computed an average ROC AUC scores on the testing set, as shown in Figure. 2, 3, 4. However, I do encounter two primary problems with the classifier. Firstly, due to the imbalanced nature of the dataset, the classifier is not able to score as expected compared to published models literature. Next, during training, it can be observed that the model is overfitting on the data, which indicates that the learning rate and other parameters used are not optimized for this task, which may have been the case in my project. These findings may explain as an example, the fluctuating training loss-scores evident in Figure. 2. As mentioned previously in section 4 results, I did observe the distribution of unbalanced scores on the classes used for this project was unbalanced as some classes received higher ROC AUC scores than other classes, which may have been a direct result of the imbalanced distribution of training examples for specific classes. Looking at the loss during training, we can also see that the loss starts to stay constant after 2000 iterations, which may be due to the variation in sequence lengths, the zero-padding step during prepossessing, or the size of the model. However, it can be observed that the model predicts specific classes better which indicates that the model is maybe too small and as a result cannot distinguish specific classes from each..

The negative effect occurs as a result of both the lack of filter regarding the maximum allowed sequence length and the non-complex nature of the model as as evident in Figure. 6. Furthermore, I do observe that there is a difference in ROC AUC scores between the classes, as shown in Figure. 3, even though a pooling layer is used, which further supports the theory that the dataset is very imbalanced, thus resulting in the values presented in section 4. Using a shorter sequence length, instead of the maximum length would have also helped the model perform better. Furthermore, the model was able to detect some signal in the data, suggesting a more rigorous study with better model design and setup and filtering in the future would yield a better performance. Moreover, (Kulmanov et al., 2018) explored the idea of using both the Hierarchical Class structure and protein-protein interaction and applied on a similar dataset resulting in an average ROC AUC score of 0.89. However, in the study, the authors do also mention the drawbacks of their model, which includes, as previously mentioned, the drawbacks of using a CNN on data with variational length and how zero-padding greatly contributes to the decrease in their model performance. Moreover, (Hochreiter et al., 2007) applied a Long Short Term Memory neural network (LSTM) to a well-known benchmark for remote protein homology detection, where a protein must be classified as belonging to a SCOP superfamily. Their LSTM were five magnitudes faster than traditional approaches and two magnitudes faster than SVM based approaches and scored much better than traditional methods with an average ROC AUC score of 0.98. This may indicate that the usage of LSTM in this domain may be better.

In conclusion, a three-layered one-dimensional CNN with a max-pooling layer was used to predict corresponding sequence labels in the form of GO-classes. By using a feature vector, which represented the inputs as an nxk matrix where n is the number of amino acids and the k is the maximum sequence length in the dataset. Even though the model's performance was not as expected, it does show that CNNs could be used for this task with the addition of more preprocessing. I plan to explore possible solutions that could solve the limitations discussed in this report. In future work, I intend to apply a Long Short Term Memory (LSTM) Neural Network. Their ability to accept multi-length sequence inputs, work with long-term dependencies, and design, which makes it not to be affected by the vanishing gradient problem, would solve some of the limitations discussed (Hochreiter & Jürgen Schmidhuber, 1997). Finally, I plan in future work to explore the prediction of one class and using other classes as auxiliary classes, which would allow the model to perform better. Furthermore, using larger models may further improve the performance on the discussed dataset. Finally, as previously discussed, the application of LSTMs on this task may yield better results in theory, which I plan to further explore in future work.

References

Ding, S., Zhao, H., Zhang, Y., Xu, X. & Nie, R., 2015. Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1), pp.103–115.

Gene Ontology Consortium, 2004. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(90001), pp.258D–261. Available at: <https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkh036> [Accessed 2018-11-23].

Gene, T. & Consortium, O., 2000. Gene Ontology : tool for the. *Gene Expression*, 25(may), pp.25–29. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/10802651> [Accessed !no date!].

Hochreiter, S. & Jürgen Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780. Available at: <http://www7.informatik.tu-muenchen.de/{~}hochreit{%}0Ahttp://www.idsia.ch/{~}juergen> [Accessed 2018-10-19].

Hochreiter, S., Heusel, M. & Obermayer, K., 2007. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14), pp.1728–1736.

Kulmanov, M., Khan, M. A. & Hoehndorf, R., 2018. DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4), pp.660–668.

Lee, A. Y., Agaram, N. P., Qin, L.-x., Kuk, D., Curtin, C., Brennan, M. F. & Singer, S., 2017. A global interaction network maps a wiring diagram of cellular function A. , 23(3), pp.818–825.

O'Shea, K. & Nash, R., 2015. An Introduction to Convolutional Neural Networks. , (December). Available at: <http://arxiv.org/abs/1511.08458> [Accessed 2019-03-19].

Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R. & Wu, C. H., 2007. UniRef: Comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, .

6 Appendix A

6.1 Gene-Ontology (GO)

For this project , the Gene-Ontology (GO) Database was used to acquire the labels for the computational predictions. GO is a bioinformatics database which its goal is to unify the representation of gene and gene product attributes across all species.

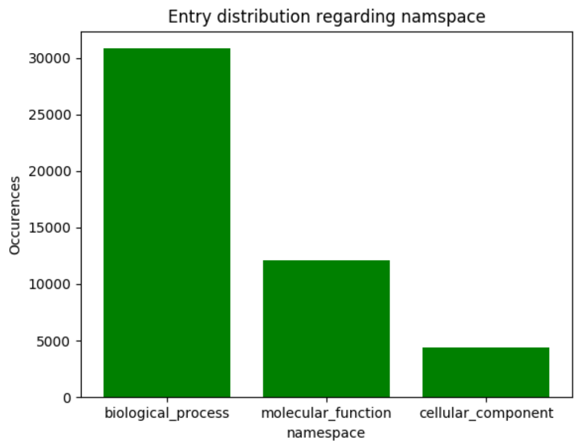
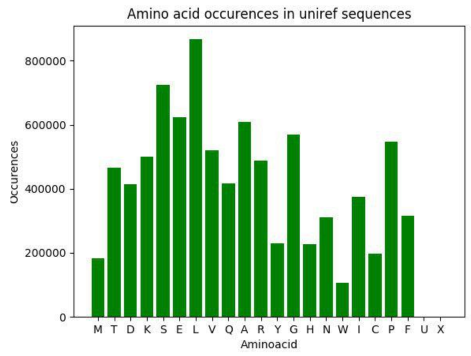


Figure 5: Name-space distribution in Gene-Ontology on 18 February 2018.

6.2 UniProt References Clusters (UniRef)

For this experiment, UniProt Reference Clusters is used to obtain the protein sequences which will be annotated. The UniProt Reference Clusters (UniRef) provide clustered sets of sequences from the UniProt Knowledgebase (UniProtKB) and selected UniProt Archive (UniParc) records in order to obtain complete coverage of the sequence space at several resolutions while hiding redundant sequences (but not their descriptions) from view (Suzek et al., 2007). UniRef50 is built by clustering UniRef100 sequences at the 50 percent sequence identity levels. UniRef entries typically contain summary cluster and membership information, including the sequence of a representative (best-annotated) protein, member count and common taxonomy of the cluster, the accession numbers of all the merged entries, and links to a rich functional annotation in UniProtKB to facilitate biological discovery (Suzek et al., 2007). The UniRef50 contains 15306 relevant sequences that correspond to the UniProt IDs, which are taken from the UniProt GAF file. The Sequence maximum length was seen to be 34500, a median of 415, and a mean of 516 amino acids.



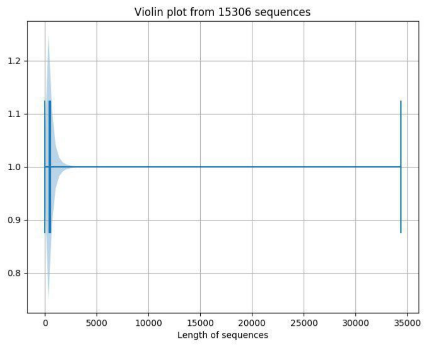


Figure 6: Left: Amino acid distribution in the Uniref50 file, used to determine the number of input features. Right: A Violin graph which displays the distribution of sequence length found in the sequence file. The Sequences maximum length was seen to be 34500, a median of 415 and a mean of 516.

6.3 Gene Association File (GAF)

Gene Association Files (GAF) are tab-delimited plain text files, where each line in the file represents a single association between a gene product and a GO term, with an evidence code, the reference to support the link between them, and other information, which is comprised of 17 tab-delimited fields.

Field	Description
DB	Refers to the database from which the identifier in DB object ID (column 2) is drawn
DB Object ID	A unique identifier from the database in DB for the item being annotated.
DB object Symbol	A (unique and valid) symbol to which DB object ID is matched.
Qualifier	Flags that modify the interpretation of an annotation one
GO ID	The GO identifier for the term attributed to the DB object ID
DB: Reference	One or more unique identifiers for a single source cited as an authority for the attribution of the GO ID to the DB object ID.
Evidence Code	An evidence code to indicate how the annotation to a particular term is supported
With (or) From	This field is used to hold an additional identifier for annotations using certain evidence codes
Aspect	Refers to the namespace or ontology to which the GO ID belongs
DB Object Name	Name of gene or gene product
DB Object Synonym	Gene symbol
DB Object Type	A description of the type of gene product being annotated
Taxon	Taxonomic identifier
Date	Date on which the annotation was made; format is YYYYMMDD
Assigned By	The database which made the annotation.
Annotation Extension	Contains cross references to other ontologies that can be used to qualify or enhance the annotation.

6.4 Model Structure and Parameters

Table 1: Model structure used for this project with parameters included

Layer	Type	parameters
1	1-D Convolution+ReLU Activation	in_channels=22, out_channels=128, kernel_size=3
2	1-D Convolution+ReLU Activation	in_channels=128, out_channels=512, kernel_size=3
3	1-D Convolution+ReLU Activation	in_channels=512, out_channels=512, kernel_size=5
4	maxpooling	dimension=2
5	Linear	in_features=512, out_features= number of classes