Comparative Analysis of Transformer-based and Convolutional Neural Network Models for Protein Function Prediction Using Amino Acid Sequence Data

Andrew Liu & Vinay Swamy

Received May 13, 2024
Accepted August 27, 2024
Electronic access September 15, 2024

Proteins are essential building blocks to life that serve countless roles within the human body. Predicting protein function holds immense value in multiple domains, including drug discovery and disease understanding. Traditional wet laboratory methods, while precise, are time consuming and costly. This study focuses on sequence based protein function prediction by comparing a transformer-based model with pre-trained transformer layers to a convolutional neural network (CNN). Using a dataset of approximately 70,000 protein sequences from UniProt, we constructed and validated both models. The transformer-based model achieved an accuracy of 94.6% and an F1-score of 0.926, while the CNN model slightly outperformed it with an accuracy of 96.0% and an F1-score of 0.949. Despite the marginally lower performance, the transformer-based model showed more consistent accuracy across different classes, indicating greater robustness. We also acknowledge potential biases and suggest that training on a larger, more uniform dataset could improve both performances. Future research could explore an ensemble approach, combining both models to use their respective strengths. This work highlights the potential of using machine learning with biological research to drive significant advancements in protein engineering and healthcare.

Keywords: protein function prediction, large language models, convolutional neural network

Introduction

Proteins are fundamental building blocks of life, consisting of unique sequences of amino acids folded to have a three dimensional shape[1]. Proteins play a multitude of crucial roles in the human body and are responsible for the functions, structure, and regulation of cells and organs. These roles are tied to a protein's amino acid sequence and its structure. For example, enzymes, a class of proteins, aid in digestion and speed up chemical reactions in our bodies, while hemoglobin, another protein, transports oxygen and carbon dioxide in our bloodstream[2,3].

The accurate prediction of protein function is incredibly important in improving our understanding of biological processes and in addressing practical challenges such as drug discovery and disease treatment[4–6]. Current wet laboratory methods analyze protein 3D structures for accurate protein function predictions, but they are time-consuming and costly, and are not able to keep up with the pace at which new sequences are being discovered[7]. Because of this drawback, we have decided to solely use sequence data to predict protein function, which offers a much faster and more scalable method.

Traditionally, convolutional neural networks (CNN) have been used to predict protein function from amino acid sequences[8–10]. These networks utilize kernels and learn local context. Transformer based models, such as large language models (LLMs), have recently become very popular, due to their self attention mechanism which allows them to capture long range dependencies[11,12]. Because transformer based models also specialize in processing sequential data, we have decided to compare a CNN with a transformer based model, to determine if transformer based models should be used for protein function prediction.

For the purposes of this research, we have decided to incorporate pre trained transformer layers, which would save us computing resources and also enhance accuracy. We use the Evolutionary Scale Modeling (ESM) model, a transformer based protein language model developed by Meta AI's Fundamental AI Research Team[13]. Our methodology includes processing a dataset of approximately 70,000 protein sequences sourced from UniProt, constructing a model using the transformer layers of the ESM model, and comparing its results to an existing state of the art CNN model. The final validation process of the transformer based model results in an accuracy of 94.6% and an F1 score of 0.926, while the CNN model results 96.0% and an F1 score of 0.949.

Methods

Dataset

The dataset, comprising 69,991 proteins, was acquired from UniProt, a high quality database of protein sequence and functional information[14]. The amino acid sequences in this dataset all consist of at most 20 amino acids and each amino acid is represented with a character from the alphabet. We chose enzyme classification number (EC number) as the label, "a numerical classification scheme for enzymes" that is representative of the protein's function. EC numbers provide a hierarchical numerical classification scheme for enzymes, each uniquely categorized based on the chemical reaction they catalyze[15]. An EC number consists of four hierarchical levels: the main class, followed by two subclasses, and ending with a serial number, thus offering a standardized and concise way to represent protein functions.

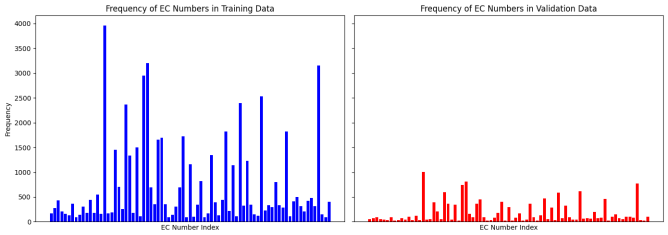

Fig. 1 displays the distribution of EC numbers within the training and validaiton data, shown on the left and right, respectively. Each bar in the bar chart represents an EC number, with the bar height indicating the frequency of the EC number in its respective dataset The similarity between the two distributions ensures generalization for the model.

Data Processing

We padded all sequences to length 256 and removed sequences with length more than 256. We selected a length of 256 because it covers 99.5% of the dataset and minimizes risks of overfitting to outlier sequences. We also removed EC numbers with frequency less than one hundred, to reduce the risk of not having enough data to learn these EC numbers. To get the training and validation datasets, we split the dataset such that 80% was the training dataset and the other 20% was the validation dataset. Although the distribution of EC numbers in the dataset is skewed and uneven, the distributions in the training and validation data are roughly similar (Figure 1). We still attempted to account for any potential biases by assigning a weight inversely proportional to frequency for each EC number when calculating loss. We did this by assigning the weight of an EC number to the reciprocal of its count, and then normalizing the weights by dividing all weights by the minimum weight.

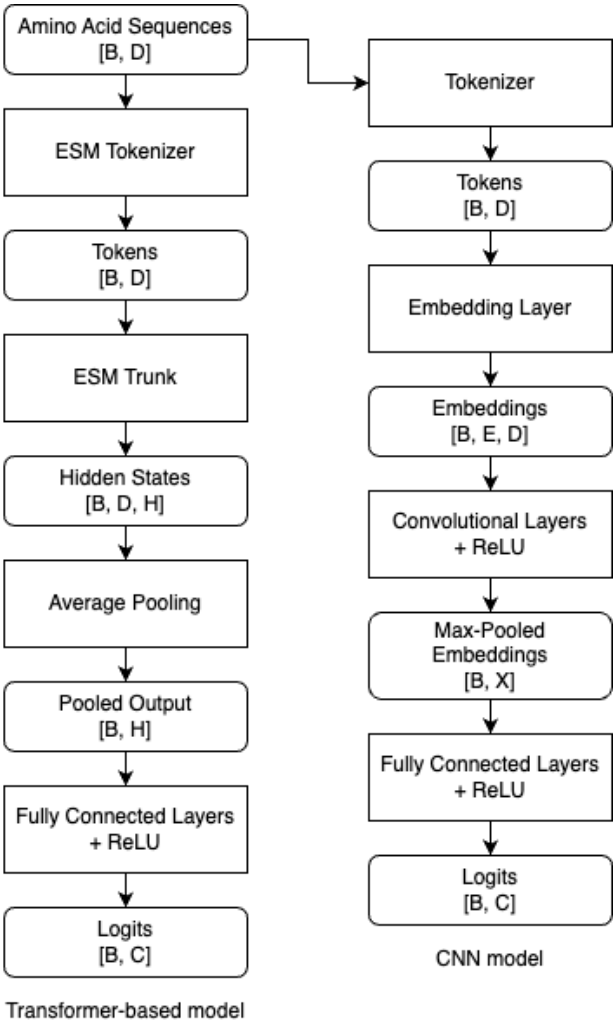

Fig. 2 depicts the workflow of protein function prediction for the transformer based model (left) and CNN model (right). For the transformer based model, the amino acid sequences are tokenized and processed by the transformer layers of the ESM model. The output embeddings are then passed through two fully connected layers with dimensions 200 and 128. For the CNN model, the amino acid sequences are similarly tokenized, then processed through multiple convolutional layers, followed by max pooling. The resulting features are flattened and concatenated, then passed through fully connected layers. Notations within the diagram include B, which denotes the batch size (8); D, which denotes amino acid sequence length (256); C, which denotes the number of EC numbers (79); H, the number of hidden states; E, the number of embedding dimensions; and X, the dimension after passing through convolution layers and pooling.

Model

The transformer based model is a custom built model that incorporates the ESM model's transformer layers. The tokenizer used is a built in tokenizer of the ESM model that is already pre trained, such that it recognizes the characters in the amino

acid sequences[14]. We use the ESM model as the trunk and implement our own head to fine tune the model for our specific representation of protein sequences (Figure 2). By doing so, we get a significantly more accurate model that is better fitted to our dataset. Our model also features two hidden linear layers with dimensions of 200 and 128, which we implemented to adapt to the classification head. The softmax activation function is used to output the classification prediction.

The CNN model used replicates the methodology of DeepGOPlus, a CNN model that predicts protein function from sequence. We use a series of increasing kernel sizes, which the paper claims to help with understanding short motifs in the protein sequences. The tokenizer uniquely maps each amino acid, represented with a character, to a number.

Loss

We used the cross entropy loss function for both models[16]. The equation for cross entropy loss is $L = -\sum_{i=1}^{m} y_i \log(\hat{y}_i)$

where m is the number of classes, y_i signifies the true probability distribution, and $\hat{y}_i$ stands for the predicted probability distribution. The cross entropy loss function is useful for our purposes as it quantifies the difference between the actual and predicted probability distributions, which we use to determine how closely fitted the model is to the dataset.

Optimizer

Optimizing the model's performance required adjusting model parameters to minimize the cost function. We use the Adam optimizer, which uses the gradient descent algorithm by iteratively updating parameters based on the gradient of the cost function[17]. The equation for the Adam optimizer is: $W_t = W_{t-1} - \frac{\eta}{\sqrt{v_t}+\varepsilon} m_t$ where the parameter values, represented by W_t, are adjusted from the previous parameter values based on a modified learning rate. This rate is determined by the first moment and adjusted by an estimate of the second moment.

Implementation

We used Hugging Face's transformers library for loading the ESMModel and tokenizer[18]. All code was written in Google Colab and ran using T4 GPU's.

Results

We found the most successful learning rate to be 0.001 for both models. We implemented an early stopping method such that the model would stop when there would be improvement less than 0.001 for the last five epochs. The transformer based model trained for forty two epochs and the CNN model trained for thirteen. The transformer based model achieved an accuracy of 94.6% and an F1 score of 0.925, and the CNN model achieved an accuracy of 96.0% and an F1 score of 0.949. During training, both models displayed significant decrease in loss in the first couple of epochs and the loss leveling off in the remaining epochs (Figure 3). The accuracy of both models shows a similar trend (Figure 4).

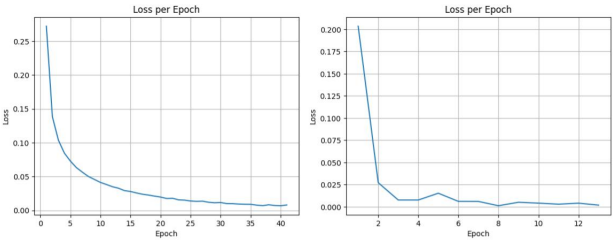

Fig. 3 illustrates the total loss per epoch during training for the transformer based model (left) and CNN model (right). Both models display a significant decrease in the first couple of epochs, after which the reduction rate begins to level off for the subsequent epochs. Both plots plateau towards the end.

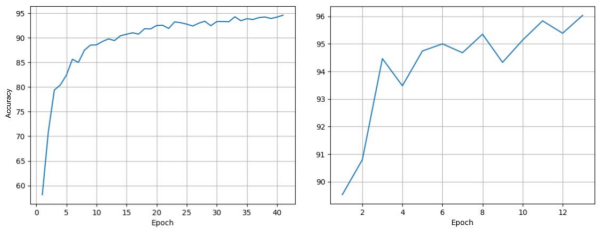

Fig. 4 displays the average accuracy per epoch during training for the transformer based model (left) and CNN model (right). The curve increases a lot in the first half of training for both plots and plateaus towards the end.

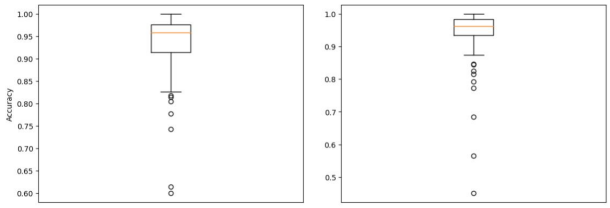

Fig. 5 shows a box plot of class accuracies for all EC numbers for the transformer based model (left) and CNN model (right). The median accuracy is represented by the orange line at 0.958 (left) and 0.963 (right). The first and third quartiles, located below and above the median, are at 0.915 and 0.978 (left) and 0.935 and 0.984, respectively, to yield an interquartile range of 0.063 (left) and 0.049 (right). The plots show seven outliers (left) and nine outliers (right).

To further assess the consistency of the two models, we analyzed each model's distribution of accuracy scores using box plots (Figure 5). The transformer based model had a median of 0.958 and an interquartile range of 0.063, indicating that the model performs well on most of the data. This model produced

seven outliers. The CNN model had a larger median of 0.963 and a smaller interquartile range of 0.049. The CNN model produced nine outliers.

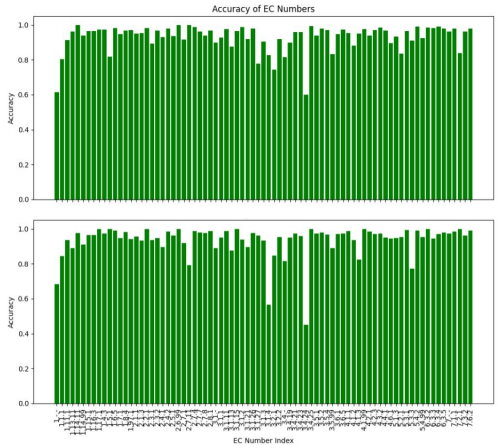

Fig. 6 demonstrates the class accuracies of EC numbers for the transformer based model (top) and CNN model (bottom), where each bar corresponds to an EC number and its height indicates the accuracy of predicting it. The distribution across the bars is relatively consistent for the transformer based model, but is inconsistent for the CNN model.

Overall, the CNN model outperformed the transformer based model in terms of mean and median accuracy, as well as a tighter interquartile range, indicating a more consistent performance across the majority of classes. However, the transformer based model excelled in maintaining a more stable and reliable performance, characterized by fewer outliers and a smaller range in class accuracies (Figure 6). This suggests that the transformer based model had a more robust performance across all classes, even though its average metrics were slightly lower.

The transformer based model and CNN model both had seven and nine outliers, respectively. The two models shared five outliers together, which were 1.-.-, 1.1.1, 3.2.1, 3.4.-, and 3.4.24. These EC numbers had accuracies ranging from 60% to 81.5% for the transformer based model and 45% to 84.4% for the CNN model. Interestingly, the other outliers, despite having poor accuracies for one model, had high accuracies for the other. For example, the transformer based model had an outlier 1.5.1 with an accuracy of 81.8%, but the CNN model had 100% accuracy. We speculate that the reason the five shared outliers have low accuracies is likely due to the low class counts, as all these outliers had below average counts. We also think that these protein sequences with outlier EC numbers might also have features that are hard to learn which inherently makes them hard to classify.

Discussion

We trained a transformer based model using the ESM model's transformer layers for protein function prediction and compared it to a current state of the art CNN model. The transformer based model performed slightly worse than the CNN model in terms of mean accuracy, median class accuracy, and interquartile range; however, the transformer based model had less outliers and had more consistent class accuracies, while the CNN model had some low class accuracies. In this regard, we think that the transformer based model had a more robust performance, even though its average metrics were slightly lower.

Acknowledging the limitations of these practices: despite the attempt to account for class imbalances with custom weights, there likely were still biases within the two models, due to the number of outliers. We also speculate that training on a larger dataset might produce better results, especially for the transformer based model which could produce far better results. However, this requires testing.

Moving forward, we may also exploit an ensemble model: a model that combines both models to output a prediction [19]. Through analyzing the outliers, we have seen that for some areas that the transformer based model showed weak accuracy in classification, while the CNN model showed strong accuracy, and vice versa.

In summary, the potential between machine learning and biological research holds immense promise, with potential applications ranging from refined protein engineering to transformative advancements in human health. As our understanding of both machine learning and biology continues to evolve, we can anticipate groundbreaking innovations at the intersection of artificial intelligence and life sciences.

References

1 M. Watford and G. Wu, *Advances in Nutrition*, 2018, **9**, 651–653.

2 P. K. Robinson, *Essays in Biochemistry*, 2015, **59**, 1–41.

3 M. H. Ahmed, M. S. Ghatge and M. K. Safo, *Sub-Cellular Biochemistry*, 2020, **94**, 345–382.

4 N. Borkakoti and J. M. Thornton, *Current Opinion in Structural Biology*, 2023, **78**, 102526.

5 R. Matheson, *Model learns how individual amino acids determine protein function*, 2019, `https://news.mit.edu/2019/machine-learning-amino-acids-protein-function-0322`.

6 T. K. Chaudhuri and S. Paul, *The FEBS Journal*, 2006, **273**, 113–1349.

7 C. J. Jeffrey, *Frontier Bioinformatics*, 2023, **3**, 1222183.

8 M. Kulmanov and R. Hoehndorf, *Bioinformatics*, 2020, **36**, 422–429.

9 F. V. Song, J. Su, S. Huang, N. Zhang, K. Li, M. Ni and M. Liao, *Briefings in Bioinformatics*, 2024, **25**, bbae196.

10 T. Sanderson, M. L. Bileschi, D. Belanger and L. J. Colwell, *eLife*, 2023, **12**, 80942.

11 A. Tam, *What are large language models*, 2023, https://machinelearningmastery.com/what-are-large-language-models/.

12 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention is all you need*, 2023, https://arxiv.org/abs/1706.03762.

13 Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. dos Santos Costa, M. Fazel-Zarandi, T. Ser-cu, S. Candido and A. Rives, *Science*, 2023, **379**, 1123–1130.

14 *Uniprot Database*, 2024, https://www.uniprot.org/.

15 Q. Hu, H. Zhu, X. Li, M. Zhang, Z. Deng, X. Yang and Z. Deng, *PLoS One*, 2012, **7**, e52901.

16 I. Koppert-Anisimova, *Cross-entropy loss in ML*, 2021, https://medium.com/unpackai/cross-entropy-loss-in-ml-d9f22fc11fe0.

17 D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017, https://arxiv.org/abs/1412.6980.

18 T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, *Huggingface's transformers: state-of-the-art natural language processing*, 2020, https://arxiv.org/abs/1910.03771.

19 V. Kotu and B. Deshpande, *Ensemble Modeling*, 2019, https://www.sciencedirect.com/topics/computer-science/ensemble-modeling.