Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks

Ananthan Nambiar*
Department of Bioengineering
Carl R. Woese Inst. for Genomic Biol.
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA
nambiar4@illinois.edu

Maeve Heflin*
Department of Computer Science
Carl R. Woese Inst. for Genomic Biol.
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA

Simon Liu*
Department of Computer Science
Carl R. Woese Inst. for Genomic Biol.
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA

Sergei Maslov
Department of Bioengineering
Department of Physics
Carl R. Woese Inst. for Genomic Biol.
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA

Mark Hopkins†
Department of Computer Science
Reed College
Portland, OR, USA

Anna Ritz†
Department of Biology
Reed College
Portland, OR, USA
aritz@reed.edu

ABSTRACT

The scientific community is rapidly generating protein sequence information, but only a fraction of these proteins can be experimentally characterized. While promising deep learning approaches for protein prediction tasks have emerged, they have computational limitations or are designed to solve a specific task. We present a Transformer neural network that pre-trains task-agnostic sequence representations. This model is fine-tuned to solve two different protein prediction tasks: protein family classification and protein interaction prediction. Our method is comparable to existing state-of-the-art approaches for protein family classification while being much more general than other architectures. Further, our method outperforms all other approaches for protein interaction prediction. These results offer a promising framework for fine-tuning the pre-trained sequence representations for other protein prediction tasks.

CCS CONCEPTS

• **Applied computing** → **Computational proteomics**; • **Computing methodologies** → **Machine learning algorithms**; **Neural networks**; **Natural language processing**.

KEYWORDS

Neural networks, protein family classification, protein-protein interaction prediction

*These authors contributed equally to this research.
†These authors contributed equally to this research.

BCB '20, September 21–24, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7964-9/20/09…$15.00
https://doi.org/10.1145/3388440.3412467

ACM Reference Format:
Ananthan Nambiar, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz. 2020. Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '20), September 21–24, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3388440.3412467

1 INTRODUCTION

The advent of new protein sequencing technologies has accelerated the rate of protein discovery [29]. While protein sequence repositories are growing exponentially, existing methods for experimental characterization are not able to keep up with the present rate of novel sequence discovery [10, 26]. Currently, less than 1% of all amino acid sequences in the UniProtKB database have been experimentally characterized [10]. The explosion of uncharacterized proteins presents opportunities in computational approaches for protein characterization. Harnessing protein sequence data to identify functional characteristics is critical to understanding cellular functions as well as developing potential therapeutic applications [9]. Sequence-based methods to computationally infer protein characteristics have been critical for inferring protein function and other characteristics [32]. Thus, the development of computational methods to infer protein characteristics (which we generally describe as "protein prediction tasks") has become paramount in the field of bioinformatics and computational biology. Here, we adapt a Transformer neural network to establish task-agnostic representations of protein sequences, and use the Transformer network to solve two protein prediction tasks.

1.1 Background: Deep Learning

In applying deep learning to sequence-based protein characterization tasks, we first consider the field of natural language processing (NLP), which aims to analyze human language through computational techniques [24]. Deep learning has recently proven to be a critical tool for NLP, achieving state-of-the-art performance on benchmarks for named entity recognition, sentiment analysis, question answering, and text summarization, among others [6, 37].

Neural networks are functions that map one vector space to another. Thus, in order to use them for NLP tasks, we first need to represent words as real-valued vectors. Often referred to as *word embeddings*, these vector representations are typically "pre-trained" on an auxiliary task for which we have (or can automatically generate) a large amount of training data. The goal of this pre-training is to learn generically useful representations that encode deep semantic and syntactic information [37]. Then, these representations can be used to train systems for NLP tasks for which we have only a moderate amount of training data.

Google recently developed a state-of-the-art contextualized word embedding technique called Bidirectional Encoder Representations from Transformers, or BERT [13], to create "deeply bidirectional" embeddings. BERT's architecture stacks multiple Transformer encoder layers [35], whose architecture we explain in Section 2.2, to train a model that generates token representations by simultaneously incorporating the leftward and rightward context.

1.2 Applying Deep Learning to Protein Prediction Tasks

Because of the unprecedented success in applying deep learning to NLP tasks, one of the recent interest areas in computational biology has been applying NLP-inspired techniques to amino acid sequence characterization. These techniques typically treat amino acids as analogous to individual characters in a language alphabet. Following the pre-training regime established in NLP, tools such as seq2vec and ProtVec have been developed to create amino acid sequence embeddings for protein prediction tasks. These two representations methods are based on the ELMo model and Skip-Gram technique, respectively, and demonstrated state-of-the-art accuracy when applied to bioinformatics tasks [4, 19].

Given the success of BERT in NLP, we pre-train a Transformer network on amino acid sequence representations for protein prediction tasks and compare its performance to current state-of-the-art methods. Instead of using the original BERT training procedure, we use a more recent procedure called A Robustly Optimized BERT Pretraining Approach (RoBERTa) [22]. The main differences lie in the pre-training step, which we highlight later in Section 2.3. We evaluate the performance of our sequence representations on two protein prediction tasks, as described below: (1) protein family classification and (2) binary protein-protein interaction prediction.

1.2.1 Task: Protein Family Classification. Protein families are groups of evolutionarily-related proteins that typically share similar sequence, structural, and functional characteristics. Traditionally, family classification has required the comparison of experimentally identified characteristics. However, methods have also been developed to computationally classify proteins based solely on sequence similarity. This approach enables us to infer functional and structural characteristics of proteins in a high-throughput manner.

Current computational approaches for protein family classification include methods such as BLASTp and profile hidden Markov models (pHMMs) which compare sequences to a large database of pre-annotated sequences. However, inference using these alignment-based methods is computationally inefficient, as they require repeated comparison of sequences to an exponentially growing database of labeled family profiles and are limited by expensive, manually-tuned processing pipelines [11]. With the exponential growth of protein discovery, the development of more scalable approaches is required to overcome traditional bottlenecks [8].

Guided by a deep learning framework, recent models based on convolutional neural network (CNN) and recurrent neural network (RNN) architectures have been successful in achieving state-of-the-art accuracy on the protein family classification task [8, 19]. However, these methods still produce task-specific models that are unable to generalize towards a broader range of protein prediction tasks. One recent model, UDSMProt, used an RNN architecture in a similar pre-training and fine-tuning framework to predict whether a given protein is contained in the same superfamily or fold of a reference protein [33]. Such an approach requires pairwise comparison of sequences against multiple reference proteins, which may not be entirely representative of a protein family. Instead, we present two variants for the protein family classification task: (1) a multi-class classification problem of predicting a family label for a given sequence and (2) a binary classification problem of predicting whether a sequence is a member of a chosen family label.

1.2.2 Task: Protein-Protein Interaction (PPI) Prediction. The next protein prediction task we highlight is protein-protein interaction (PPI) prediction. Defined as physical contacts involving molecular docking between proteins in a specific context, PPIs are fundamental to most cellular processes [12]. However, experimental identification of PPIs has proven to be a complex and time-consuming process, thus creating the need for an efficient and reliable method of computationally predicting PPIs.

We define PPI prediction as a binary classification task to predict whether two proteins will interact given their amino acid sequences. Traditionally, computational identification of PPIs has relied on genomic, structural, or domain information of the interacting proteins [16]. However, such knowledge is not readily available for most proteins. More recent work has leveraged deep learning-based architectures such as stacked autoencoders, recurrent neural networks (RNNs), and recurrent convolutional neural networks (RCNNs) [9, 15, 34]. These models have achieved state-of-the-art accuracy in the binary PPI classification task, as well as the ability to generalize to similar PPI characterization tasks such as interaction type prediction and binding affinity estimation [9, 15]. Despite this success, they still demonstrate an inability to transfer learned knowledge to more general protein prediction tasks.

1.3 Contributions

We propose applying a Transformer neural network, which we call Protein RoBERTa (PRoBERTa) to pre-train task-agnostic vector representations of amino acid sequences. PRoBERTa modifies the RoBERTa procedure by reducing the number of transformer layers, using the LAMB optimizer, and training the model on amino acid sequences tokenized using byte-pair encoding. We then fine-tune these representations towards two protein prediction tasks: *protein family classification* and *binary PPI prediction.* We compare PRoBERTa to current state-of-the-art methods for both tasks.

In the protein family classification task, we apply our model to both the multi-class classification problem and the binary classification problem. We show that the embeddings produced by PRoBERTa can be used to produce models for family classification that contain more information about protein family membership than the pre-trained embeddings, and have comparable performance to current methods that use specialized task-specific architectures.

Additionally, by evaluating PRoBERTa on the binary PPI prediction task, we demonstrate how our trained sequence embeddings can generalize to other protein prediction tasks. In the PPI prediction task, PRoBERTa outperforms other methods in two classification settings. To the best of our knowledge, this is the first reported application of a Transformer network for the protein family classification and binary PPI prediction tasks.

PRoBERTa is also much more computationally efficient than recent work that applies Transformer networks to encode protein sequences to predict protein secondary structure. Using a BERT-based model, Rives et. al (2019) pre-trained their model on 128 NVIDIA V100 GPUs for 4 days [30]. In comparison, we pre-train PRoBERTa on 4 NVIDIA V100 GPUs in 18 hours using (1) a modified architecture, (2) the RoBERTa training procedure [22], and (3) the LAMB optimizer [36]. By using this framework, we can use a smaller pre-training corpus while obtaining state-of-the-art accuracies, increasing the computational efficiency for pre-training by a factor of 170 compared to the most recently published model.

2 METHODS

We treat proteins as a "language" and draw ideas from the state-of-the-art techniques in natural language processing to obtain a vector representation for proteins. For a sequence of amino acids to be treated as a sentence, the alphabet of the language is defined to be the set of symbols

$$\Sigma = \{\text{A, R, N, D, B, C, E, Q, Z, G, H, I, L, K, M, F, P, O, S, U, T, W, Y, V, X}\},$$

where each symbol represents one of 22 amino acids as well as three additional symbols (B, X, and Z). Two of these are reserved for when it is not possible to differentiate between asparagine/aspartic acid (B) and glutamine/glutamic acid (Z). The last symbol (X) is used for unknown amino acids. This convention is based on the official IUPAC amino acid one-letter notation [1].

2.1 Tokenization with Byte Pair Encoding

Before amino acid sequences can be interpreted as a language, we must first define what a word is. This is more challenging for proteins than most natural languages because unlike the space character in languages like English, there is no single character (or amino acid) that is used to divide parts of an amino acid sequence into meaningful chunks. In the past, deep learning models have either used individual amino acids as input [9, 18] or have chosen to group every three amino acids as a "word" [4]. However, there has been recent interest [3] in statistically determining segments of amino acids to be used as inputs for downstream machine learning algorithms using an NLP method called byte pair encoding (BPE) [14]. Byte pair encoding was originally developed as a compression algorithm although it has been adapted more recently as a method for identifying subword units [31].

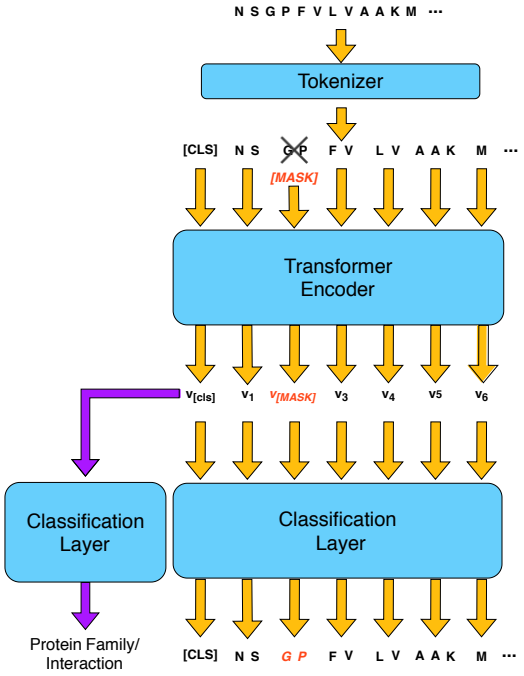

Figure 1: PRoBERTa pre-training and fine-tuning.

In our application, given an amino acid sequence $s = \langle \sigma_1, ..., \sigma_m \rangle$ such that $\sigma_i \in \Sigma$, a *tokenization function* is a mapping τ such that $\tau(s) = \langle t_1, t_2, ..., t_n \rangle$ and each t_i is a nonempty substring of s such that $s = t_1 \cdot ... \cdot t_n$ The BPE algorithm iteratively merges the most frequent pair of tokens to form a new token [14, 21].

2.2 Transformer Network Architecture

PRoBERTa uses the language representation model architecture called Bidirectional Encoder Representations from Transformers (BERT) [13]. Inspired by the BERT architecture, PRoBERTa consists of (1) an embedding layer, followed by (2) $T = 5$ stacked Transformer encoder layers, and (3) a final layer which constructs a task-specific output (Figure 1). By stacking multiple Transformer encoder layers, the aim is to capture complex higher-level information and relationships from the amino acid sequence. In total, our model has approximately 44M trainable parameters.

Model Input. A tokenized amino acid sequence $\langle u_1, u_2, ..., u_n \rangle$ is either truncated or padded to a fixed-length sequence of 512 tokens. Concretely, the model input $t = \langle t_1, t_2, ..., t_{512} \rangle$ is defined:

$$t_i = \begin{cases} u_{i+1} & \text{if } 1 \le (i+1) \le n \\ [\text{CLS}] & \text{if } i = 1 \\ [\text{PAD}] & \text{if } n < i < 512 \\ [\text{EOS}] & \text{if } i = 512 \end{cases}$$

where [CLS], [PAD], and [EOS] are reserved symbols.

Embedding Layer. To prepare an input sequence $t = \langle t_1, t_2, ..., t_n \rangle$ for the Transformer encoder layers, we train an embedding layer which independently converts each token t_i into a vector with dimension $d = 768$. The choice of 768 was made after empirically testing $d \in \{192, 384, 768\}$. Because the model does not contain any convolution or recurrence, we incorporate sequence order

information by adding positional encodings to the input embedding vectors [35].

Transformer Encoder Layer. Each Transformer encoder layer contains two sub-layers – a multi-head self-attention mechanism and a fully-connected feed-forward network – with residual connections around each sub-layer followed by a layer normalization operation [5]. Each sub-layer, and thus the entire encoder layer, takes as input and produces a list of n vectors, each of dimension $d = 768$.

Given an input list of vectors $x = \langle x_1, x_2, ..., x_n \rangle$, each vector x_i first travels through the multi-head self-attention mechanism. This mechanism is composed of $a = 12$ separate randomly initialized attention heads, which are trained to identify and then focus on certain subsets of positions in x based on their computed context relevance to x_i. Using this mechanism, the sub-layer encodes context information from each vector x_j in x, weighted by its relevance to x_i, into an output vector y_i.

The initial input vector x_i is then added to the output vector y_i, after which y_i undergoes a layer-normalization step and passes through a fully-connected feed-forward network which has a single hidden layer of size $h = 3072$ and uses a GeLU activation [20]. The choice of 3072 comes from multiplying 768 by 4 as suggested in the original BERT publication [13]. Each vector y_i passes independently through the same feed-forward network to generate the output vector z_i. The vector y_i is then added to z_i, after which z_i undergoes another layer-normalization step. The output for the entire Transformer layer is the list of vectors $\langle z_1, z_2, ..., z_n \rangle$ [35]

Model Output. Without adding any task-specific heads to the architecture, the model output is a list of $l = 512$ vectors, each with length $d = 768$. The first vector, which corresponds to the special [CLS] token, acts as an aggregate sequence representation which we use for sequence classification tasks. We refer to the entire output as the deep representation of the amino acid sequence.

2.3 Model Pre-training

Following the BERT framework, we train PRoBERTa in two stages: *pre-training* and *fine-tuning* [13]. In the pre-training stage, our objective is to train the model to learn task-agnostic deep representations that capture the high-level structure of amino acid sequences.

Based on the RoBERTa procedure, we pre-train PRoBERTa using only the unsupervised Masked Language Modeling (MLM) task, which adds a Language Modeling head to the network architecture. MLM randomly masks certain tokens and then trains the network to predict their original value [22]. Specific training hyperparameters and optimization are detailed in Section 2.5.

Given a tokenized input sequence $\langle t_1, t_2, ..., t_n \rangle$, we select a random sample of tokens in the sequence to be replaced with a special token [MASK]. Then we use the cross-entropy loss to train the network to predict the masked tokens. Unlike the original BERT procedure, following the RoBERTa procedure, we generate a new masking pattern every time we feed a sequence to the model [22]. The original BERT procedure also included a Next Sentence Prediction (NSP) task. However, given that proteins are not made up of multiple sentences, as we have defined them, NSP is not an appropriate pre-training task. In addition, removing the NSP task has been shown to improve downstream performance in NLP [22].

Pre-training Data. We use UniProtKB/Swiss-Prot (450K unique sequences with a mean tokenized length of 129.6 tokens), a collection of experimentally annotated and reviewed amino acid sequences [10]. Sequences are tokenized with the BPE algorithm described in Section 2.1. In our experiments, the maximum vocabulary size was set to 10,000 because we empirically observed that the mean token length increased very little beyond 10,000 tokens, indicating that most of the longer tokens were detected in the first 10,000 iterations.

2.4 Model Fine-tuning

The pre-trained model can then be specialized for downstream protein prediction tasks. In the fine-tuning stage, we initialize the model with the pre-trained parameters. We then modify the pre-trained architecture by replacing the output layer with a task-specific layer with dimensions tailored to the specific task. Parameters are fine-tuned using labeled data from the prediction tasks.

Here, we fine-tune the pre-trained model for our two specific prediction tasks: family classification and protein-protein interaction prediction. For our selected tasks, we feed the aggregate sequence representation corresponding to the special [CLS] token, as described in Section 2.2, into an output layer, which consists of a single-layer feed-forward neural network and softmax classifier.

2.4.1 Task: Protein Family Classification. For this task, we perform two modes of classification: *binary family classification* and *multi-class family classification.* In binary family classification, we train a separate classifier for each protein family to identify which sequences belongs to a given family. This classifier performs logistic regression on the trained sequence representations from the pre-trained model. We create a balanced training dataset for each classifier consisting of all the positive examples and the same number of negative examples drawn uniformly at random without replacement from outside the family. In multi-class family classification, we train a single classifier that outputs a probability distribution over the set of all protein families.

Fine-tuning Data. For the Family Classification tasks, we use 313,214 unique protein sequences from UniProtKB/Swiss-Prot whose manually curated annotations include protein family information and are not associated with multiple families or hierachical family classifications [10].

2.4.2 Task: PPI Prediction. Given a pair of tokenized amino acid sequences

$$\langle t_1^{(1)}, t_2^{(1)}, ..., t_n^{(1)} \rangle \text{ and } \langle t_1^{(2)}, t_2^{(2)}, ..., t_n^{(2)} \rangle,$$

we pack them together into a single input sequence separated by a special token [SEP], which in the RoBERTa procedure is composed of two [EOS] tokens. The input representation becomes

$$\langle [\text{CLS}], t_1^{(1)}, t_2^{(1)}, ..., t_n^{(1)}, [\text{SEP}], t_1^{(2)}, t_2^{(2)}, ..., t_n^{(2)}, [\text{EOS}] \rangle.$$

We truncate each tokenized amino acid sequence to 254 tokens before concatenation so the maximum combined length of the input sequence after the addition of the special tokens is $l = 512$ tokens.

For this problem, we use the fine-tuning procedure to add a binary classifier layer to the existing pre-training architecture.

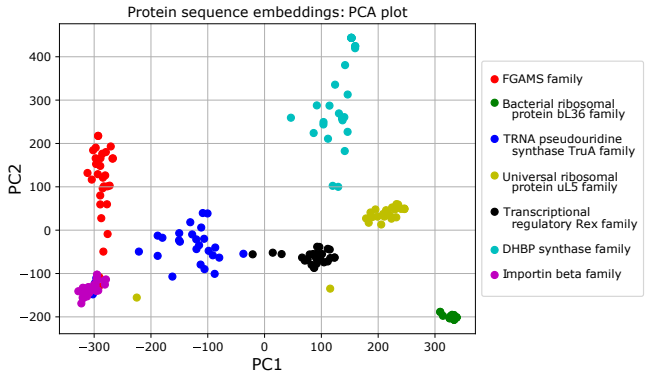

Figure 2: The first two principal components of pre-trained embeddings for 189 amino acid sequences.

Fine-tuning Data. For the PPI prediction task, we use experimentally identified human PPIs from the HIPPIE database that are confidence scored and functionally annotated [2]. Because HIPPIE only reports interacting protein pairs, we generated two sets of putative non-interacting protein pairs to serve as negative examples. In the "conservative" scenario, we generated 275,401 pairs using randomly selected human proteins from UniProt [17] that are not reported to interact in HIPPIE, resulting in a PPI dataset of 536,545 pairs. In the "aggressive" scenario, we generated 275,401 pairs using randomly selected proteins from HIPPIE that are not reported to interact, also resulting in a PPI dataset of 536,545 pairs. This random generation of negative examples is possible because of the assumption that protein-protein interaction networks are sparse, although we note the possibility that these negative examples may include interacting protein pairs not reported in HIPPIE.

2.5 Hyperparameters and Optimization

Our two models use the same hyperparameters and optimization values, with differences between the pre-training and fine-tuning stages described below. We use the fairseq toolkit to train and evaluate our model [27]. We train our model with the LAMB optimizer [36], which is a layerwise adaptive large batch optimization technique developed to increase performance and reduce training time for attention-based models. We use the hyperparameter values from the original LAMB implementation: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon =$ 1e-8, and weight decay rate $\lambda = 0.01$.

We use a minibatch size of 8192 examples. For the pre-training stage, the learning rate is warmed up linearly over the first 3,125 updates to the peak value of 0.0025. For the fine-tuning stage, the learning rate is warmed up over the first 312 updates. Afterwards, it is adjusted using a polynomial decay policy. We selected the learning rate and warmup period using the square root LR scaling heuristic and linear-epoch warmup scheduling because of their success when applied to BERT-based models [36].

Per the original RoBERTa training procedure, we use a dropout of 0.1 on all Transformer layers and attention weights and a GeLU activation function [13]. To avoid overfitting and balance model performance with computational efficiency, we use early stopping with a patience value of 3 (training stops after 3 consecutive epochs with no improvement in either MLM validation loss during pre-training or task-specific validation accuracy during fine-tuning).

3 RESULTS

We first describe the sequence features learned from the pre-trained model. We then show PRoBERTa's performance when the model is fine-tuned for the Protein Family Classification and PPI Prediction tasks. Finally, we perform a robustness analysis by limiting the amount of labeled input data during fine-tuning.

3.1 Protein Embeddings from the Pre-Trained Model

We pre-trained the PRoBERTa model as described in Section 2.3 on 4 NVIDIA V100 GPUs in 18 hours. We first explored whether the pre-trained model captured any biological meaning from the amino acid sequences. We created protein embeddings by concatenating the vectors of each protein's first 128 tokens and plotted the first two principal components of thirty proteins from seven related protein families (Figure 2). We concatenated the vectors because the concatenated vectors appeared to provide better visualization results than the [CLS] token. Figure 2 shows that the pre-trained model is already able to distinguish between these protein families.

To systematically evaluate how well the pre-trained protein embeddings distinguish protein families, we clustered 9,151 protein embeddings from the manually-annotated human proteins in UniProt and compared the clusters to the 3,860 annotated protein families using Normalized Mutual Information (NMI), an entropy based metric that measures the agreement between two sets of labels [28]. However, because this unsupervised learning approach is especially susceptible to the curse of dimensionality, we substituted the model described in Section 2.2 with a similar model but with embedding dimension $d = 192$ instead of $d = 768$ [7] and summed the output vectors. This lower-dimensional embedding is only used for the unsupervised learning task. To cluster, we reduced the pre-trained protein embeddings to twenty dimensions using PCA and applied k-means clustering using Euclidean distance, setting $k = 4000$ to approximate the number of annotated families. The mean NMI of the k-means clusters, averaged over 20 runs, is 0.798213, which is significantly higher than the NMI of randomly assigned clusters (Figure 3, t-test p-value: 8×10^{-180}).

3.2 Protein Family Classification

Given the promise of the protein embeddings, we then evaluated the performance of the PRoBERTa model on the protein family classification task (Section 2.4.1). Clustering the embeddings after fine-tuning on the protein family classification task shows a higher NMI than clustering after pre-training (Figure 3), suggesting that the fine-tuned embeddings capture more protein family information.

For the binary classification task, we trained a separate logistic regression classifier for each protein family with more than 50 proteins and measured the weighted mean accuracy as 0.98. The classifier corresponding to the lowest scoring family, made up of 57 proteins, had an accuracy of 0.77. To train these classifiers, we randomly withheld 30% of the proteins from each family to be used as the test set. We compared PRoBERTa+logistic to three other NLP based embedding methods: ProtVec, which is a protein

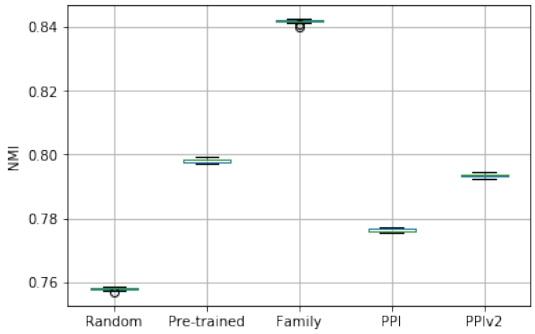

Figure 3: Plot of NMI values comparing the unsupervised clustering on three different versions of PRoBERTa embeddings with the true protein families given by UniProt. PPI: conservative scenario; PPIv2: aggressive scenario.

Table 1: Comparison of binary family classification (left) and multi-class family classification (right).

Method	Accuracy
ProtVec+logistic	0.89
ProtFreqVec+logistic	0.98
ProtDocVec+logistic	0.98
PRoBERTa+logistic	0.98

Method	Accuracy
DeepFam	0.95
Simple CNN	0.72
PRoBERTa	0.92

embedding method inspired by Word2Vec; ProtDocVec, which modifies ProtVec to use Doc2Vec; and ProtFreqVec, which uses the frequency of triplets of amino acids to form embedding vectors [4, 25]. PRoBERTa+logistic performs better than ProtVec+logistic and similarly to ProtDocVec+logistic and ProtFreqVec+logistic (Table 1). This result supports the idea that contextual embeddings do not significantly help binary family classification [25].

In the multi-class family classification task, we used fine-tuning to add an output layer that maps to protein family labels to the PRoBERTa model. This was done using the dataset of 313,214 UniProt proteins with only one associated family. These proteins were split into train/validation/test sets (0.8/0.1/0.1), and our fine-tuned classifier achieved an accuracy of 0.92 on the test set. We then compared this to two other multi-class family classifiers including a simple CNN made up of four convolution layers as a baseline and DeepFam, a CNN method that is the current state-of-the-art method for protein family classification. In particular, DeepFam is made up of a convolution layer with eight different kernel sizes and 250 convolution units for each kernel size [26]. PRoBERTa with a classification layer performed better than the baseline method and had comparable accuracy to DeepFam (Table 1).

3.3 PPI Prediction

We next assessed PRoBERTa on the PPI task using the conservative and aggressive scenarios for sampling non-interacting protein pairs (Section 2.4.2).

3.3.1 Conservative Scenario. We first evaluated how well the fine-tuned model's embeddings capture protein family information compared to the pre-trained embeddings. In the conservative scenario, the fine-tuned model produces embeddings that cluster with a lower NMI with protein families compared to the pre-trained embeddings (Figure 3), indicating that the parameters of the model fine-tuned on predicting interactions are not as tuned to protein family classification. Evaluated on the test set, the fine-tuned PRoBERTa PPI classifier had an accuracy of 0.96 with a precision and recall of 0.95 and 0.97, respectively, (Table 2) and an ROC AUC of 0.99 (Figure 4). In these runs, we only used 20% of the available training and validation data from the train/validation/test (0.8/0.1/0.1) split. This was done because some of the methods we compare against were not able to scale up to using 80% of the data for training and thus would not be able to make a fair comparison to the PRoBERTa model trained with the entire train set. We compare our results to PIPR, which is one of the top PPI prediction neural networks currently available, using a similar number of interactions from our dataset [9]. PIPR uses a residual convolutional neural network (RCNN) architecture to extract both sequential information as well as local features relevant for PPI prediction. We also compare our embeddings to the ProtVec embeddings combined with a feed-forward neural network with three hidden layers (which we call *DeepInteract*) that predicts PPI. Finally, we try a biologically-motivated transfer learning approach by first training a DeepFam network on protein family classification and then using one of the hidden layers as the vector representations of the proteins to be used by DeepInteract. As seen in Figure 4, PRoBERTa with a classification layer outperforms all of these methods by a large margin. Further, when using the complete dataset, the accuracy reaches 0.98 (Table 2 and Figure 6).

Table 2: PPI prediction results using 20% of training data (top) and using 100% of training data (bottom).

	Conservative			Aggressive		
Method	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.
DeepFam+DI*	0.79	0.79	0.66	0.75	0.76	0.73
PIPR	0.81	0.75	0.77	0.77	0.77	**0.77**
ProtVec+DI*	0.80	0.78	0.70	0.73	0.72	0.76
PRoBERTa	**0.96**	**0.95**	**0.97**	**0.79**	**0.82**	0.73
PRoBERTa (100% training)	**0.98**	**0.98**	**0.99**	**0.84**	**0.83**	**0.84**

*DI: Deep Interact

3.3.2 Aggressive Scenario. Similar to the conservative scenario, the model fine-tuned on the aggressive PPI dataset produces embeddings with a lower NMI with protein families than the pre-trained embeddings (Figure 3). For this scenario, PRoBERTa had an accuracy of 0.79 with a precision and recall of 0.82 and 0.73, respectively, (Table 2) and a ROC AUC of 0.87 (Figure 5). Similar to above, we only use 20% of the available training and validation data. As seen in Figure 5, PRoBERTa with a classification layer still performs better than PIPR, which performs better than the other two methods. Moreover, when the full training dataset is used, the accuracy of our model improves to 0.84 (Table 2 and Figure 6). However, PRoBERTa does show lower performance compared to PIPR and over-fits (train accuracy: 0.98, test accuracy: 0.77) when trained on smaller datasets, such as the yeast interactions dataset (11,164 interactions) on which PIPR was originally tested [9].

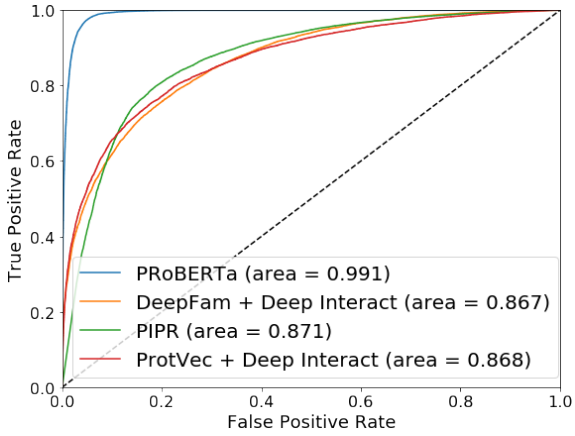

Figure 4: Receiver operating characteristic (ROC) curves for PPI prediction in the conservative scenario.

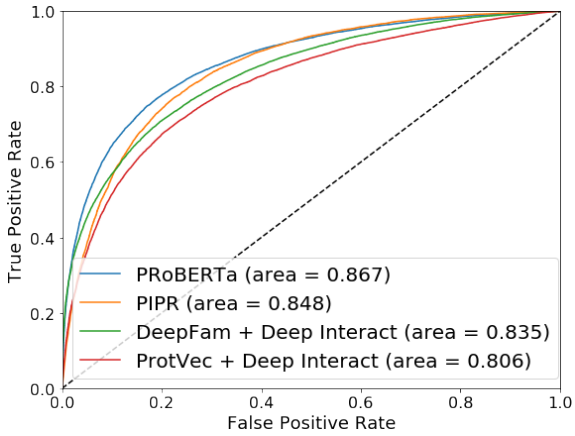

Figure 5: Receiver operating characteristic (ROC) curves for PPI prediction in the aggressive scenario.

3.4 PRoBERTa Scalability and Robustness

We also investigated the robustness of the models by varying the amount of training data for both fine-tuning tasks. For the PPI prediction task in Section 3.3, we used 20% of the training data in order to compare to existing methods; here, we can use all of the training data, improving the performance in both the conservative and aggressive scenarios (Table 2).

To assess the robustness of PRoBERTa on the protein prediction tasks, we used fractions of the 0.8 and 0.1 split that made up the fine-tuning train and validation set, respectively, for task training. For example, if 90% of the train and validation set was used, this meant that $90\% \times 0.9$ or 81% of the entire dataset was seen during training. Figure 6 shows the change in accuracy with different fractions of the train set used. This shows that all three models were somewhat robust to different amounts of training data. The PPI models appear to be more robust (they have smaller slopes) than the Protein Family model. However, it should be noted that the complete dataset for the Protein Family model contained 313,214 proteins, while the PPI dataset had 536,545 interactions in both scenarios. The difference

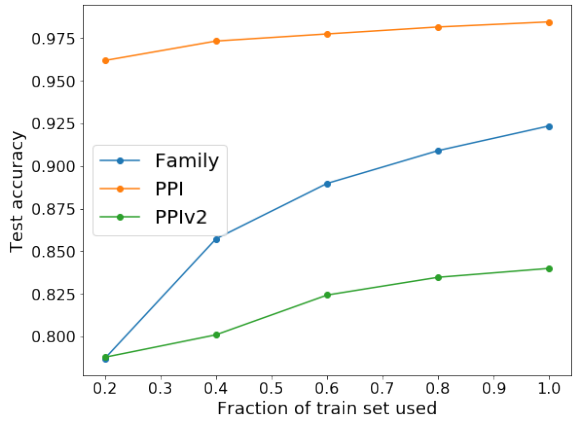

Figure 6: Varying amount of training data for fine-tuning. PPI: conservative scenario; PPIv2: aggressive scenario.

in robustness could be due to the absolute difference in the number of training data points.

4 DISCUSSION

In this paper, we propose a Transformer based neural network architecture, called PRoBERTa, for protein characterization tasks. This neural network is based on the BERT architecture and the RoBERTa training procedure with a reduced number of Transformer layers and using the LAMB optimizer. We found that the pre-trained embeddings contain general yet biologically relevant information regarding the proteins and fine-tuning pushes the embeddings to have more specific information at the cost of generality. While there are notable differences between protein sequences and natural language corpuses [23, 33], leveraging the architecture from BERT tuning can capture this biologically relevant information. Altering the architecture to include prior knowledge unique to biological sequences could further improve the embedding space.

We found that using the embeddings for Protein Family Classification produced results that were comparable to the current best methods. In particular, we performed two different forms of classification; a binary classification that classified a protein as "in the family" or "not in the family" and a multi-class family classification. The multi-class family classification was based on the simplification that there is only one class per protein. Proteins that belong to more than one family were excluded from this classifier but not the binary classifiers.

Furthermore, we used embeddings from PRoBERTa for a fundamentally different problem, PPI prediction, using two different datasets generated from the HIPPIE database and found that with sufficient data, it substantially outperforms the current state-of-the-art method in the conservative scenario and still performs better than the other methods in the aggressive scenario. When evaluated on the aggressive dataset, the model trained on the conservative dataset scores an overall accuracy of 0.59, with a precision and recall of 0.54 and 0.94, respectively. This suggests that the model in the conservative scenario performs something closer to a protein classification task to identify which proteins are present in HIPPIE and are thus more likely to correspond to positive examples.

The efficiency of PRoBERTa over existing methods (a speedup in pre-training time by a factor of 170 compared to the similar BERT-based model by Rives et. al (2019) [30]) provides unprecedented opportunities for using the growing amount of sequence data in protein prediction tasks. Further, PRoBERTa's success in these two different protein prediction tasks alludes to the generality of the embeddings and their potential to be used in other tasks such as predicting protein binding affinity, protein interaction types and identifying proteins associated with particular diseases. In light of the COVID-19 pandemic, we are currently working on adapting PRoBERTa for vaccine design.

5 MODEL SCRIPTS

Scripts for pre-training, fine-tuning, and evaluating models, as well as links to datasets and trained weights can be found at: https://github.com/annambiar/PRoBERTa

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation (awards #1750981 and #1725729). This work has also been partially supported by the Google Cloud Platform research credits program (to AR, MH, and AN). AN would like to thank Mark Bedau, Norman Packard and the Reed College Artificial Life Lab for insightful discussions and Desiree Odgers for inspiring the idea of taking a linguistic approach to a biological problem.

REFERENCES

[1] 1984. Nomenclature and Symbolism for Amino Acids and Peptides. *European Journal of Biochemistry* 138, 1 (1984), 9–37. https://doi.org/10.1111/j.1432-1033.1984.tb07877.x

[2] Gregorio Alanis-Lobato, Miguel A. Andrade-Navarro, and Martin H. Schaefer. 2016. HIPPIE v2.0: enhancing meaningfulness and reliability of protein-protein interaction networks. *Nucleic Acids Research* 45, D1 (10 2016), D408–D414.

[3] Ehsaneddin Asgari, Alice C. McHardy, and Mohammad R. K. Mofrad. 2019. Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). *Scientific Reports* 9, 1 (2019), 3577.

[4] Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2015. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE* 10, 11 (11 2015), 1–15.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] Mark A. Bedau, Nicholas Gigliotti, Tobias Janssen, Alec Kosik, Ananthan Nambiar, and Norman Packard. 2019. Open-Ended Technological Innovation. *Artificial Life* 25, 1 (2019), 33–49. https://doi.org/10.1162/artl_a_00279 PMID: 30933632.

[7] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When Is "'Nearest Neighbor'" Meaningful?. In *Proceedings of the 7th International Conference on Database Theory (ICDT '99)*. Springer-Verlag, Berlin, Heidelberg, 217–235.

[8] Maxwell L. Bileschi, David Belanger, Drew Bryant, Theo Sanderson, Brandon Carter, D. Sculley, Mark A. DePristo, and Lucy J. Colwell. 2019. Using Deep Learning to Annotate the Protein Universe. *bioRxiv* (2019).

[9] Muhao Chen, Chelsea J T Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. 2019. Multifaceted protein-protein interaction prediction based on Siamese residual RCNN. *Bioinformatics* 35, 14 (07 2019), i305–i314.

[10] The UniProt Consortium. 2018. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research* 47, D1 (11 2018), D506–D515.

[11] Sayoni Das and Christine A. Orengo. 2016. Protein function annotation using protein domain family resources. *Methods* 93 (2016), 24 – 34.

[12] Javier De Las Rivas and Celia Fontanillo. 2010. Protein-Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. *PLOS Computational Biology* 6, 6 (06 2010), 1–8.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[14] Philip Gage. 1994. A New Algorithm for Data Compression. *C Users J.* 12, 2 (Feb. 1994), 23–38.

[15] Yi Guo and Xiang Chen. 2019. A deep learning framework for improving protein interaction prediction using sequence properties. *bioRxiv* (2019).

[16] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. 2008. Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. *Nucleic Acids Research* 36, 9 (04 2008), 3025–3030.

[17] Tobias Hamp and Burkhard Rost. 2015. Evolutionary profiles improve protein-protein interaction prediction from sequence. *Bioinformatics* 31, 12 (02 2015), 1945–1950.

[18] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. 2018. Predicting protein-protein interactions through sequence-based deep learning. *Bioinformatics* 34, 17 (09 2018), i802–i810.

[19] Michael Heinzinger, Ahmed Elnaggar, Yu Wang, Christian Dallago, Dmitrii Nachaev, Florian Matthes, and Burkhard Rost. 2019. Modeling the Language of Life – Deep Learning Protein Sequences. *bioRxiv* (2019).

[20] Dan Hendrycks and Kevin Gimpel. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *arXiv preprint arXiv:1606.08415* (2016).

[21] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. , 66–71 pages. https://doi.org/10.18653/v1/D18-2012

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://openreview.net/forum?id=SyxS0T4tvS

[23] Yunan Luo, Lam Vo, Hantian Ding, Yufeng Su, Yang Liu, Wesley Wei Qian, Huimin Zhao, and Jian Peng. 2020. Evolutionary context-integrated deep sequence modeling for protein engineering. *bioRxiv* (2020). https://doi.org/10.1101/2020.01.16.908509

[24] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

[25] Ananthan Nambiar, Mark Hopkins, and Anna Ritz. 2019. Computing the Language of Life: NLP Approaches to Feature Extraction for Protein Classification. In *ISMB/ECCB 2019: Poster Session*. https://doi.org/10.7490/f1000research.1118014.1

[26] Minsik Oh, Seokjun Seo, Sun Kim, and Youngjune Park. 2018. DeepFam: deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics* 34, 13 (06 2018), i254–i262.

[27] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[29] Laura Restrepo-Pérez, Chirlmin Joo, and Cees Dekker. 2018. Paving the way to single-molecule protein sequencing. *Nature nanotechnology* 13, 9 (2018), 786–796.

[30] Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2019. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv* (2019).

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725.

[32] Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

[33] Nils Strodthoff, Patrick Wagner, Markus Wenzel, and Wojciech Samek. 2020. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics* (01 2020). btaa003.

[34] Tanlin Sun, Bo Zhou, Luhua Lai, and Jianfeng Pei. 2017. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC Bioinformatics* 18, 1 (2017), 277.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[36] Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Reducing BERT Pre-Training Time from 3 Days to 76 Minutes. *CoRR* abs/1904.00962 (2019).

[37] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent Trends in Deep Learning Based Natural Language Processing. *CoRR* abs/1708.02709 (2017).