An Analysis of Protein Language Model Embeddings for Fold Prediction

Amelia Villegas-Morcillo[*], Angel M. Gomez, Victoria Sanchez

Department of Signal Theory, Telematics and Communications, University of Granada, Spain

Abstract

The identification of the protein fold class is a challenging problem in structural biology. Recent computational methods for fold prediction leverage deep learning techniques to extract protein fold-representative embeddings mainly using evolutionary information in the form of multiple sequence alignment (MSA) as input source. In contrast, protein language models (LM) have reshaped the field thanks to their ability to learn efficient protein representations (protein-LM embeddings) from purely sequential information in a self-supervised manner. In this paper, we analyze a framework for protein fold prediction using pre-trained protein-LM embeddings as input to several fine-tuning neural network models which are supervisedly trained with fold labels. In particular, we compare the performance of six protein-LM embeddings: the LSTM-based UniRep and SeqVec, and the transformer-based ESM-1b, ESM-MSA, ProtBERT, and ProtT5; as well as three neural networks: Multi-Layer Perceptron (MLP), ResCNN-BGRU (RBG), and Light-Attention (LAT). We separately evaluated the pairwise fold recognition (PFR) and direct fold classification (DFC) tasks on well-known benchmark datasets. The results indicate that the combination of transformer-based embeddings, particularly those obtained at amino acid-level, with the RBG and LAT fine-tuning models performs remarkably well in both tasks. To further increase prediction accuracy, we propose several ensemble strategies for PFR and DFC, which provide a significant performance boost over the current state-of-the-art results. All this suggests that moving from traditional protein representations to protein-LM embeddings is a very promising approach to protein fold-related tasks.

Key words: Protein Fold Prediction, Protein Language Models, Fine-Tuning Neural Networks, Embedding Learning

1 Introduction

Despite recent breakthroughs in predicting protein three-dimensional structures with high accuracy (AlphaFold [1, 2] and RoseTTAFold [3]), there is still a special interest in identifying the fold type of a protein. This allows for a better understanding of the functionality of the newly solved structures (e.g. AlphaFold DB [4]), and it is often accomplished by classifying them according to structural and sequential similarities with respect to known proteins. In this regard, structural classification databases such as SCOP [5, 6] and CATH [7, 8] already provide a hierarchical grouping of protein domains from the protein data bank (PDB) [9, 10] into different categories. In SCOP these are named structural class, fold, superfamily and family, with increasing amino acid sequence similarity at each level. Among them, the focus is on obtaining accurate predictions at the fold level, where protein domains have a similar arrangement of structural elements but substantially differ in the amino acid sequence, a problem widely known as protein fold recognition [11–15].

During the past few decades, many computational methods have been proposed to predict the fold class of a protein domain. These can be divided according to the task they aim to solve. The first one is *pairwise fold recognition* (PFR), in which the fold class of the query protein is inferred by comparing with templates with known structure [16–18]. PFR approaches mainly include methods based on homology modeling (sequence alignments [19], profile alignments [20], and Markov random fields [21]); threading [22–28]; machine learning for binary classification [29–31]; multi-view learning [32–34]; and learning to rank [35–37]. Another set of methods use deep learning to learn *fold-representative embeddings*, which are then used to measure the structural similarity between two protein domains. DeepFR [38] introduced this methodology, which has been followed by more recent approaches [39–45] using either predicted contact maps, or evolutionary information as input representation of the protein. The second task is *direct fold classification* (DFC), in which the protein sequences are directly mapped into a pre-defined group of fold classes [46]. Most of the proposed methods [47–56] had used evolutionary information and machine learning to classify only a small portion of all possible SCOP folds (i.e.

[*]Corresponding author. ameliavm@ugr.es

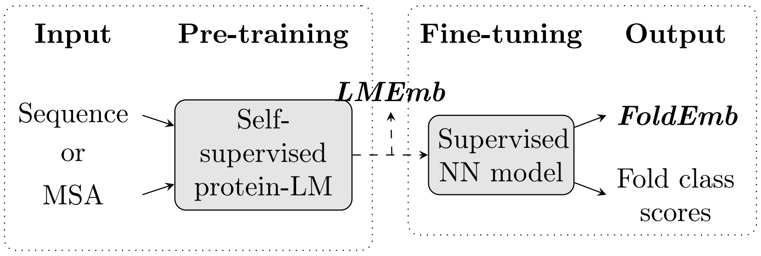

Figure 1: Overview of our approach for protein fold prediction. First, we extract a protein embedding representation from the amino acid sequence or multiple sequence alignment (MSA) using protein language models (protein-LMs) which have been pre-trained in a self-supervised manner (i.e. using the input sequential information itself). As a result, we obtain a protein-LM embedding (*LMEmb*) of size $L \times F$, where L is the length of the protein sequence and F is the size of the amino acid-level embedding. Then, we fine-tune this embedding through a neural network (NN) model that is trained, in a supervised manner, to map the input protein into K fold classes. The outputs of this model are, on the one hand, a fold-representative embedding of the protein (*FoldEmb* with fixed-size 512), used to perform the pairwise fold recognition (PFR) task; and, on the other hand, the scores for each fold class, used in the direct fold classification (DFC) task.

the most populated ones). In contrast to them, DeepSF [57] was the first method to perform fold classification into one of the more than thousand existing folds in SCOP through deep learning.

Traditionally, the state-of-the-art methods for different protein-related tasks have used evolutionary information in the form of multiple sequence alignment (MSA) as input source. Following a more modern approach, recent methods use protein representations that are extracted from pre-trained protein language models (protein-LMs). These models have been taken from the field of NLP (natural language processing) [58], by treating the protein sequences as sentences, and the amino acids as word equivalents. More specifically, the models learn meaningful representations of the proteins (*protein-LM embeddings*) in a self-supervised manner [59] by using the vast amount of unlabeled sequences contained in protein databases such as Swiss-Prot [60], Pfam [61], and UniRef [62] (all based on UniProt [63]); and metagenomic databases such as the big fantastic database (BFD) [64, 65]. This way, ProtVec [66], based on word2vec [67, 68], was the first method proposed to extract protein representations from their sequences. Unlike word2vec, more sophisticated protein-LMs take into account both the context and order of amino acids in the sequence through LSTM (long short-term memory) recurrent units [69]. These include methods such as UDSMProt [70] and UniRep [71], using weight-dropped LSTMs and multiplicative LSTMs, respectively; as well as two methods based on the ELMo model [72]: SeqVec [73] and the language model part from [74]. The next generation of protein-LMs come from the use of transformer architectures based on self-attention mechanisms [75]. Examples are TAPE-Transformer [76], ESM-1b [77], ESM-MSA [78] and ProtBERT [79], which have been inspired in the BERT model [80]. In addition to BERT, the ProtTrans project [79] explored the use of other five transformer architectures [81–85] for protein representation learning.

These pre-trained models allow for transfer learning to different protein-related downstream tasks in which the amount of labeled sequences in the databases is significantly smaller. By using simple supervised models, protein-LM embeddings have proven to be successful in predicting protein secondary structure, subcelullar localization, and remote homologs [73, 76, 79, 86], as well as protein function [87–89] and sequence variation [90, 91]. It has been also found that the attention layers in transformer models are able to learn protein contact map information directly from self-supervised training on sequences [77, 78, 92].

Our proposal here (summarized in Figure 1) is to leverage several pre-trained protein-LM embeddings for fold prediction. We hypothesize that self-supervised training of the protein-LMs might capture fold information from millions of protein sequences, and therefore the learned representations could be useful for comparison of structurally similar proteins (PFR task), as well as classification into fold classes as defined by SCOP (DFC task). To test this, we followed the same idea of the DeepSF, DeepFR, and subsequent deep learning models for fold prediction. These neural network models allow for the fine-tuning of protein-LM embeddings (here *LMEmb*) to learn new fold-representative embedding vectors (here *FoldEmb*), while performing fold classification. In both tasks, PFR and DFC, we compared the performance of different neural network architectures working on either amino acid-level or protein-level embeddings. Comparison with previous methods also allowed us to analyze the impact of changing traditional protein evolutionary features by protein-LM embeddings as input representations. All in all, we found that transformer-based protein-LM embeddings are particularly useful for protein fold prediction, outperforming the state-of-the-art results for both fold recognition and fold classification.

Table 1: Characteristics of the protein language model (protein-LM) embeddings used in this analysis.

Embeddings	Size (F)	Language Model	#Layers	#Parameters	Training Database
UniRep [71]	1900	mLSTM	1	18M	UniRef50 (24M seqs)
SeqVec [73]	1024	ELMo (BLSTM)	2	93M	UniRef50 (33M seqs)
ESM-1b [77]	1280	BERT (Transformer)	33	650M	UniRef50 (27M seqs)
ESM-MSA [78]	768	BERT (Transformer)	12	100M	UniRef50 (26M MSAs)
ProtBERT [79]	1024	BERT (Transformer)	30	420M	BFD (2B seqs)
ProtT5 [79]	1024	T5 (Transformer)	24	3B	BFD (2B seqs) + UniRef50 (45M seqs)

2 Materials and Methods

2.1 Input Protein Information

Our framework for fold prediction (Figure 1) takes as input sequential information of the protein, either the amino acid sequence or a multiple sequence alignment (MSA). To build the MSAs for our protein domains, we followed the pipeline specified in [78]. That is, we first generated the MSA by running HHblits [93] against the `uniclust30_2017_10` database [94], with number of iterations equal to 3. The resulting MSA was then subsampled by filtering the number of sequences down to 256 with hhfilter [93]. If more than 256 sequences were returned, we applied the diversity maximizing strategy from [78] to select those sequences with highest average hamming distance.

2.2 Pre-Trained Protein-LM Embeddings

As protein representations, we used self-supervised embeddings from pre-trained protein language models (protein-LMs). We analyzed the performance of LSTM-based protein-LMs such as UniRep [71] and SeqVec [73], as well as several transformer-based models such as ESM-1b [77], ESM-MSA-1b (here ESM-MSA) [78], ProtBERT-BFD and ProtT5-XL-U50 (here ProtBERT and ProtT5) [79]. We denote these self-supervised embeddings as *LMEmb*, in order to differentiate them from our fine-tuned embeddings, *FoldEmb*, which are supervisedly trained using fold labels. The total size of an *LMEmb* embedding for a protein of length L is $L \times F$, where F is the size of the individual embedding provided by the protein-LM for each amino acid. By averaging the embedding matrix over the length dimension, we can obtain an alternative fixed-size representation for the protein domain (size F). We will refer to this representation as protein-level embeddings or *LMEmb-Prot* (size F) in contrast to the amino acid-level embeddings defined above and denoted as *LMEmb-AA* (size $L \times F$). Table 1 summarizes the training details for each protein-LM embedding included in the analysis.

LSTM-Based Models. UniRep [71] and SeqVec [73] are two protein-LMs trained using recurrent layers with long short-term memory (LSTM) [69] units. Both models were trained in an auto-regressive manner, trying to predict the next amino acid given all previous amino acids in a protein sequence. The *UniRep* model consists of one layer of multiplicative LSTM (mLSTM) [95] with 1900 hidden units, trained on 24 million protein sequences from the UniRef50 database. We used the TAPE [76] implementation[1] and the pre-trained model from UniRep to extract $L \times 1900$ dimensional embeddings for our protein domains. On the other hand, *SeqVec* is based on the ELMo model [72] from NLP. The SeqVec model was trained on 33 million protein sequences from UniRef50. Its architecture is formed by one CharCNN layer to embed the input characters (amino acids), followed by two layers of bidirectional LSTMs (BLSTM) [96] with shared parameters for the forward and backward passes. We obtained $L \times 1024$ dimensional SeqVec embeddings by concatenating both directions of the LSTMs and then adding the outputs of the three layers. To do so, we used the official code[2] and the pre-trained model of SeqVec.

Transformer-Based Models. We consider two sets of transformer-based protein-LMs—evolutionary scale modeling (ESM) [77, 78] and ProtTrans [79]. The ESM models are based on the BERT transformer architecture [80]. Unlike auto-regressive LSTM-based protein-LMs, ESM models were trained to predict masked amino acids using all preceding and following amino acids in the sequence. This training objective is referred to as masked language modeling (MLM). In our analysis, we consider the pre-trained *ESM-1b* model [77], which has 33 transformer layers and a total of 650M parameters, and was trained on 27 million protein sequences from UniRef50. Instead of individual protein sequences, the *ESM-MSA* model [78] was trained on multiple sequence alignments (MSAs) constructed from sequences in UniRef50 (26 million of MSAs). This model uses the axial attention mechanism from [97] and has fewer transformer layers (12) and parameters (100M) than ESM-1b.

[1] `https://github.com/songlab-cal/tape`
[2] `https://github.com/mheinzinger/SeqVec`

To obtain an embedding for each protein domain in our datasets we used the official code[3] and the pre-trained ESM-1b and ESM-MSA models. For ESM-1b, we extracted amino acid-level embeddings from the last transformer layer, resulting in $L \times 1280$ vectors. In contrast, the ESM-MSA model provides embeddings for each sequence in the MSA, so the output of the last transformer layer is of size $256 \times L \times 768$. We averaged over all sequences in the MSA to obtain final embeddings of size $L \times 768$.

In contrast to the ESM models, the ProtTrans project [79] scaled up the transformer-based protein-LMs to leverage metagenomic databases with billions of protein sequences, leading to architectures with several billions of parameters. In this work, we consider models based on two auto-encoder transformers, BERT and T5 [83], denoted here as ProtBERT and ProtT5 respectively. The *ProtBERT* model has 30 layers and a total of 420M parameters, and has been trained on 2 billion protein sequences from the BFD database. Unlike ProtBERT and the previous ESM models, which only train the encoder component, *ProtT5* includes both the encoder and decoder, with 24 layers and a total of 3B parameters. It was trained on the BFD database and later refined using 45 million sequences from UniRef50. To extract $L \times 1024$ dimensional embeddings for our protein domains, we used the official ProtTrans implementation[4] and the pre-trained models for ProtBERT and ProtT5. Following the recommendations in [79], we only used the encoder part of ProtT5 to embed our protein domains.

2.3 Neural Network Models for Protein Embedding Fine-Tuning

In this subsection we describe the neural architectures and the corresponding supervised training procedures for fine-tuning the protein-LM embeddings *LMEmb* into the fold representative embeddings *FoldEmb* (see Figure 1).

2.3.1 Neural Architectures

Our neural network models include a supervised embedding extractor followed by a linear classifier. Specifically, the embedding extractors accept as input either the amino acid-level protein-LM embeddings *LMEmb-AA* or the averaged protein-level embeddings *LMEmb-Prot*. We used three different neural architectures to extract supervised embeddings, *FoldEmb*, of fixed-size (512): a Multi-Layer Perceptron (MLP), our previously proposed Residual-Convolutional network and Bidirectional Gated Recurrent Unit (RBG) [45], and the Light-Attention (LAT) architecture from [86]. These architectures are illustrated in Figure 2. For the last step classifier we use a single fully-connected (FC) layer which projects the *FoldEmb* embeddings into K output elements (i.e. logits) corresponding to the fold classes. The supervised embedding extractor and classifier are trained together in an end-to-end fashion.

Multi-Layer Perceptron (MLP). As a reference model, we processed the *LMEmb-Prot* embeddings through a simple MLP. The architecture consists of two FC layers with 1024 and 512 neurons each (see Figure 2a) with ReLU activation. After each layer, we also apply batch-normalization [98] and dropout [99] with drop probability $p = 0.5$ to prevent overfitting during training.

Residual-Convolutional Network and Bidirectional Gated Recurrent Unit (RBG). To process *LMEmb-AA* embeddings, we consider our ResCNN-BGRU (RBG) model architecture (Figure 2b) which obtained state-of-the-art performance on protein fold recognition [45]. This architecture consists of three distinct parts, which we briefly describe here (further details can be found in [45]). First, the residual-convolutional part consists of two identical residual blocks with skip connections. Each block contains two 1D-convolutions with 512 and F filters of length 5, where F is the dimensionality of the input embedding (see Table 1). The 1D-convolutions are followed by ReLU activation, batch-normalization and dropout ($p = 0.2$). Second, a bidirectional recurrent layer is applied on top of the $L \times F$ outputs of the residual-convolutional part. This consists of a bi-directional gated recurrent unit (GRU) [100] layer with 1024 state units for each direction. To obtain a fixed-size vector, we concatenate the last states from both forward ($\overrightarrow{\mathbf{h}_L}$) and backward ($\overleftarrow{\mathbf{h}_L}$) GRU layers into a vector of 2048 elements. Finally, we project this vector into a 512-dimensional embedding (*FoldEmb*) using an FC layer of size 512, followed by hyperbolic tangent (tanh) activation and dropout ($p = 0.2$).

Light-Attention (LAT). We also include in our analysis the Light-Attention (LAT) model from [86], which has been recently proposed for predicting protein subcellular location, using *LMEmb-AA* embeddings as inputs. The architecture is shown in Figure 2c. It applies two parallel 1D-convolutions with F filters of length 9, to produce the attention coefficients and value features separately. The attention weights are obtained from the coefficients by applying the softmax operation over the length dimension. The resulting weights are used to compute a weighted sum of the values, producing an F-dimensional vector independent of the protein length. This vector is then concatenated with the max-pooled values (across the length dimension) to produce a vector

[3]https://github.com/facebookresearch/esm
[4]https://github.com/agemagician/ProtTrans

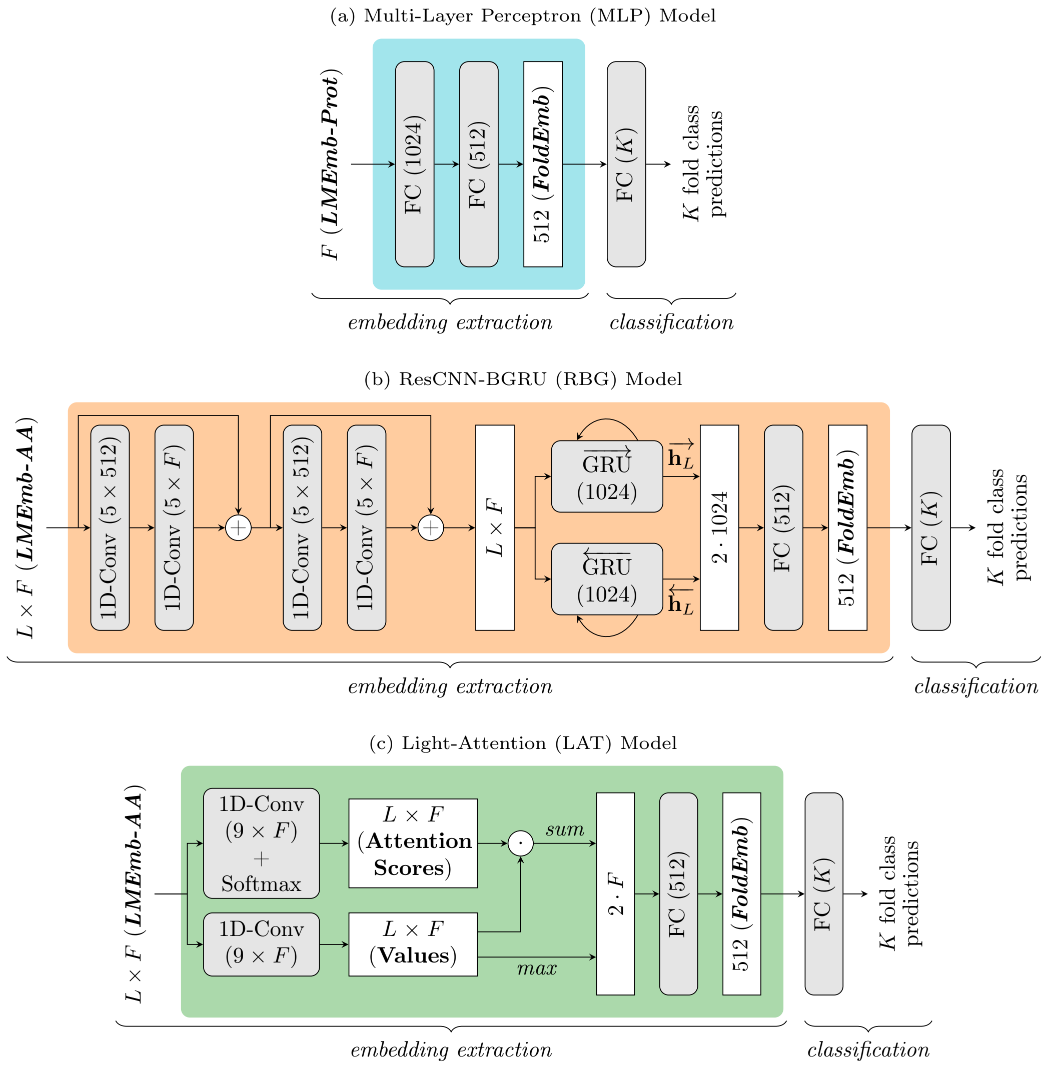

Figure 2: Neural network models used to fine-tune the protein-LM embeddings (*LMEmb*) to fold-representative embeddings (*FoldEmb*), as well as to perform direct fold classification. Three neural architectures are used as embedding extractors (identified with three distinct background colors). **(a)** The Multi-Layer Perceptron (MLP) model processes the protein-level embeddings (*LMEmb-Prot*) through two fully-connected (FC) layers. **(b)** The ResCNN-BGRU (RBG) model [45] processes the amino acid-level embeddings (*LMEmb-AA*) through two residual-convolutional blocks, a bidirectional gated recurrent unit (GRU) layer, and an FC layer. **(c)** The Light-Attention (LAT) model, adapted from [86], also processes *LMEmb-AA* through an attention mechanism followed by an FC layer.

of size $2F$. To compute the 512-dimensional *FoldEmb* embeddings, we adapted the MLP part of the original LAT architecture by including an FC layer similar to that described above for the RBG model.

2.3.2 Model Optimization

The fine-tuning models were trained to minimize the *large margin cosine loss* (LMCL) [101] between the predicted and true fold classes for each protein domain in the training dataset. The LMCL is an L_2-normalized and margin discriminative version of the softmax cross-entropy loss. The L_2 normalization enforces the *FoldEmb* vectors to be distributed on the surface of a hypersphere. It is formally defined as:

$$L_{lmc} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{k\neq y_i} e^{s\cos(\theta_{k,i})}}, \tag{1}$$

where N is the number of training samples in the mini-batch and K is the number of fold classes. The cosine is computed as $\cos(\theta_{k,i}) = \hat{\mathbf{w}}_k^T \hat{\mathbf{x}}_i$, where $\hat{\mathbf{w}}_k$ and $\hat{\mathbf{x}}_i$ are L_2-normalized versions of the k-th class weight vector $\mathbf{w}_k$ (from the last classification layer, bias is set to zero for simplicity) and the i-th embedding vector $\mathbf{x}_i$ (here the 512-dimensional *FoldEmb*). As can be noticed from Eq. 1, this loss introduces two hyperparameters, the scale and margin $(s, m \geq 0)$. The scaling hyperparameter s controls the radius of the hypersphere on which the embeddings are distributed, while the margin m controls the decision boundaries between fold classes, and so the capacity of learning more discriminative embeddings. Following previous results [45] and light hyperparameter tuning using cross-validation, we use the same scale for all our networks ($s = 30$) and a margin with value $m = 0.2$ for the MLP model and $m = 0.6$ for the models working on *LMEmb-AA* embeddings (that is, RBG and LAT).

For training, we use mini-batches with 64 protein domains each, and the Adam optimizer [102] with an initial learning rate equal to 10^{-3}. To prevent overfitting, along with the batch-normalization and dropout techniques specified before, we apply L_2 penalty with a weight decay of $5 \cdot 10^{-4}$. All models were trained for 80 epochs and we decreased the learning rate by a factor of 10 at epoch 40, as it proved to improve the performance in previous works [44, 45]. We implemented our models using PyTorch [103] and executed them on a single GPU (NVIDIA Tesla V100) with 32GB of memory.

2.4 Evaluation Tasks

This subsection details the scoring procedures, ensembling strategies, and performance metrics used to evaluate the two protein fold-related tasks: pairwise fold recognition (PFR) and direct fold classification (DFC). As a reference, an evaluation where the protein-LM embeddings are directly used without fine-tuning (i.e. without additional supervised training) was also considered for both tasks.

2.4.1 Pairwise Fold Recognition (PFR)

PFR Task. The pairwise fold recognition task involves evaluating the structural similarity of two protein domains. To do so, we used the 512-dimensional supervised embeddings (*FoldEmb*) extracted from our neural network models (Figure 2). These embeddings were used to compute a fold similarity score for each of two domains within the test set, indicating whether they belong to the same fold class or not. Following previous works, we used the cosine similarity as the similarity metric [38, 44, 45].

Ensemble Strategies. Since ensemble methods have been found to be promising for PFR in previous works [38, 43–45], we also leveraged ensembling techniques here to obtain a better fold similarity score from our best performing *FoldEmb* embeddings. The first type of ensembling strategy, which we refer to as *average ensemble*, involves directly averaging the cosine similarity scores, provided by the chosen models, for each pair of protein domains in the test set. The second ensembling strategy consists of training a random forest (RF) model using our cosine similarity scores along with the 84 pairwise similarity measures from [30, 31]. We refer to this strategy as *random forest ensemble*. The RF model was trained to determine whether the protein domains in each pair share the same fold class using the vector of pairwise scores as input, and a total of 500 decision trees. Note that this strategy involves training an RF model, so we evaluated using a 10-stage cross-testing setting over the test set as in [38, 44, 45].

Ranking and Evaluation. To evaluate an individual protein domain (query), we ranked the rest of domains in the test set (templates) by fold similarity score, and then assigned the fold class of the most similar one to the query. As originally proposed in [13] and following subsequent works [38, 44, 45], this evaluation was performed at three levels with increasing difficulty, according to the SCOP hierarchy—*family, superfamily* and *fold* [5]. At each level, a positive pair contains two protein domains sharing the same class in the selected level,

Table 2: Number of protein domains and fold classes evaluated in each test set.

Task	Test Set	Full Set		Family Level		Superfamily Level		Fold Level	
		#Domains	#Folds	#Domains	#Folds	#Domains	#Folds	#Domains	#Folds
PFR	LINDAHL	976	330	555	121	434	79	321	38
DFC	LINDAHL_1.75	871	244	591	177	210	107	70	37
	SCOP_2.06	2533	550	742	316	1754	418	37	15

The LINDAHL test set is evaluated on the pairwise fold recognition (PFR) task, whereas the LINDAHL_1.75 and SCOP_2.06 test sets are evaluated on the direct fold classification (DFC) task. We also provide the number of domains and folds evaluated at the family, superfamily and fold levels.

but a different class in the level immediately below. For example, at the superfamily level, two domains within a positive pair belong to the same superfamily class (and therefore the same fold class), but different family classes. Irrespective of the level, all negative pairs were evaluated, each one containing two protein domains from different fold classes. After the ranking and fold assignment, we computed the ratio of hits (*top 1 accuracy*), as well as the ratio of finding the correct fold class within the 5 top-ranked templates (*top 5 accuracy*).

2.4.2 Direct Fold Classification (DFC)

DFC Task. In the direct fold classification task, we evaluated the ability of our neural network-based models to classify the input test domains into K fold classes. This task was originally proposed in [57]. In this case, instead of extracting the supervised embeddings (*FoldEmb*), we obtained a score (i.e. logit) for each fold class from the last classification layer of our models (Figure 2), and the predicted fold as the one maximizing the class scoring vector.

Ensemble Strategies. As with the PFR task, we combined the best performing fine-tuning models to generate several ensembles. In this case, we followed the *soft voting ensemble* approach to get a better prediction for each tested protein domain. This strategy involves computing the vector of logits from each model and accumulating them in a new class scoring vector. Then, we assign the fold which maximizes this scoring vector for each query domain.

Evaluation. We also assessed the DFC task at the *family, superfamily* and *fold* levels. In contrast to the PFR task, now the full test set is split into three subsets, each one containing test domains that only share the specific level with domains from the training dataset. As performance metric, we consider the ratio of protein domains that were correctly classified (*top 1 accuracy*), as well as the ratio of finding the correct fold within the 5 top-scoring classes (*top 5 accuracy*).

2.4.3 Baseline Evaluation of Protein-LM Embeddings

To provide a baseline comparison, we also evaluated the PFR and DFC tasks directly with the pre-trained protein-LM embeddings (*LMEmb*). We used the cosine similarity metric for the protein-level embeddings *LMEmb-Prot*. In contrast, for the amino acid-level embeddings *LMEmb-AA*, we computed the soft symmetric alignment (SSA) proposed in [74] but using cosine similarity instead of L_1 distance. That is, given two sequences of embeddings $x_1, ..., x_{L_1}$ and $y_1, ..., y_{L_2}$, SSA is obtained as:

$$SSA = \frac{1}{A} \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} a_{ij} \cos(x_i, y_j), \tag{2}$$

where $A = \sum_{i=1}^{L_1} \sum_{j=1}^{L_2} a_{ij}$ acts as a normalization factor, and $a_{ij} = \alpha_{ij} + \beta_{ij} - \alpha_{ij}\beta_{ij}$ represent the alignment matrix, whose components are computed as:

$$\alpha_{ij} = \frac{e^{\cos(x_i, y_j)}}{\sum_{k=1}^{L_2} e^{\cos(x_i, y_k)}}, \beta_{ij} = \frac{e^{\cos(x_i, y_j)}}{\sum_{k=1}^{L_1} e^{\cos(x_k, y_j)}}. \tag{3}$$

Note that, while the baseline PFR task is performed using pairs from the test set only, the DFC one involves computing the fold similarity metric for each test embedding against all the training ones, and then assigning the fold class of the closest training domain.

Table 3: Performance of the *LMEmb* embeddings in the pairwise fold recognition (PFR) task, using the LINDAHL test set.

Embeddings	**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
LMEmb-AA $(L \times F)$						
UniRep	28.5	49.2	25.3	38.9	14.3	35.8
SeqVec	48.3	66.5	27.2	47.0	13.7	29.3
ESM-1b	7.4	14.8	2.1	6.7	0.6	8.1
ESM-MSA	27.9	47.7	13.8	29.5	12.1	22.4
ProtBERT	18.9	34.8	4.4	15.9	4.4	15.9
ProtT5	76.4	87.4	34.3	56.5	14.0	29.9
LMEmb-Prot (F)						
UniRep	45.6	60.9	35.0	47.7	19.3	35.8
SeqVec	62.3	77.8	$\underline{44.9}$	60.6	18.4	$\underline{37.7}$
ESM-1b	$\underline{81.6}$	88.5	$\underline{44.9}$	59.7	$\underline{21.2}$	37.1
ESM-MSA	76.6	88.1	42.9	54.4	16.2	28.0
ProtBERT	42.9	58.6	10.8	27.2	8.4	22.4
ProtT5	81.1	$\underline{90.8}$	40.3	$\underline{62.0}$	16.5	32.4

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. We compare the performance of the amino acid-level embeddings *LMEmb-AA* (using the SSA metric in Eq. 2) and protein-level embeddings *LMEmb-Prot* (using the cosine similarity metric). Underline indicates best performance.

2.5 Datasets

To assess the aforementioned tasks, we trained the fine-tuning models using 16133 protein domains from SCOPe version v2.06 [38], which are classified into $K = 1154$ folds. For PFR, we tested the models using the well-known LINDAHL dataset [13] containing 976 protein domains from 330 distinct folds. For the DFC task, we used the updated version of LINDAHL to SCOP v1.75 [44] (named LINDAHL_1.75), where, in order to directly classify the test domains, we keep only those (of the 976) that share their fold class with one of the K seen during training. This resulted in a test set with 871 domains from 244 folds.

In addition, for DFC we also evaluated the performance over the SCOP_2.06 test set [57], which contains 2533 protein domains from 550 folds. To avoid overlap between this test set and the training set described above (both are derived from SCOPe v2.06), in this particular case we used the training set proposed in [57], which contains 16712 domains classified into $K = 1195$ folds (from SCOP v1.75).

For each task and test set, Table 2 summarizes the number of individual protein domains and distinct folds evaluated at each level (family, superfamily and fold). In all cases, the protein domains within each training set share at most 95% sequence similarity. Moreover, the maximum sequence identity within each of the test sets is 40%, as well as with respect to their respective training sets.

3 Results and Discussion

3.1 Performance of Self-Supervised *LMEmb* Embeddings in PFR and DFC Tasks

We first evaluated how the self-supervised *LMEmb* embeddings perform in predicting structural similarity using the LINDAHL test set. In Table 3 we provide the pairwise fold recognition (PFR) results for the two types of embeddings, *LMEmb-AA* and *LMEmb-Prot*, from the 6 protein-LMs we considered. We find that aggregating embeddings across the protein length (*LMEmb-Prot*) helps in all cases when using cosine similarity. Note that, as can be seen in Table S1 (Supplementary Material), using L_1 distance as comparison metric shows similar results to cosine similarity for all *LMEmb-Prot* embeddings.

Amongst the *LMEmb-Prot* embeddings, the ESM-1b ones yield better overall PFR results, while the ProtBERT embeddings perform the worst at the three levels. We also notice differences across levels. For example, the ESM-1b and ProtT5 embeddings perform the best at the family level, with a high accuracy (81% top 1). This suggests that these embeddings could be used for homology searching when the amino acid sequence similarities are high. Furthermore, ESM-1b outperforms the rest of embeddings at the fold level (21.2%). Nevertheless, the accuracy values are generally low at this level. A similar trend is observed in Table S2 (Supplementary Material) for the direct fold classification (DFC) task evaluated on the LINDAHL_1.75 and SCOP_2.06 test sets. It should be noted that the overall poor performance of protein-LM embeddings at the fold level is to be expected, as they have been learned in a self-supervised manner without any information about the fold type.

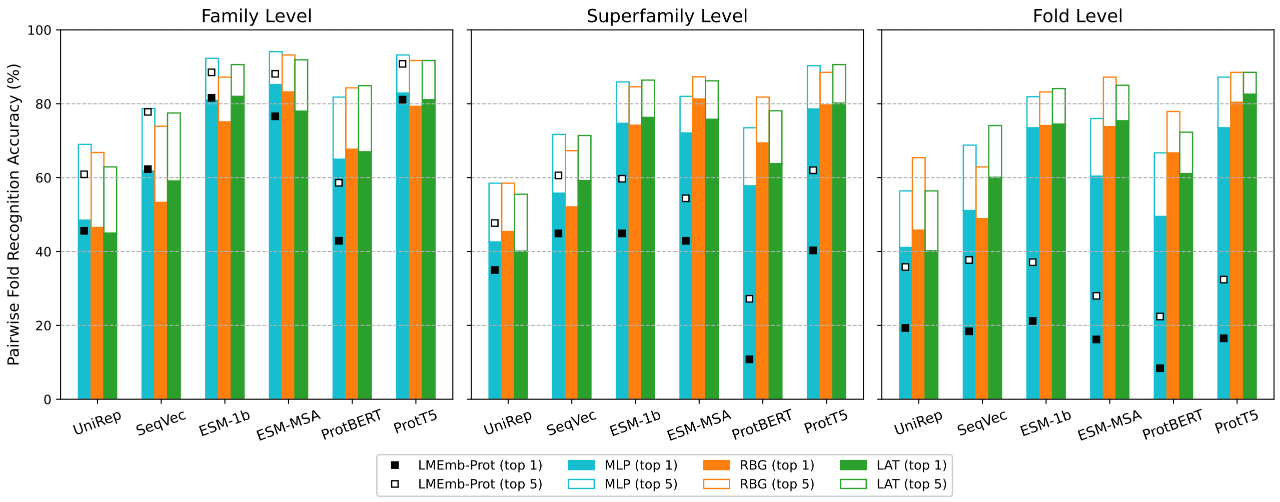

Figure 3: Pairwise fold recognition (PFR) accuracy (%) results on the LINDAHL test set. At each level (family, superfamily and fold), we compare the performance of the 6 protein-LM embeddings in Table 1 (UniRep, SeqVec, ESM-1b, ESM-MSA, ProtBERT, and ProtT5) fine-tuned by the 3 neural architectures from Figure 2 (MLP in cyan, RBG in orange, and LAT in green colored bars). For each one, top 1 accuracy is shown in filled bars, while top 5 accuracy is shown in empty bars. Baseline PFR results using the protein-level embeddings *LMEmb-Prot* are also included as square markers over the MLP bars (note that MLP uses these embeddings as input).

3.2 Performance of Fine-Tuned *FoldEmb* Embeddings in PFR Task

Figure 3 summarizes the PFR results of the fold-representative embeddings, *FoldEmb*, resulting from fine-tuning the *LMEmb* embeddings through neural network models: MLP, RBG and LAT. As a baseline, we also include the results obtained directly using the *LMEmb-Prot* embeddings (from Table 3). As can be observed, the *LMEmb* embeddings can be significantly enhanced by fine-tuning even when a simple MLP model is used for the task (*LMEmb-Prot* vs MLP in Figure 3), especially at the superfamily and fold levels. In general, after fine-tuning, the transformer-based protein-LM embeddings (ESM-1b, ESM-MSA, ProtBERT, ProtT5) show better PFR performance than the LSTM-based ones (UniRep, SeqVec). Regarding the supervised models, the RBG and LAT, both working on *LMEmb-AA*, provide better PFR results than MLP at the superfamily and fold levels. This contrasts with the results in Table 3 for *LMEmb-AA*, suggesting that the RBG and LAT models successfully exploit the protein sequence information possibly contained in these self-supervised embeddings. Thus, the top-performing model at the fold level is ProtT5 + LAT, which obtains 82.6% top 1 and 88.5% top 5 accuracy values. In contrast, our ESM-MSA + RBG model provides the best PFR results at the family and superfamily levels, with 83.2% and 81.3% top 1 accuracy, respectively. It is therefore clear that the fine-tuned *FoldEmb* embeddings are necessary to identify structural similarity in the hardest cases—when the sequence similarities are low.

We additionally performed an ablation study (see Table S3 in Supplementary Material) using the softmax cross-entropy as loss function instead of the LMCL. This modification leads to a performance drop for most combinations of input *LMEmb* and neural architecture, which is particularly noticeable at the fold level. This confirms that the LMCL is a more suitable loss function to learn a proper organization of the fold-representative *FoldEmb* vectors in the embedding space.

3.3 Performance of Fine-Tuning Models in DFC Task

We evaluated the ability of our neural network models to directly classify the input protein domains into fold classes (DFC task). The classification results for the LINDAHL_1.75 and SCOP_2.06 test sets are shown in Figure 4. As in the PFR task, we also include the results provided by the *LMEmb-Prot* embeddings as a baseline (from Table S2, Supplementary Material). The results for LINDAHL_1.75 in Figure 4a show a similar pattern to those in Figure 3 for the PFR task. However, in this case, the differences in performance between the family and fold levels are more pronounced. This behavior is expected. As test domains in the family subset share a higher sequence similarity with some domains from the training set, they are likely to be easier to predict for the models. This reinforces the idea that protein-LM embeddings capture sequence similarity in proteins. By contrast, the accuracy results at the fold level are indicative of the generalization capability of the models. Here we observe that the best performing model at both the superfamily and fold levels is ProtT5 + RBG (79.5% and 55.7% top 1 accuracy, respectively), followed by ProtT5 + LAT (77.6% and 50.0%). It is also worth noting the huge differences between top 1 and top 5 accuracies of some protein-LM embeddings in the fold subset. For

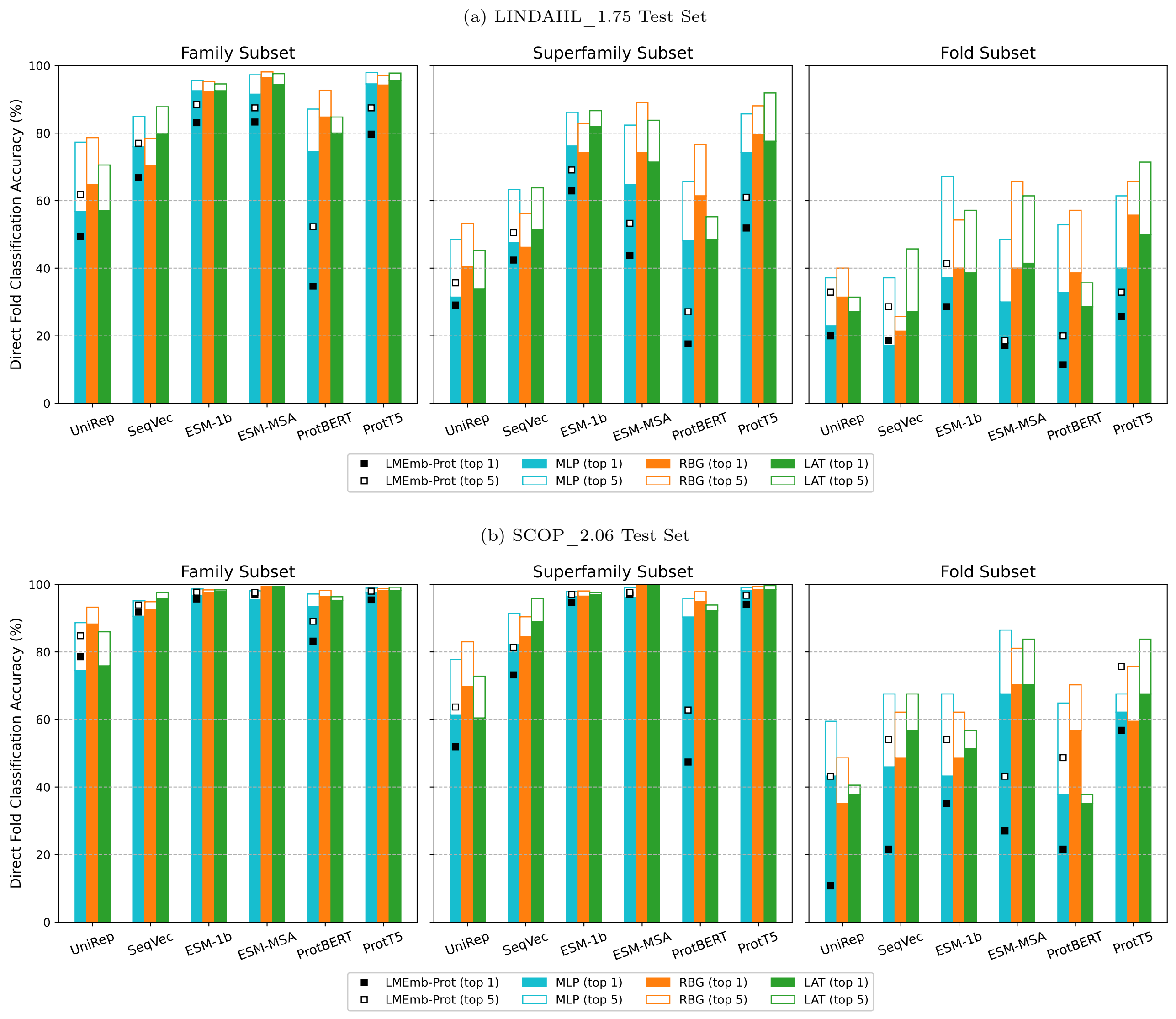

Figure 4: Direct fold classification (DFC) accuracy (%) results on the **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06 test sets. For each subset (family, superfamily and fold), we compare the performance of the 6 protein-LM embeddings in Table 1 (UniRep, SeqVec, ESM-1b, ESM-MSA, ProtBERT, and ProtT5) fine-tuned by the 3 neural architectures from Figure 2 (MLP in cyan, RBG in orange, and LAT in green colored bars). For each one, top 1 accuracy is shown in filled bars, while top 5 accuracy is shown in empty bars. Baseline DFC results using the protein-level embeddings *LMEmb-Prot* are also included as square markers over the MLP bars (note that MLP uses these embeddings as input).

example, for the ESM-MSA + RBG model the top 1 accuracy is 40.0%, whereas the top 5 accuracy reaches 65.7%.

Figure 4b shows the results of the DFC task on the SCOP_2.06 test set. As can be observed, the models predict the family and superfamily levels subsets with high accuracy (close to 100% in some cases). However, similarly to the LINDAHL_1.75 test set, the fold subset in SCOP_2.06 is also difficult to classify correctly. Nevertheless, our fine-tuned models outperform the original *LMEmb* embeddings. Compared to the rest of embeddings, ESM-MSA works particularly well here for all considered neural architectures (MLP, RBG and LAT), with top 1 accuracy values around 70% at the fold level.

3.4 Ensemble Approaches for Increased Accuracy in PFR and DFC Tasks

Given the good performance of the transformer-based ESM-1b, ESM-MSA, and ProtT5 embeddings processed by the RBG and LAT models, we now explore the benefits of integrating them through ensemble strategies for the two assessed tasks; using average ensemble for PFR, and soft voting ensemble for DFC. Provided that the predictions from individual models are sufficiently uncorrelated, we expect an increase in performance after ensembling. In Figure 5 we show the accuracy results at the fold level for the ensembling integration of all 6 aforementioned models (the 3 protein-LMs by the 2 neural architectures). This approach outperforms other integration options that can be found in the Supplementary Material (Table S4 for PFR and Table S5 for DFC).

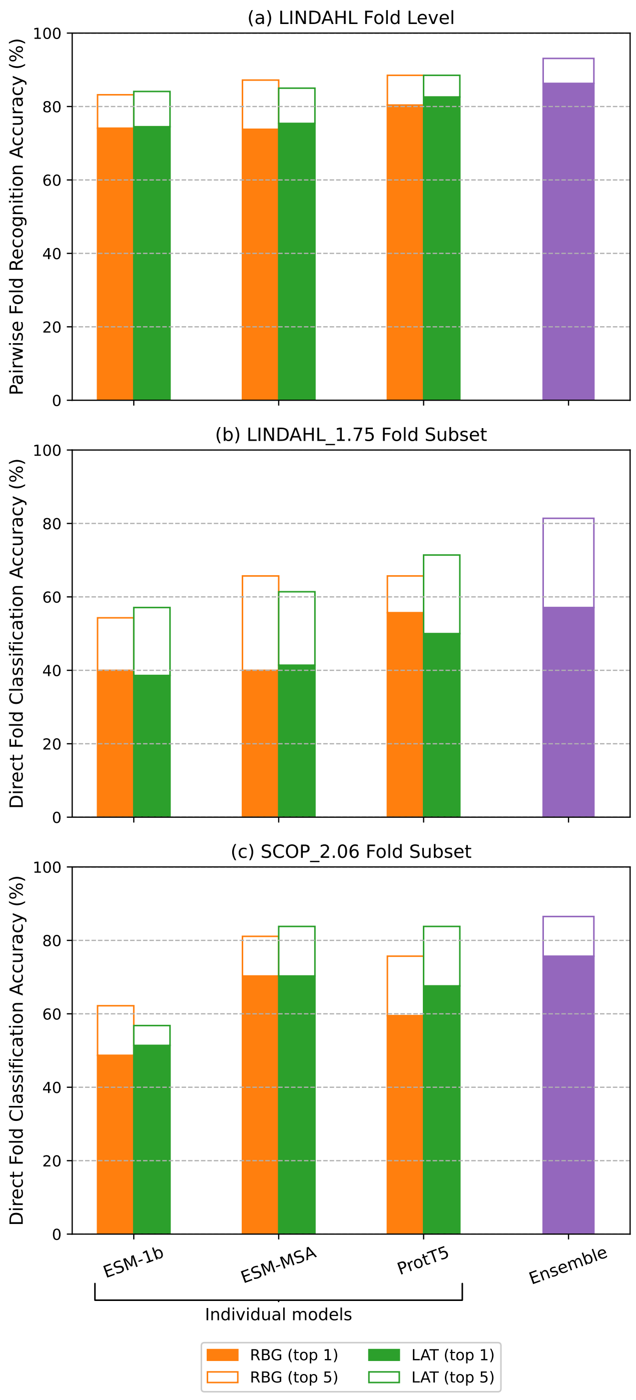

Figure 5: Ensemble strategy accuracy (%) results at the fold level for the **(a)** pairwise fold recognition (PFR) task using the LINDAHL test set, and the direct fold classification (DFC) task using both **(b)** LINDAHL_1.75 and **(c)** SCOP_2.06 test sets. Here the best performing ensemble strategy is shown (in purple), which involves the integration of 6 individual models. As a reference, the results of such models are also included—the 3 protein-LM embeddings (ESM-1b, ESM-MSA, and ProtT5) with neural architectures RBG (in orange) and LAT (in green). Top 1 and top 5 accuracies for each technique are represented in filled and empty bars, respectively.

As a reference, we also include the performance of the 6 individual models used in the ensemble.

For the PFR task, the LINDAHL test set is used to evaluate the ensemble strategy (Figure 5a). By simply averaging the cosine similarity scores we obtain 86.3% top 1 accuracy (93.1% top 5) at the fold level, almost 4 percentage points over the best individual model (ProtT5 + LAT). For the DFC task evaluated on LINDAHL_1.75 (Figure 5b), the ensemble approach obtains 57.1% top 1 accuracy at the fold level, slightly better than the best individual ProtT5 + RBG model (55.7%). This suggests that accurate classification at the fold level is still difficult even after ensembling. However, a noticeable improvement is observed when considering the top 5 accuracy (81.4% over the 65.7% in ProtT5 + RBG). Additionally, the ensemble approach yields better performance for the SCOP_2.06 test set (Figure 5c) in terms of both top 1 and top 5 accuracy (75.7% and

Table 4: Three-level LINDAHL pairwise fold recognition (PFR) results of the best individual model and the ensembling strategy (average ensemble), in comparison with the state-of-the-art.

Method	Family		Superfamily		Fold	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PSI-BLAST [13]	71.2	72.3	27.4	27.9	4.0	4.7
HHpred [23]	82.9	87.1	58.0	70.0	25.2	39.4
RAPTOR [23]	**86.6**	89.3	56.3	69.0	38.2	58.7
BoostThreader [23]	86.5	90.5	66.1	76.4	42.6	57.4
SPARKS-X [24]	84.1	90.3	59.0	76.3	45.2	67.0
FOLDpro [31]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [31]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [31]	84.5	91.2	61.5	76.5	33.6	60.7
RFDN-Fold [31]	84.7	91.5	65.7	78.8	37.7	61.7
MRFalign [38]	85.2	90.8	72.4	80.9	38.6	56.7
CEthreader [44]	76.6	87.2	69.4	81.8	52.3	70.4
DeepFR (s1) [38]	67.4	80.9	47.0	63.4	44.5	62.9
DeepFR (s2) [38]	65.4	83.4	51.4	67.1	56.1	70.1
CNN-BGRU [44]	71.0	87.7	60.1	77.2	58.3	78.8
VGGfold [42]	67.9	84.3	53.2	68.4	58.3	73.5
FoldTR [43]	55.5	79.8	62.4	78.6	62.6	82.6
FoldHSphere [45]	76.4	89.2	72.8	86.4	75.1	84.1
ProtT5 + LAT	81.1	91.7	80.2	90.6	82.6	88.5
Ensemble Strategy	86.5	**94.6**	**81.1**	**90.8**	**86.3**	**93.1**

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. Boldface indicates best performance.

Table 5: Three-level LINDAHL pairwise fold recognition (PFR) results of the random forest (RF) ensemble, in comparison with the state-of-the-art.

Method	Family		Superfamily		Fold	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
DeepFRpro (s1) [38]	**85.6**	91.9	66.6	82.0	57.6	73.8
DeepFRpro (s2) [38]	83.1	92.3	69.6	82.5	66.0	78.8
CNN-BGRU-RF+ [44]	85.4	93.5	73.3	87.8	76.3	85.7
FoldTRpro [43]	83.8	92.8	76.0	89.2	58.3	87.2
FSD_XGBoostpro [43]	82.7	**94.6**	77.9	91.5	68.2	91.9
FoldHSpherePro [45]	85.2	93.0	79.0	89.2	81.3	90.3
RF Ensemble	79.6	92.8	**82.5**	**92.4**	**90.3**	**94.4**

The top 1 and top 5 accuracy (%) results are provided at the family, superfamily and fold levels. Boldface indicates best performance.

86.5%, respectively), that is, more than 5 percentage points over the best individual model (ESM-MSA + RBG).

When taking into account the family and superfamily levels (Tables S4 and S5, Supplementary Material), a performance boost over the individual models is also achieved by the ensembling approach in both tasks (PFR and DFC).

3.5 Comparison with State-Of-The-Art Methods for Fold Recognition and Fold Classification

Finally, we compare the performance of our best individual models, as well as the ensemble strategy we propose, against the state-of-the-art results for fold recognition and fold classification. First, we compare to several methods intended for the PFR task, which can be grouped into three categories: (i) alignment and threading methods such as PSI-BLAST [104], HHpred [20], RAPTOR [23], BoostThreader [22], SPARKS-X [24], MRFalign [21], and CEthreader [28]; (ii) machine learning methods such as FOLDpro [29], RF-Fold [31], DN-Fold [31], RFDN-Fold [31]; and (iii) deep learning methods such as DeepFR [38], CNN-BGRU [44] VGGfold [42], FoldTR [43], and FoldHSphere [45]. Table 4 shows the PFR accuracy results achieved by these methods on the LINDAHL test set, as well as the best performing model ProtT5 + LAT and the average ensemble. We notice that ProtT5 + LAT alone outperforms all state-of-the-art methods at the superfamily and fold levels. At the family level, it also obtains better accuracy than the rest of deep learning-based methods which, at this level, tend to perform worse than alignment methods. Furthermore, as we discussed in the previous section, the ensemble strategy shows a performance boost at the fold level, but also at the family level. At this level, it achieves a top 1 accuracy similar to the best alignment methods, and clearly outperforms all in top 5. Therefore, the use of the protein-LM embeddings as the input representation not only bridges the gap at the family and fold levels, but also increases the performance of fold recognition consistently at all levels.

In addition, in order to compare with DeepFRpro [38], CNN-BGRU-RF+ [44], FoldTRpro [43], FSD_XGBoostpro

Table 6: Three-level **(a)** LINDAHL_1.75 and **(b)** SCOP_2.06 direct fold classification (DFC) results of the best individual models and the ensembling strategy (soft voting ensemble), in comparison with the state-of-the-art.

(a) LINDAHL_1.75 Test Set

Method	**Full Set**		**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
DeepFR (s1)	57.3	72.5	70.7	83.9	34.8	55.7	11.4	25.7
DeepFR (s2)	66.5	76.0	79.5	86.5	45.2	57.6	20.0	42.9
CNN-BGRU	70.3	85.1	80.7	92.7	48.6	71.0	47.1	62.9
FoldHSphere	82.0	90.9	92.7	97.3	66.2	81.4	38.6	65.7
ProtT5 + RBG	87.6	92.4	94.3	97.1	79.5	88.1	55.7	65.7
Ensemble Strategy	**92.3**	**97.5**	**97.6**	**99.3**	**89.1**	**97.6**	**57.1**	**81.4**

(b) SCOP_2.06 Test Set

Method	**Full Set**		**Family**		**Superfamily**		**Fold**	
	Top 1	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
DeepFR (s1)	89.2	94.2	88.1	93.9	91.1	95.4	24.3	37.8
DeepFR (s2)	91.2	95.6	91.1	95.8	92.3	96.4	37.8	54.1
DeepSF [57]	73.0	90.3	75.9	91.8	72.2	90.1	51.4	67.6
CNN-BGRU	89.9	96.5	91.5	96.8	90.1	97.0	48.7	64.9
FoldHSphere	96.5	98.3	97.2	98.4	97.1	98.9	51.4	67.6
ESM-MSA + RBG	99.2	99.4	**99.5**	99.5	99.7	99.8	70.3	81.1
Ensemble Strategy	**99.3**	**99.6**	99.3	**99.6**	**99.8**	**99.9**	**75.7**	**86.5**

The top 1 and top 5 accuracy (%) results are provided for the full test set, and the family, superfamily and fold subsets. Boldface indicates best performance per set.

[43], and FoldHSphere [45], which apply a random forest (RF) ensemble, we implemented and tested the same RF strategy in our ensemble approach (see Materials and Methods section). It must be noted that these results cannot be directly compared to the previous ones, as this approach involves additional training of the RF models on the test set in a 10-stage cross-testing manner. From Table 5 we can see that the RF ensemble introduced here consistently outperforms all previous state-of-the-art methods at the superfamily and fold levels. However, it provides lower accuracy at the family level. Interestingly, this evaluation scenario seems to lead to unexpected results due to cross-testing. That is why we believe average ensembling provides more reliable results than the RF ensemble, and can be compared more fairly against the individual models.

For the DFC task we compare to deep learning methods that allow for the direct classification of protein domains into different folds. In Table 6 we show the results for the LINDAHL_1.75 and SCOP_2.06 test sets. Using both sets we evaluated the state-of-the-art DeepFR, CNN-BGRU, and FoldHSphere methods. In addition, for SCOP_2.06 we include the results of the DeepSF method [57], which originally introduced the SCOP_2.06 set and the DFC task. In the case of LINDAHL_1.75 (Table 6a), we observe that the top-performing ProtT5 + RBG model obtains better results than previous methods at all three levels, considerably outperforming them at the superfamily and fold levels. For SCOP_2.06 (Table 6b), the top-performing ESM-MSA + RBG model seems to fit the family and superfamily subsets almost perfectly, obtaining accuracy values above 99%. It also generalizes better than previous methods in the fold subset, with a top 1 accuracy of 70.3% (81.1% top 5) which is more than 15 percentage points higher than the previous methods DeepSF and FoldHSphere (both obtained 51.5% top 1 and 65.7% top 5 accuracies). Moreover, as discussed before, the ensemble strategy already outperformed the best individual models on both test sets and, therefore, all the considered methods from the state-of-the-art.

Taken together, these results show the superiority of protein-LM embeddings over other sequence representations such as the PSSM and secondary structure (DeepSF, CNN-BGRU, and FoldHSphere), or two-dimensional representations such as contact maps (DeepFR, VGGfold, and FoldTR).

4 Conclusion

This work provides a comparative analysis of different pre-trained embeddings from protein language models (protein-LMs) in protein fold prediction. To do so, we fine-tuned the protein-LM embeddings (*LMEmb*) through several state-of-the-art neural network models using fold labels for supervised training. The performance was evaluated in two differentiated tasks: pairwise fold recognition (PFR) and direct fold classification (DFC). For the PFR task, we extracted a fold-representative embedding vector (*FoldEmb*), which we used to predict the protein fold class by pairwise comparison with known protein domains. In contrast, in the DFC task we directly mapped the protein domain into one of the more than thousand existing fold classes in the SCOP database. For both tasks, the protein-LM embeddings learned by pre-trained transformer-based models proved to be more effective at identifying the protein fold than those extracted from LSTM-based models. Thus, the ESM-MSA and ProtT5 amino acid-level embeddings in combination with the RBG and LAT architectures provided the best PFR and DFC results. Compared to the state-of-the-art, these models alone were able to predict the fold class with higher accuracy at the three levels—family, superfamily and fold. Furthermore, our proposed ensemble approaches provided a significant performance boost over these individual models and thus over the current state-of-the-art. These results demonstrate the suitability of protein-LM embeddings over other traditional protein representations for fold prediction.

5 Funding

This work has been supported by the project PID2019-104206GB-I00 funded by MCIN/ AEI /10.13039/ 501100011033, as well as the FPI grant BES-2017-079792.

References

[1] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[2] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[3] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[4] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 2021.

[5] Alexey G. Murzin, Steven E. Brenner, Tim Hubbard, and Cyrus Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *journal of Molecular Biology*, 247(4):536–540, 1995.

[6] John-Marc Chandonia, Lindsey Guan, Shiangyi Lin, Changhua Yu, Naomi K. Fox, and Steven E. Brenner. SCOPe: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 2021.

[7] Christine A. Orengo, Alex D. Michie, Susan Jones, David T. Jones, Mark B. Swindells, and Janet M. Thornton. CATH — a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[8] Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P. Waman, Paul Ashford, Harry M. Scholes, Camilla S. M. Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, et al. CATH: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[9] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[10] Stephen K. Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich, Li Chen, Gregg V. Crichlow, Cole H. Christie, Kenneth Dalenberg, Luigi Di Costanzo, Jose M. Duarte, et al. RCSB Protein Data Bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Research*, 49(D1):D437–D451, 2021.

[11] Cyrus Chothia and Alexei V. Finkelstein. The classification and origins of protein folding patterns. *Annual Review of Biochemistry*, 59(1):1007–1035, 1990.

[12] David T. Jones, W. R. Taylor, and Janet M. Thornton. A new approach to protein fold recognition. *Nature*, 358(6381):86, 1992.

[13] Erik Lindahl and Arne Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295(3):613–625, 2000.

[14] R. Dustin Schaeffer and Valerie Daggett. Protein folds and protein folding. *Protein Engineering, Design & Selection*, 24(1-2):11–19, 2010.

[15] Rachel Kolodny, Leonid Pereyaslavets, Abraham O. Samson, and Michael Levitt. On the universe of protein folds. *Annual Review of Biophysics*, 42:559–582, 2013.

[16] Mohammed S. Abual-Rub and Rosni Abdullah. A survey of protein fold recognition algorithms. *Journal of Computer Science*, 4(9):768–776, 2008.

[17] Junjie Chen, Mingyue Guo, Xiaolong Wang, and Bin Liu. A comprehensive review and comparison of different computational methods for protein remote homology detection. *Briefings in Bioinformatics*, 19(2):231–244, 2018.

[18] Katarzyna Stapor, Irena Roterman-Konieczna, and Piotr Fabian. Machine learning methods for the protein fold recognition problem. In *Machine Learning Paradigms*, volume 149, pages 101–127. Springer, 2019.

[19] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[20] Johannes Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[21] Jianzhu Ma, Sheng Wang, Zhiyong Wang, and Jinbo Xu. MRFalign: Protein homology detection through alignment of Markov random fields. *PLoS Computational Biology*, 10(3):e1003500, 2014.

[22] Jinbo Xu, Ming Li, Dongsup Kim, and Ying Xu. RAPTOR: optimal protein threading by linear programming. *journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[23] Jian Peng and Jinbo Xu. Boosting protein threading accuracy. In *Annual International Conference on Research in Computational Molecular Biology*, pages 31–45, 2009.

[24] Yuedong Yang, Eshel Faraggi, Huiying Zhao, and Yaoqi Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, 2011.

[25] Jianzhu Ma, Jian Peng, Sheng Wang, and Jinbo Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):i59–i66, 2012.

[26] Juan A. Morales-Cordovilla, Victoria Sanchez, and Martin Ratajczak. Protein alignment based on higher order conditional random fields for template-based modeling. *PLoS ONE*, 13(6):e0197912, 2018.

[27] Daniel W. A. Buchan and David T. Jones. EigenTHREADER: analogous protein fold recognition by efficient contact map threading. *Bioinformatics*, 33(17):2684–2690, 2017.

[28] Wei Zheng, Qiqige Wuyun, Yang Li, SM Mortuza, Chengxin Zhang, Robin Pearce, Jishou Ruan, and Yang Zhang. Detecting distant-homology protein structures by aligning deep neural-network based contact maps. *PLoS Computational Biology*, 15(10):1–27, 2019.

[29] Jianlin Cheng and Pierre Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22(12):1456–1463, 2006.

[30] Taeho Jo and Jianlin Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):S14, 2014.

[31] Taeho Jo, Jie Hou, Jesse Eickholt, and Jianlin Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:17573, 2015.

[32] Ke Yan, Xiaozhao Fang, Yong Xu, and Bin Liu. Protein fold recognition based on multi-view modeling. *Bioinformatics*, 35(17):2982–2990, 2019.

[33] Ke Yan, Jie Wen an Yong Xu, and Bin Liu. Protein fold recognition based on auto-weighted multi-view graph embedding learning model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[34] Ke Yan, Jie Wen, Yong Xu, and Bin Liu. MLDH-Fold: Protein fold recognition based on multi-view low-rank modeling. *Neurocomputing*, 421:127–139, 2021.

[35] Bin Liu, Yulin Zhu, and Ke Yan. Fold-LTR-TCP: protein fold recognition based on triadic closure principle. *Briefings in Bioinformatics*, 2019.

[36] Jiangyi Shao, Ke Yan, and Bin Liu. FoldRec-C2C: protein fold recognition by combining cluster-to-cluster model and protein similarity network. *Briefings in Bioinformatics*, 2020.

[37] Jiangyi Shao and Bin Liu. ProtFold-DFG: protein fold recognition by combining Directed Fusion Graph and PageRank algorithm. *Briefings in Bioinformatics*, 2020.

[38] Jianwei Zhu, Haicang Zhang, Shuai Cheng Li, Chao Wang, Lupeng Kong, Shiwei Sun, Wei-Mou Zheng, and Dongbo Bu. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contacts. *Bioinformatics*, 33(23):3749–3757, 2017.

[39] Bin Liu, Chen-Chen Li, and Ke Yan. DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in Bioinformatics*, 2019.

[40] Chen-Chen Li and Bin Liu. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Briefings in Bioinformatics*, 2019.

[41] Yihe Pang and Bin Liu. SelfAT-Fold: protein fold recognition based on residue-based and motif-based self-attention networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[42] Yan Liu, Yi-Heng Zhu, Xiaoning Song, Jiangning Song, and Dong-Jun Yu. Why can deep convolutional neural networks improve protein fold recognition? a visual explanation by interpretation. *Briefings in Bioinformatics*, 2021.

[43] Yan Liu, Ke Han, Yi-Heng Zhu, Ying Zhang, Long-Chen Shen, Jiangning Song, and Dong-Jun Yu. Improving protein fold recognition using triplet network and ensemble deep learning. *Briefings in Bioinformatics*, 22(6):bbab248, 2021.

[44] Amelia Villegas-Morcillo, Angel M. Gomez, Juan A. Morales-Cordovilla, and Victoria Sanchez. Protein fold recognition from sequences using convolutional and recurrent neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2848–2854, 2021.

[45] Amelia Villegas-Morcillo, Victoria Sanchez, and Angel M. Gomez. FoldHSphere: deep hyperspherical embeddings for protein fold recognition. *BMC Bioinformatics*, 22(1):1–21, 2021.

[46] Leyi Wei and Quan Zou. Recent progress in machine learning-based methods for protein fold recognition. *International journal of Molecular Sciences*, 17(12):2118, 2016.

[47] Chris H. Q. Ding and Inna Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[48] Hong-Bin Shen and Kuo-Chen Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, 22(14):1717–1722, 2006.

[49] Qiwen Dong, Shuigeng Zhou, and Jihong Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 25(20):2655–2662, 2009.

[50] Jian-Yi Yang and Xin Chen. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins: Structure, Function, and Bioinformatics*, 79(7):2053–2064, 2011.

[51] James Lyons, Abdollah Dehzangi, Rhys Heffernan, Yuedong Yang, Yaoqi Zhou, Alok Sharma, and Kuldip Paliwal. Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. *IEEE Transactions on Nanobioscience*, 14(7):761–772, 2015.

[52] Daozheng Chen, Xiaoyu Tian, Bo Zhou, and Jun Gao. ProFold: Protein fold classification with additional structural features and a novel ensemble classifier. *BioMed Research International*, 2016:1–10, 2016.

[53] Jiaqi Xia, Zhenling Peng, Dawei Qi, Hongbo Mu, and Jianyi Yang. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics*, 33(6):863–870, 2016.

[54] Wisam Ibrahim and Mohammad S. Abadeh. Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis. *Neural Computing and Applications*, 31(8):4201–4214, 2019.

[55] Sanjay Bankapur and Nagamma Patil. An enhanced protein fold recognition for low similarity datasets using convolutional and skip-gram features with deep neural network. *IEEE Transactions on NanoBioscience*, 20(1):42–49, 2020.

[56] Wessam Elhefnawy, Min Li, Jianxin Wang, and Yaohang Li. DeepFrag-k: a fragment-based deep learning approach for protein fold recognition. *BMC Bioinformatics*, 21(6):1–12, 2020.

[57] Jie Hou, Badri Adhikari, and Jianlin Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[58] Ming Zhou, Nan Duan, Shujie Liu, and Heung-Yeung Shum. Progress in neural nlp: Modeling, learning, and reasoning. *Engineering*, 6(3):275–290, 2020.

[59] Dan Ofer, Nadav Brandes, and Michal Linial. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 19:1750–1758, 2021.

[60] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J. Martin, Karine Michoud, Claire O'Donovan, Isabelle Phan, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[61] Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A. Salazar, Erik L. L. Sonnhammer, Silvio C. E. Tosatto, Lisanna Paladin, Shriya Raj, Lorna J. Richardson, et al. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, 2021.

[62] Baris E. Suzek, Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, Cathy H. Wu, and UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[63] UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 2021.

[64] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):1–8, 2018.

[65] Martin Steinegger, Milot Mirdita, and Johannes Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, 16(7):603–606, 2019.

[66] Ehsaneddin Asgari and Mohammad R. K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11):e0141287, 2015.

[67] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[68] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[70] Nils Strodthoff, Patrick Wagner, Markus Wenzel, and Wojciech Samek. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8):2401–2409, 2020.

[71] Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, 16(12):1315–1322, 2019.

[72] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[73] Michael Heinzinger, Ahmed Elnaggar, Yu Wang, Christian Dallago, Dmitrii Nechaev, Florian Matthes, and Burkhard Rost. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, 20(1):1–17, 2019.

[74] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. In *International Conference on Learning Representations*, 2019.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[76] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with TAPE. In *Advances in neural information processing systems*, 2019.

[77] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

[78] Roshan M. Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856, 2021.

[79] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Wang Yu, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2021.

[80] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[81] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 2978–2988, 2019.

[82] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 2019.

[83] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[84] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: a lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[85] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[86] Hannes Stärk, Christian Dallago, Michael Heinzinger, and Burkhard Rost. Light attention predicts protein location from the language of life. *Bioinformatics Advances*, 11 2021. vbab035.

[87] Amelia Villegas-Morcillo, Stavros Makrodimitris, Roeland C. H. J. van Ham, Angel M. Gomez, Victoria Sanchez, and Marcel J. T. Reinders. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162–170, 2021.

[88] Maria Littmann, Michael Heinzinger, Christian Dallago, Tobias Olenyi, and Burkhard Rost. Embeddings from deep learning transfer GO annotations beyond homology. *Scientific reports*, 11(1):1–14, 2021.

[89] Irene van den Bent, Stavros Makrodimitris, and Marcel Reinders. The power of universal contextualized protein embeddings in cross-species protein function prediction. *Evolutionary Bioinformatics*, 17:1–15, 2021.

[90] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. In *Advances on Neural Information Processing Systems*, 2021.

[91] Celine Marquet, Michael Heinzinger, Tobias Olenyi, Christian Dallago, Michael Bernhofer, Kyra Erckert, Dmitrii Nechaev, and Burkhard Rost. Embeddings from protein language models predict conservation and variant effects. *Research Square preprint*, 2021.

[92] Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Rajani. BERTology meets biology: Interpreting attention in protein language models. In *International Conference on Learning Representations*, 2021.

[93] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J. Haunsberger, and Johannes Söding. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, 20(1):1–15, 2019.

[94] Milot Mirdita, Lars von den Driesch, Clovis Galiez, Maria J. Martin, Johannes Söding, and Martin Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Research*, 45(D1):D170–D176, 2017.

[95] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.

[96] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[97] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[98] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[99] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[100] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[101] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5265–5274, 2018.

[102] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[103] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[104] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.