META-EMBEDDING FOR UNSUPERVISED PROTEIN EMBEDDING MODELS

Jules Berman, Michael Liu, and Serhat Bakirtas

November 2021

ABSTRACT

Recently, the focus of the protein ML research has shifted towards unsupervised representation learning with models borrowed from the NLP space. With respect to word embeddings it has been shown that these models capture complementary features and ensembling can increase the performance on a variety of tasks. We show that the similar performance gains can be made by using these ensemble methods applied protein embeddings generated from unsupervised protein models. Specifically we use source embedding from the UniRep, Elmo, and Transformer models and we generate meta-embeddings using concatenation, averaging, dimensionality reduction, and various autoencoders. We compare the performance of these meta-embeddings against baselines in a remote homology classification task. Most of the ensemble methods outperform the individual source embeddings and the benchmark. The best performance is observed when a weighted concatenation of the embeddings is taken as the meta-embedding. Applying SVD on this weighted concatenation is found to be robust, performing close to the benchmark, with an almost 95% reduction in dimensionality. Overall these results show that unsupervised protein models learn a non-negligible set of complementary features and generating meta-embeddings is a robust way to increase performance on downstream biological tasks.

1 Introduction

Traditional machine learning research on protein modeling has focused on building supervised models that learn from specific classification tasks. These models have often been built by representing proteins with features which require algorithmically complex preprocessing of protein data (BLAST [1], MSA, etc.) or experimentally verified labeling (e.g. Swissprot). But due to the handcrafted nature of these feature choices these methods have been unable to take advantage of the exponential growth of unlabeled protein sequence data over the past couple years which can be seen in protein databases such as Uniprot [2]. Inspired by the successes of unsupervised models in the NLP space [3], recent research in protein ML has focused on the task of unsupervised representation learning. That is, learning meaningful distributed representations, known as embeddings, of proteins based on their amino-acid sequences alone. These models can take advantage of the vast amount of unlabeled protein data and have been shown to produce protein embeddings which are useful on a wide range of downstream tasks [4].

While the potential universality of using unsupervised protein embeddings for all downstream tasks has been argued in the protein ML literature [5, 6], no single model or architecture has been shown to have supremacy with regard to all downstream tasks [4]. The variance in performances suggests that different models may capture different biologically relevant features in the embedding space. This leads to the natural question of whether these features are complementary. That is, whether *ensembling* embeddings would achieve better results than an individual embedding alone.

NLP research gives good reason to believe this to be the case. There is substantial work on the analogous problem of ensembling word embeddings with the vocabulary of a particular language. Bansal *et al.* has shown that tailoring the representations into *meta-embeddings* leads to further improvements [7]. Further experiments have shown that considerable accuracy improvements are possible with ensembles of embeddings [8] [9].

In this project we draw inspiration from NLP ensemble methods and apply them to the domain of proteins. We employ a variety of ensemble methods over three pretrained protein embedding models to create meta-embeddings and assess their performance on a key downstream task, that of remote homology classification. We show that these meta-embeddings can easily outperform the individual embeddings with regard to remote homology.

The structure of this report is as follows: in Section 2, we describe the remote homology task and its corresponding dataset, as well as the pretrained protein embedding models used to create the source embeddings. In Section 3 we present the ensembling techniques we used to create meta-embeddings. In Section 4, we present the experiments and discuss the results. Finally we derive conclusions in Section 5 and their relevance to protein ML.

2 Task

As protein embedding models have become an active area of research, Rao *et al.* introduced a standardized set of five classification tasks spread across different domains of protein biology: secondary structure prediction, stability prediction, evolutionary classification, and protein function prediction, collectively known as the Tasks Assessing Protein Embeddings (TAPE) [4] The authors provide curated training, validation, and test splits which ensure that, "each task tests biologically relevant generalization that transfers to real-life scenario". In addition they benchmark many state of the art protein embedding models on these tasks which serve as the baseline for our experiments. This paper focuses specifically on the remote homology task.

2.1 Remote Homology

Proteins are considered homologous when they share a common evolutionary ancestor, which is taken as an indicator of similar biological structure and function. This relationship can often be easily inferred by residue level sequences similarity detected through methods such as MSA or BLAST. Remote homology however presents a more challenging problem where proteins may still have a meaningful evolutionary, structural, and/or functional relationship but low sequence similarity. So a model's performance on this task is a strong indicator that it has captured meaningful biological features about a protein's structure and function beyond those indicated by sequence similarity. This being one of the key goals of a successful protein embedding model. In addition the problem has many explicit applications such as the classification of enzymes [10], antibiotic-resistant gene detection [11], and many others.

2.2 Dataset

The remote homology problem is formalized as a sequence classification problem, where an input protein (i.e. an amino acid sequence) is assigned to a class representing its protein fold (a key aspect of its structure and function).

We use the dataset given in [12] as prepared by [4]. In this set, each protein is assigned to one of the 1195 known homology classes according to SCOP (Structural Classification of Proteins) [13], which is a manually annotated protein database. The dataset is split into a training part consisting of 12312 proteins and a validation set of size 736. The test set is split into three parts consisting of 718 proteins in the fold-level, 1254 proteins in the superfamily-level and 1272 proteins in the family-level. The "family" set contains the most closely related protein, while "superfamily" is less related and "fold" is the least related. Thus the test sets get respectively harder and are indicators of a model's ability to generalize to unseen distributions.

2.3 Models

In this paper, we use three embedding models, UniRep by Alley *et. al.* [14], Elmo described in [15], and the Transformer as implemented by Rao *et. al.* [4]. All models are trained on millions of unlabeled protein sequences using unsupervised or semi-supervised methods. We then use these pretrained model weights to produce source embeddings over our dataset. Each model outputs an embedding for each amino acid in a protein sequence. We then average these together to get a single embedding for a single protein (except in the case of CNN which is discussed later).

2.3.1 UniRep

The Unified Representation (UniRep) *is a versatile protein summary that can be applied across protein engineering informatics.* The model is trained using next residue prediction, while minimizing cross-entropy loss. The model architecture is a 1900-hidden unit mLSTM trained on 24 million raw primary amino acid sequences from the UniRef50 database [14] [16]. This model outputs an embedding of dimension $d_1 = 1900$.

2.3.2 Elmo

The Elmo model we use is based on the NLP model Embedding from Language Models (Elmo) given in [17]. The model consists of a character-level Convolutional Neural Network followed by two layers of bidirectional LSTMs. The CNN is used to embed each residue (amino acid) into the latent space, where the LSTMs are used to model the context of the surrounding residues [15]. The model is pretrained on the Pfam [18] database consisting of 31 million unlabeled proteins. This model outputs embeddings with dimension of $d_2 = 1024$.

2.3.3 Transformer

We use the transformer model implemented in Rao *et. al.* [4]. The transformer is a 12-layer one with 512 hidden units and 8 attention heads, leading to a 38M-parameter model. This model is, like Elmo, is pretrained on the Pfam database. It outputs representations in the dimension of $d_3 = 768$.

2.3.4 Meta-Embedding Classifier

In order to evaluate an embedding against a specific objective there must be some downstream model which maps the embeddings to specific classes in a classification task. While in the context of the remote homology tasks our baseline [4] uses a fairly complex attention based model, there is evidence that with good embedding simpler models can be used to solve downstream tasks [17]. In order to reduce compute time and the number of variables at play while evaluating ensemble methods we

chose to use a simple logistic regression model to classify all embeddings for the remote homology task.

3 Ensemble Methods

In this project, we explored many different ensemble methods over our three source embedding. These methods are, concatenation (CONC), weighted concatenation (W-CONC), averaging (AVG), SVD, and various auto encoder architectures.

3.1 CONC

As suggested by Yin and Schütze [8], CONC is a method where we simply concatenate the embedding vectors obtained from different models after performing L2 normalization on each vector. Suppose we have r embeddings with dimensions $d_1, ..., d_r$. For each protein, CONC outputs a new embedding vector of dimension $d_{conc} = d_1 + ... + d_r$ which consists of r source embeddings, each with unit norm.

Coates and Bollegala [19] provide a theoretical interpretation of CONC in the following way: concatenation is the summing of zero-padded vectors where the padded zeros for each embedding are located in the dimension where other embeddings are potentially non-zero. This enforces all the zero-padded embedding vectors to be orthogonal.

To explain this we consider the case of concatenating two vectors, although the result can be easily generalised to more than two source embeddings. These two vectors are represented by two source embeddings S_1, S_2 with Euclidean distances E_1, E_2 in the respective representation spaces. The orthogonality imposed by zero-padding, in turn implies that the Euclidean distance E_{CONC} in the new representation space is

$$E_{CONC} = \sqrt{E_1^2 + E_2^2}$$

Thus, concatenation -at the expense of increased dimensionality- benefits from *i*) the increase in the Euclidean distance with respect to the original source embeddings and *ii*) the diversity introduced by multiple source embeddings. The latter can be explained as follows: take two words u, v of distinct classes, then the probability of u and v being falsely close in two or more representations is lower than the case when we have just one representation.

3.2 Weighted CONC (W-CONC)

Different source embeddings clearly will yield different performances on any downstream task. If we want to make use of the differences in the individual performances, it is suggested that we assign different weights to the source embeddings [8]. The weighting is applied to each embedding after the L2 normalization and then these weighted vectors are concatenated. The weights can be considered as hyper-parameters, which can be determined by validation or picked according to prior beliefs based on the performance of source embedding for the task at hand. In our case we weighted Elmo by a factor of 6, transformer by a factor of 3, and Unirep by a factor of 1. This weighting is directly correlated with each source embedding's individual performance on the remote homology task.

3.3 Averaging (AVG)

Another simple ensembling method discussed by Coates and Bollegala in [19], is taking the average of the source embedding vectors. In the presence of dimension mismatch, we apply zero-padding to the embedding vectors of lower dimensions. In addition we weight each source embedding using the weights given in W-CONC.

Consider two words u, v and two source embeddings S_1, S_2 with Euclidean distances E_1, E_2 in the respective representation spaces. Then the Euclidean distance E_{AVG} in the new representation space is

$$E_{AVG} = \frac{1}{2}\sqrt{E_1^2 + E_2^2 - 2E_1E_2\cos(\theta)}$$

Unlike CONC, we have a term including the angle between the (potentially zero-padded) source embedding vectors. Here Coates and Bollegala emphasize that source embeddings typically are of large dimensions, containing thousands of words, and the angles between embedding vectors are sufficiently random, and they argue and show empirically that the angle θ follows a Gaussian distribution with mean $\frac{\pi}{2}$, implying

$$\mathbb{E}\left[E_{AVG}\right] \approx \frac{1}{2}E_{CONC}$$

This means that AVG may approximate CONC without the drawback of increased dimensionality. We explore this result in the context of our source embeddings.

3.4 SVD

SVD is an ensemble method which utilizes tools from linear algebra. As suggested by Yin and Schütze [8], we perform SVD on the weighted concatenated embedding vectors. We use the same weights for SVD as W-CONC. For a set of concatenated embeddings of size n, the SVD decomposition is computed as:

$$C = U\Sigma V^T$$

where C is the $|\mathcal{V}| \times d_{conc}$ dimensional matrix corresponding to the concatenated source embeddings. Then the matrix $U_d\Sigma_d$ consisting of the first d columns of U scaled by the first d singular values is chosen as the matrix corresponding to the new meta-embedding. This results in a d dimensional meta-embedding for each protein.

The SVD needs to be calculated over the full corpus (training and testing data), rather than a separate calculation for each type of data. Otherwise the model would encounter different distributions in testing and training data. Thus SVD can be thought of as an unsupervised data preprocessing technique.

3.5 Autoencoders

An autoencoder consists of an encoder network, which generates a vector representation in the latent space given the input features, and a decoder network, which reconstructs an output using the latent features. The structure of the encoder and decoder network usually mirrors each other. Taken together, the two parts of an autoencoder typically minimizes the difference between the input and output. Formally, an autoencoder minimized the loss function

$$\arg\min_{E,D} \mathbb{E}[\Delta(\mathbf{x}, D \circ E(\mathbf{x}))]$$

where $E : \mathbb{R}^n \to \mathbb{R}^p$ is the encoder function, $D : \mathbb{R}^p \to \mathbb{R}^n$ is the decoder function, and Δ is the reconstruction loss that can be customized with regard to different downstream tasks [20].

By representing the input features as a vector in a latent space an autoencoder is essentially a tool to de-noise the input data [21], such that only information essential for its reconstruction is kept. Latent representations could then be used as input features in downstream supervised tasks such as classification and regression. When the dimension of the latent space is smaller than that of the input space, autoencoder is a dimension reduction method [22], but it is not necessarily so, especially when autoencoders are used as an ensemble method.

Autoencoders have wide application in computer vision [23], anomaly detection [24, 25], and natural language processing. When it comes to amino acid sequence specifically, autoencoders has proven

to be a useful tool for predicting the properties of protein such as protein solvent accessibility and contact number [26], predicting the effects of mutations [27, 28, 29], and even to generate novel functional protein variants [30].

However, past research on using autoencoders in protein modeling suffers from several constraints. The biggest of those is the lack of pretrained embeddings. In fact, most research that applies autoencoders to protein function inference feeds one-hot vectors as the input for autoencoders [26, 30]. Given that protein embeddings trained on labeled dataset have achieved SOTA performance in many classification tasks [6], we are interested in this research whether feeding pretrained embeddings to autoencoders could boost the performance of downstream classification tasks.

Moreover, we attempt to use autoencoders as an ensemble learning method that learns a meaningful embedding from protein embeddings trained with various models. Even though there have been studies investigating the possibility of using autoencoders in ensemble learning scenarios, they tend to ensemble several autoencoders rather than one to learn from an ensembled embedding [24]. To our knowledge, Bollegala *et. al.* [9] are the first to use autoencoders for meta-embedding learning, which makes this study the first to apply autoencoders to generate meta-embedding of proteins.

In this paper, we experiment with various choices of latent dimension to assess its effect on the accuracy of the downstream classification task. We will also experiment with different choices of network architecture which is described in the following sub-sections.

3.5.1 Deep Autoencoder (DAE)

We employ a deep autoencoder architecture consisting of two fully connected hidden layers in the encoder and decoder (not counting the latent dimension). The input to the network is the weighted concatenations of all three sources embedding. We use the encoded protein from the latent dimension as input to our classifier and examine the effects of the size of the latent dimension. The hope is that in learning a lower dimensional compression of all three source embeddings, the auto encoder will balance complementary and common features between the embeddings, generating a concise and efficient embedding in the latent space.

3.5.2 Averaged Autoencoded Meta-Embedding (AAEME)

Averaged Autoencoded Meta-Embedding, a method proposed by Bollegala *et. al.*, is a variation on the deep autoencoder which also learns a meta-embedding in some target latent dimension [9]. In the case of the DAE architecture the encoder learns to encode the concatenated source embeddings in some lower dimensional target space and the decoder learns to recreate the concatenated source embedding from that space. By contrast, AAEME trains an encoder for each source embedding which encodes them into some target space of the same dimension. Then, in this space, they are all averaged into a single meta-embedding. Finally using the same averaged meta-embedding separate decoders learn to recreate their respective source embedding.

3.5.3 Convolutional Neural Network Auto Encoder (CNN-AE)

Another type of autoencoder we consider is based on the convolutional neural network architecture. CNNs are designed to process two dimensional inputs, i.e. data that come in the form of multiple arrays [31], so it is a good fit for learning meta embeddings from protein sequences. Notice that in all previous models, we first had to average the embeddings of all amino-acids in a protein sequence before feeding the averaged feature vector to the corresponding model. Although averaging has its merits we may lose important residue level information, which may be essential for inferring key properties of proteins. By contrast, CNN preserves this information and could potentially draw inferences from local spatial and temporal dependencies.

	Method	Fold	Superfamily	Family
Benchmark	TAPE Benchmark	0.260	0.430	0.920
Embeddings	Elmo (E)	0.242	0.433	0.931
	Transformer (T)	0.232	0.398	0.913
	Unirep (U)	0.224	0.337	0.858
Concatenation	CONC (E+T)	0.267	0.435	0.933
	CONC (E+U)	0.257	0.424	0.930
	CONC (T+U)	0.260	0.446	0.952
	CONC (E+T+U)	0.274	**0.465**	**0.954**
	W-CONC (E+T+U)	**0.290**	0.446	0.937
Averaging	AVG	0.240	0.428	0.922
	W-AVG	0.266	0.435	0.941
Dim. Reduction	SVD (d=200)	0.242	0.443	0.927
	DAE (d=200)	0.249	0.419	0.918
	AAEME (d=200)	0.235	0.412	0.915

Table 1: Results on fold, superfamily and family for CONC, AVG, and dimensionality reduction.

CNNs are not without their problems when it comes to processing sequences. One issue is the increased computational cost coming from the additional parameters in the convolutional layers. Another is that the model needs to deal with amino acid sequences of various lengths. One common solution is to use a sliding window [26, 32], but this suffers from dealing with dependencies outside of the window. In our study, we first pre-process the sequences by either padding or trimming such that all sequences have equal length before passing the data to the autoencoders.

4 Results & Discussion

4.1 Concatenation Outperforms Benchmark

When looking at the pairwise CONC combinations every source embedding improved overall performance when combined with another source embedding. The worst performing pair was Unirep and Elmo suggesting that they represent more common features relative to the transformer which represents more complementary features. This could be attributed to the fact that Unirep and Elmo are both based on an LSTM architecture whereas the transformer architecture is unique among the three.

When all three source embeddings are concatenated the results outperform any pairwise combination, and furthermore outperform the best in class TAPE model benchmarked on this data set. Taking the best of CONC and W-CONC we are able to surpass the benchmark by more than 3 percent accuracy across every test set. This is a significant result in that it is a strong indicator that the embedding spaces represented by each model contain a meaningful amount of complementary information. Additionally it shows that (W)-CONC provides a robust way of combining this complementary information.

4.2 Dimensionality Reduction

While concatenation methods had the best overall performance, a clear drawback of the method is that it increases the dimension of the meta-embedding space with each additional source embedding. So as part of our experiments, we investigated the effect of the dimensionality reduction in the meta-embedding on the performance.

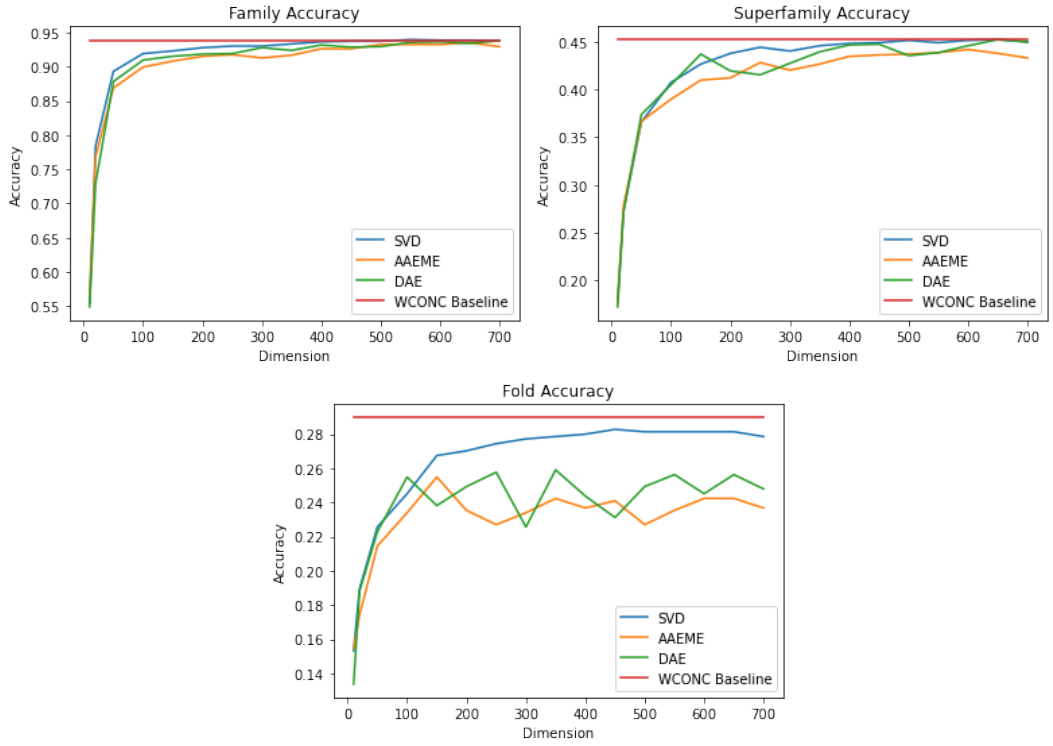

Figure 1: Performance of dimensionality reduction with SVD, AAEME, DAE against the baseline WCONC at family, superfamily and fold levels.

The methods SVD, DAE, and AAEME all combine the source embeddings into a lower dimension space of our choosing. We examine the effect of the size of this dimension and obtain the performances given in Figure 1. For the family-level accuracy, all the methods achieved similar performances and converged towards the baseline given sufficient dimensionality. However, as the relationship between the proteins becomes more distant in superfamily and fold levels, we see that the performances of AAEME and DAE begin to fluctuate, and they are outperformed by SVD, which displays a robust behaviour at all three levels. Generalization to the fold test set requires the least dependence on direct sequence similarity. This suggests that autoencoders may overemphasize direct sequence similarity in their optimization process at the cost of features which are key to understanding correlations between the most remote proteins.

4.3 Averaging Improves Performance at No Cost

From Table 1, we can see that the weighted average (W-AVG) of the embeddings outperforms the benchmark and the individual source embeddings. This is an impressive result as W-AVG improves the performance without increasing the dimensionality, or performing any potentially computationally costly operation like various dimensionality reduction methods described above which may be of great size.

Neural Networks Structure	Latent Dimension	Fold
Convolutional	128	0.248
	512	0.253
	3692	0.226
DAE	128	0.254
	512	0.256
	3692	0.240
AAEME	128	0.254
	512	0.237
	3692	0.224

Table 2: Autoencoder results compared by architecture.

4.4 Autoencoder Performance

We investigated three types of network architectures for the autoencoders, CNNs, DAEs, and AAEMEs. All architectures use ReLU activation functions and a mean squared reconstruction loss.

Table 2 shows that meta-embeddings learned by DAEs achieve slightly higher accuracy in the downstream classification task compared to convolutional autoencoders. Even though residue level information is preserved with the convolutional autoencoders, we suspect that its performance is limited by several computational challenges such as requiring more epochs to converge and lost sequential information when trimming the input sequence. Additionally because convolutional layers deal with significantly higher dimensional inputs it is possible they might perform better on downstream tasks with larger datasets.

We show that autoencoders which target a larger latent dimension are not better. In this case all three types of autoencoders perform better when using a smaller latent dimension when compared to the full dimensional (3692) case. We suspect that this is due to the noisy nature of amino-acid sequences. A larger latent dimension preserves fortuito concurrences of amino-acids while a smaller latent dimension forces the network to retain only the essential features.

While effective to some extent, the autoencoders did not perform as well as some the other methods. They may have been limited by the fact that they were only trained over the remote homology dataset rather than the full unsupervised data set which the source embedding models had access to during pre-training. If the autoencoders had access to this full training corpus, they may have been able to learn more effective encoding schemes for creating meta-embeddings. While we were unable to perform this experiment due to computational constraints, we believe it may be a fruitful path for future work.

5 Conclusions

We have shown experimentally that the embeddings output by UniRep [14], Elmo [17] and Transformer [4] models have complementary features. We found that this complementarity can be exploited via ensemble methods with differing degrees of complexity. In addition, these methods have been found to have their own advantages and disadvantages.

One of the simplest ensemble methods, weighted concatenation (W-CONC), is found to be the best performing one. The weights can easily be selected as hyper-parameters and fine tuned to downstream tasks to take advantage of superior performance by some source embedding. However, a drawback of

this approach is the increased dimensionality, since the dimensionality of the meta-embedding is the sum of all embeddings.

To overcome the increased dimensionality problem, we propose several dimensionality reduction techniques. Among these, SVD with only $d = 200$ dimensions is found to show a robust behaviour and performs close to the full dimensional concatenated meta-embedding. In addition weighted averaging with zero-padding is shown to give robust results while the dimensionality is kept almost constant as new embeddings are introduced.

With respect to autoencoders, network architecture and latent dimension play a role in determining the performance of the learned meta-embeddings. Convolutional autoencoders may not effectively take advantage of the additional resolution in the input data, and all autoencoders achieve higher performance when they target a latent dimension which is smaller than their input dimension.

While we only examine one downstream task, that of remote homology, the clear gains in performance shown here are promising. Given the broad range of evolutionary, structural, and functional information needed to succeed at this task, there is good reason to believe that these performance gains will carry over to other downstream biological tasks. There is clear potential for future work in verifying this intuition as well as seeing the effect of incorporating additional source embeddings. But overall we believe that generating meta-embeddings through methods such as the ones proposed in this paper will be essential in harnessing the full potential of unsupervised protein embedding models.

References

[1] 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.

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

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, pages 9689–9701, 2019.

[5] Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, 2019.

[6] Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803, 2019.

[7] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815, 2014.

[8] Wenpeng Yin and Hinrich Schütze. Learning meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*, 2015.

[9] Danushka Bollegala and Cong Bao. Learning word meta-embeddings by autoencoding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1650–1661, 2018.

[10] Asa Ben-Hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19(suppl_1):i26–i33, 2003.

[11] Leticia Stephan Tavares, Carolina dos Santos Fernandes da Silva, Vinicius Carius Souza, Vânia Lúcia da Silva, Cláudio Galuppo Diniz, and Marcelo De Oliveira Santos. Strategies and molecular tools to fight antimicrobial resistance: resistome, transcriptome, and antimicrobial peptides. *Frontiers in microbiology*, 4:412, 2013.

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

[13] Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic Acids Research*, 48(D1):D376–D382, 11 2019.

[14] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, page 589333, 2019.

[15] Amelia Villegas-Morcillo, Stavros Makrodimitris, Roeland van Ham, Angel M Gomez, Victoria Sanchez, and Marcel Reinders. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *bioRxiv*, 2020.

[16] 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.

[17] 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.

[18] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, Erik L L Sonnhammer, Layla Hirsh, Lisanna Paladin, Damiano Piovesan, Silvio C E Tosatto, and Robert D Finn. The Pfam protein families database in 2019. *Nucleic Acids Research*, 47(D1):D427–D432, 2019.

[19] Joshua Coates and Danushka Bollegala. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. *arXiv preprint arXiv:1804.05262*, 2018.

[20] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[21] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[22] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[23] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[24] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 90–98. SIAM, 2017.

[25] Jinan Fan, Qianru Zhang, Jialei Zhu, Meng Zhang, Zhou Yang, and Hanxiang Cao. Robust deep auto-encoding gaussian process regression for unsupervised anomaly detection. *Neurocomputing*, 376:180–190, 2020.

[26] Lei Deng, Chao Fan, and Zhiwen Zeng. A sparse autoencoder-based deep neural network for protein solvent accessibility and contact number prediction. *BMC bioinformatics*, 18(16):569, 2017.

[27] Sam Sinai, Eric Kelsic, George M Church, and Martin A Nowak. Variational auto-encoding of protein sequences. *arXiv preprint arXiv:1712.03346*, 2017.

[28] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[29] Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.

[30] Alex Hawkins-Hooker, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. Generating functional protein variants with variational autoencoders. *BioRxiv*, 2020.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[32] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.