# Learning Dense Vector Representations for Proteins

Jatin Arora
University of Illinois at Urbana-Champaign
jatin2@illinois.edu

Mohammed El-Kebir
University of Illinois at Urbana-Champaign
melkebir@illinois.edu

## ABSTRACT

Protein sequences and their interactions can be seen as a natural language just like English. This opens up a wide spectrum of deep learning based language modeling techniques which can be applied to understand the inherent semantics of protein molecules. In this paper, we learn dense vector representations for proteins and compare them with popular sequence alignment methods on homology detection task. Results indicate that protein vector representations outperform alignment techniques by a significant margin in both supervised and unsupervised learning paradigms. Utilizing a fusion of pretrained protein vectors from TAPE (BERT) and DeepSF, we improve upon the existing state-of-the-art on SCOP1.75 dataset by 5% in terms of accuracy.

## 1. INTRODUCTION

With the modern day advancements in sequencing technologies, the size of protein databases has been massively increasing. Many of these proteins are massive molecules exhibiting complex primary, secondary and tertiary structures. In order to attain stability, they wrap around and fold-up into complex 3-D structures which largely affects their function and interactions with other proteins. Hence, to understand their function, it becomes very important to analyse their structure and protein folds. Currently, the Pfam[3] database has a protein bank of more than thirty-one million protein domains.

At the same time, with the advances in computing and abundance of unlabeled data, the field of deep learning has flourished. Deep neural network architectures learn dense vector representations of the input (called, embeddings) which encode the semantics of the input as a multi-dimensional vector. These vectors can then be tuned for a variety of downstream tasks.

From a natural language processing perspective, the BERT model [2] is a deep bi-directional transformer architecture trained over a huge text corpus collected from English books and Wikipedia articles. The BERT model take a sentence as input and returns a contextual word vector representation for each work in the sentence. These vectors have been found to capture the semantics of the words well and have been proven to be effective for a variety of natural language processing tasks (GLUE [14]) like question-answering, sentiment analysis, textual entailment etc.

Now, one can't help but notice that there is a one-to-one parallel between protein sequences and natural languages like English. Character-based techniques, once used on the English alphabet, like levenstein distance and sequence alignment [11] approaches have been popular and found to be effective for a variety of protein modeling tasks. Just like the English alphabet has 26 characters, protein sequences are composed of 20 naturally occurring amino acids. English words interact with each other by the rules of grammar and protein sequences interact with each other based on chemical interactions and bonds. Just like each word in English has its own semantics, proteins have their own designated functions. This motivates us to apply modern-day deep learning techniques used for natural language understanding for protein modeling tasks.

This work concentrates on one such protein modeling tasks of homology detection. We learn dense protein vector representations using TAPE (BERT-based)[12] and DeepSF (CNN-based)[7] deep neural architectures and compare it with sequence alignment based methods in both supervised and unsupervised (clustering) learning paradigms. The contributions of this work are three-fold:

1. On the unsupervised clustering task, dense protein vectors are found to be significantly more effective than sequence alignment.
2. TSNE projection analysis reveals that protein embeddings learnt from TAPE (transformer-based architecture) are found to capture protein structure semantics much better than DeepSF (CNN-based architecture).
3. TAPE embeddings when combined with DeepSF embeddings are able to push the state-of-the-art on homology detection task over SCOP1.75 dataset by 5%.

The next section describes the dataset used in all experiments. Then, we formally introduce the remote homology detection task followed by a detailed description of the approach. We then show our experiments and state our obser-


DOI: 10.1145/1235

vations and inferences drawn. We finally conclude with our learnings opening up new directions for future work.

## 2. TASK DESCRIPTION

Remote homology detection task comes under the general category of sequence classification. Given a protein sequence which is represented as a string of amino acid characters, we have to classify it into a class representing a particular protein structure. In this work, we use the SCOP1.75 dataset and focus on two-sets of output classes:

1. **Coarse-Grained**: Protein sequences are mapped to one among 7 coarse-grained structural classes.

2. **Fine-Grained**: Protein sequences are mapped to one among 1195 fine-grained protein-fold classes.

We approach the homology detection task from 2 perspectives:

1. **Supervised Learning**: Given train/validation/test splits from a labeled corpus, we train a machine learning model to classify an input protein into one of the output classes and report multi-class classification accuracy.

2. **Unsupervised Learning**: Given protein sequences, cluster them into a known set of clusters based on fine-grained or coarse-grained tasks defined above and report the adjusted rand index to evaluate clustering quality.

## 3. DATASET

We use SCOP 1.75 dataset[4] for all our experiments. Each sample in the dataset has a protein word which is a sequence of amino-acid characters with its structural class, protein-fold label, family label superfamily label and secondary-structure sequence. The secondary structure sequence has basically one 3-dimensional binary one-hot vector for each amino acid in the protein sequence. The 3-dimensions correspond to the classes (Helix, Strand, Loop). There are a total of 7 structural classes and 1195 uniquely identified protein-fold classes. However, the distribution of proteins into the protein folds is highly uneven with 5% (i.e. 61/1195) folds having $> 50$ proteins, 26% (i.e. 314/1195) folds having 6 to 50 proteins and 69% (820/1195) folds each having $< 5$ proteins making it extremely difficult to train a classifier accurately predicting all the folds, especially the small folds having very few proteins. The proteins in all the 1195 folds have a sequence length ranging from 9 to 1419 and most have length in range 9-600. The train/validation/test splits used for supervised learning tasks are summarized in Table 1.

| | Samples |
|---|---|
| Train | 12312 |
| Validation | 736 |
| Test | 718 |

**Table 1: Dataset Stats**

## 4. APPROACH

As mentioned in the task description, we segregate our approach into unsupervised and supervised learning paradigms. In the next subsections, we explore each of them separately.

### 4.1 Protein Vectors

For dense vector-based approaches we use two deep neural network architectures, namely TAPE and DeepSF.

#### *4.1.1 TAPE*

TAPE is based on a popular deep neural network architecture found to be effective in the natural language processing domain, called BERT [2]. BERT is inherently a transformer[13] model which is able to capture long term dependencies among input sequences well and works well for transfer learning. The TAPE model takes in a protein sequence (character by character), where each character represents an amino acid. It then maps each amino acid into a vector/embedding. These vectors are then conditioned and tuned by passing through a deep neural network architecture which captures the interdependencies among the amino acids in a sequence. Qualitatively, the interdependencies come from protein folding. Protein molecules fold to attain additional stability by developing hydrogen bonds among themselves. The neural network used by TAPE captures these amino-acid interactions and finally outputs a vector representation for each amino acid in a protein. We then take a mean of these amino acid embeddings to get an overall vector representation for the input protein. The TAPE model is trained on Pfam[3] dataset having more than thirty-one million protein domains.

#### *4.1.2 DeepSF*

DeepSF model is a deep convolutional neural network architecture. The input to the model is a 45-dimensional vector for each amino acid which includes its one-hot representation, along with secondary structure information (Helix, Strand, Loop) as described in the dataset section above and position specific scoring matrix (PSSM) features calculated using PSI-BLAST[1]. The model consists of 15 layers including the input layer, having 10 convolution layers. The convolution layer captures the dependence of a particular amino acid with its immediate neighbors and multiple convolution layers are able to capture long distance dependencies of amino acids in a protein sequence. We use the pretrained model (trained on SCOP1.75 training set) and given a protein sequence, take the output of the 14th layer as a feature representation (vector embedding) of the protein. This protein embedding is 300 dimensional.

### 4.2 Unsupervised Learning

In this paradigm, all we have is the actual protein instance. We do not have the output fold-label or structural class and we have to utilize all the intrinsic information/semantics that we can capture from the actual protein sequence without taking any guidance from their output labels. More specifically, we focus on the clustering task where each cluster represents a structural class or protein-fold. Quantitatively, we evaluate clustering using adjusted rand index [8] metric and qualitatively, we plot the *tSNE*[10] projections for the vector-based method.

#### *4.2.1 Global Sequence Alignment*

Given a sequence of proteins, we perform pairwise global sequence alignment [11] using BLOSUM62[6] scoring matrix. Now, these alignment scores depend on the protein sequence lengths and amount of overlap. If the protein sequences are small, then even a very high degree of overlap may assign a small score as compared to a relatively smaller degree of overlap among two long protein sequences. Also, since our overall task is of clustering, we need to convert this alignment matrix into a distance matrix ($D : P \times P \rightarrow \mathbb{R}$) which satisfies the following properties for any given set of protein sequences, $x, y, z \epsilon P$, where $P$ is the set of all proteins, we require,

1. **Non-Negativity**: $D(x, y) \geq 0$
2. **Identity of Indiscernibles**: $D(x, y) = 0$ if and only if $x = y$
3. **Symmetry**: $D(x, y) = D(y, x)$
4. **Triangle Inequality**: $D(x, y) \leq D(x, z) + D(z, y)$

For BLOSUM62 being used for alignment, [9] and [5] show that given sequence alignment matrix, $S : P \times P \rightarrow \mathbb{R}$, we can get a distance matrix $D$ described above using Equation 1. We can qualitatively interpret this distance metric as taking an arithmetic mean of individual proteins involved and subtracting the pairwise alignment score from it. Given $x, y, z \epsilon P$, we have,

$$D(x, y) = S(x, x) + S(y, y) - 2S(x, y) \tag{1}$$

Using this pairwise distance matrix, $D$, we apply K-Medoids algorithm to cluster the protein sequences into $K$ flat clusters where, $K = 1195$ (for fine-grained task, each representing a unique protein-fold) or $K = 7$ (for coarse-grained task, each representing a unique structural class). The process is simply shown in Figure 1.

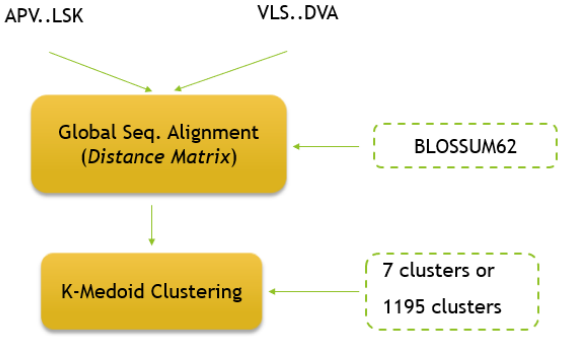


**Figure 1: Global Sequence Alignment Based Clustering**

### 4.2.2 *Dense Protein Vectors*

Given a protein sequence, we pass the sequence character-by-character through a pretrained machine learning model to get a vector, $v \epsilon \mathbb{R}^t$, representing the protein as a vector in $t$ dimensional vector space. We then apply Principal Component Analysis as a dimensionality reduction measure convert vector, $v \epsilon \mathbb{R}^t$ to $v' \epsilon \mathbb{R}^{t'}$, $t' < t$:

$$v' = PCA(v) \tag{2}$$

Then, we apply *tSNE* measure described by [10] to convert $v'$ to 2-D space. Note that, we do not apply *tSNE* directly on the vector $v$ because of computational costs involved in computing the projections. So we have, $v'' \epsilon \mathbb{R}^2$ such that,

$$v'' = tSNE(v') \tag{3}$$

Next, we apply K-Medoids clustering approach as described in the previous section on $v'$ and $v''$ separately, using the following distance measures, given any general vectors, $x, y \epsilon \mathbb{R}^d$:

1. **Euclidean Distance**: $\sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$
2. **Cosine Distance**: $\frac{\sqrt{\sum_{i=1}^{d} x_i y_i}}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$
3. **Symmetric KL Divergence**: $\sum_{i=1}^{d}(p_i \log \frac{p_i}{q_i} + q_i \log \frac{q_i}{p_i})$,

   where $p = Softmax(x)$ and $q = Softmax(y)$

   $Softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}$

Figures 2 and 3 show the clustering process for protein vectors.

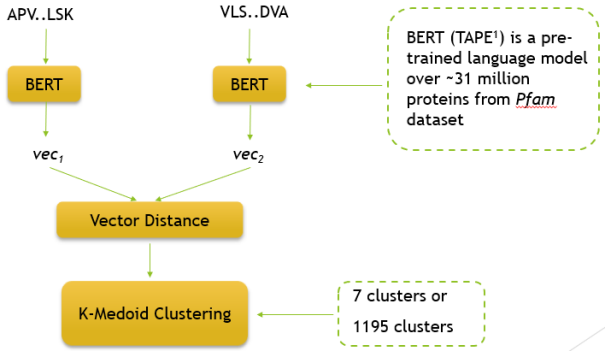


**Figure 2: Vector Based Clustering**

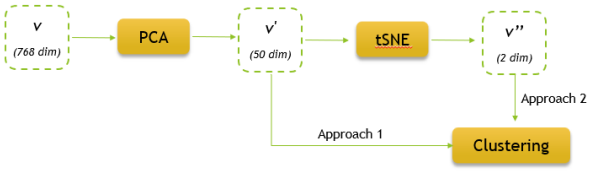


**Figure 3: PCA and tSNE projection procedure**

## 4.3 Supervised Learning

Here, we treat the problem of homology detection as a multi-class classification task and use the pretrained feature vectors from TAPE or DeepSF models or both. We pass these features from a simple classification layer and the output is converted to a probability distribution by passing through a softmax layer. The class with maximum probability is taken to be the predicted output class. The no. of output classes is taken to be 1195 (each class corresponding to a unique protein fold). The model is optimized using Adam optimizer over cross-entropy loss. We evaluate our

approaches by reporting the classification accuracy percentage. The results are summarized in the results section later in the paper.

1. **TAPE-Solo**: Here, we fine-tune the pretrained TAPE (BERT) architecture on our training set and report our results on the test set.
2. **DeepSF-Solo**: Here, we use the existing model provided by the authors, pretrained on our training set and use this to get results on our test set.
3. **TAPE-DeepSF-Max**: This is a toy model through which we analyse that by combining the information captured by TAPE and DeepSF, over the test set what is the upper bound on classification accuracy that can be achieved.
4. **TAPE-DeepSF-Joint**: For each protein sequence, we get the TAPE embeddings and DeepSF embeddings and concatenate them, then train a linear classification layer as described above. The model is shown in Figure 4.

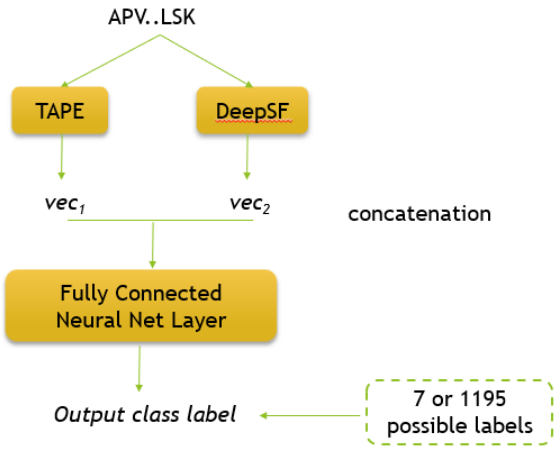


**Figure 4: TAPE-DeepSF-Joint Model Architecture**

5. **TAPE-DeepSF-Ensemble**: This technique does not involve a further learning component. For each input protein, we pass it through *TAPE-Solo* and *DeepSF-Solo* to get corresponding output probability distributions. The overall highest probability class is taken as the predicted class for the protein.
6. **TAPE-DeepSF-Prob**: This develops on top of *TAPE-DeepSF-Ensemble*. The output probability distributions from the two models may not be directly comparable. We study the output probability bias of the *TAPE-Solo* and *DeepSF-Solo* models. For each **correct** prediction done by *TAPE-Solo* or *DeepSF-Solo* models, we note the model confidence (output probability) and plot a histogram. From Figures 5 and 6, we observe that *DeepSF-Solo* model does not give correct classifications with low probability values while *TAPE-Solo* model has many correct classifications with low output probability values. So, as a heuristic, we set that if output probability value from *DeepSF-Solo* $<$ 0.3, then simply take the output class from TAPE-Solo model.

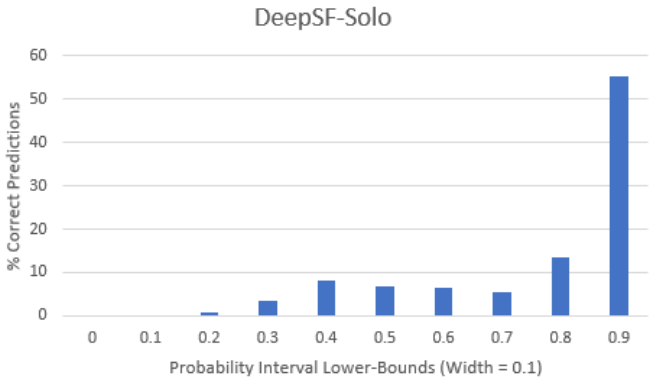


**Figure 5: DeepSF-Solo Output Probability Histogram for Correct Classifications**

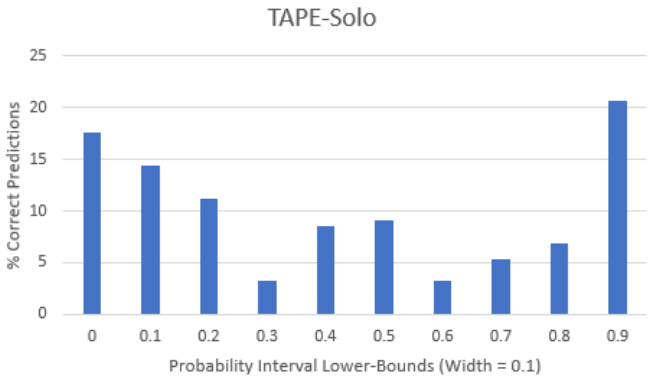


**Figure 6: TAPE-Solo Output Probability Histogram for Correct Classifications**

## 5. EXPERIMENTAL SETUP

- Our supervised learning framework is an extension of the TAPE implementation[1] and makes use of the `transformers`[2] library in `python3`. Each protein sequence irrespective of its amino-acid length is converted to 768-dimensional output vector.
- DeepSF embeddings are generated using the implementation provided by the authors on GitHub[3] using `python2.7`. Each protein sequence is converted to a 300-dimensional output vector.
- Pairwise Global Sequence Alignment is calculated using `BioPython` python package making use of `BLOSUM62` matrix. For KL-Divergence, Euclidean and Cosine distance, we use existing implementations provided by `scipy` library in python.

[1]https://github.com/songlab-cal/tape
[2]https://huggingface.co/transformers/
[3]https://github.com/multicom-toolbox/DeepSF/

- K-Medoid clustering is done using `pyclustering` python package. Initial cluster centres are picked randomly and we the reported results are an average of 5 independent runs.
- `PCA` and *tSNE* is implemented using `scikit-learn` python package and plots are prepared using `matplotlib`. `PCA` is made to convert the input vectors to 50-dimensional output vectors and *tSNE* response is 2-dimensional.
- Training of supervised learning models is done on multiple 8G NVidia GTX GPUs. The models are trained for 30 epochs and best results are reported. Training batch size is set to 64 and gradient accumulation steps is set to 16. All other training parameters are set to default values used by TAPE.
- The pre-processed SCOP 1.75 dataset is downloaded from the TAPE GitHub repository.
- The code for this work is released on GitHub[4].

## 6. RESULTS

- Table 2 summarizes the clustering results on test set (described in dataset section) for coarse-grained classification. Note that for the vector-based distance, we only report the best distance metric in this table. Table 3 presents the same analysis on fine-grained (1195) clusters.

| Approach | Adj. Rand Index |
|---|---|
| Global Seq. | 0.0649 |
| TAPE (Cosine-tSNE) | **0.3312** |
| DeepSF (Euclid-PCA) | 0.0932 |

**Table 2: Coarse Grained Clustering**

| Approach | Adj. Rand Index |
|---|---|
| Global Seq. | 0.0278 |
| TAPE (Euclid-tSNE) | **0.2093** |
| DeepSF (Euclid-tSNE) | 0.1106 |

**Table 3: Fine Grained Clustering**

- Figure 7 shows the comparison of KL-Divergence, Euclidean and Cosine distances on TAPE and DeepSF vectors for coarse-grained 7 clusters. Figure 8 presents a similar analysis for fine-grained clusters.
- Table 4 compares PSI-BLAST (alignment-based baseline, result taken directly from TAPE paper) with vector-based variations described in Section 4.3 for fine-grained protein-fold classification task.

## 7. OBSERVATIONS/INFERENCES

- From our clustering experiments shown in Table 2 and 3, we observe that for both coarse-grained and fine-grained tasks, **vector-based clustering performs much better than sequence-alignment-based distance measure**.

[4]https://github.com/jatinarora2702/protein-embeddings

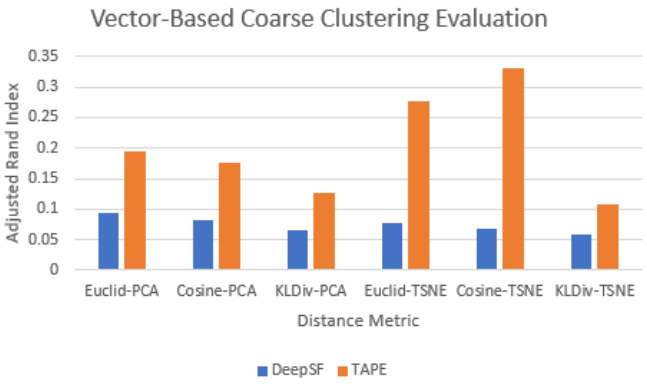


**Figure 7: Vector-Based Coarse Clustering Evaluation**

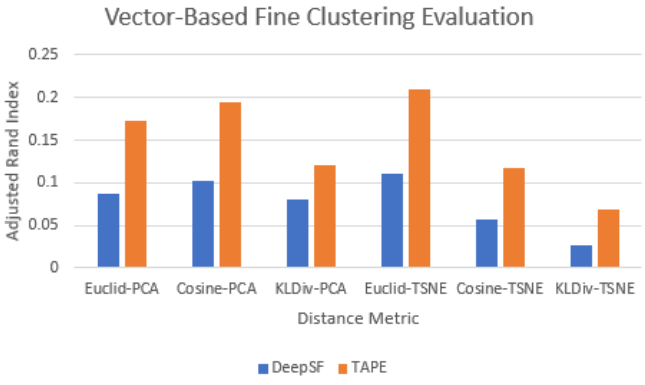


**Figure 8: Vector-Based Fine Clustering Evaluation**

| Approach | Accuracy(%) |
|---|---|
| TAPE-Solo | 26.2 |
| DeepSF-Solo | 40.95 |
| TAPE-DeepSF-Max | **50.42** |
| TAPE-DeepSF-Joint | 20.20 |
| TAPE-DeepSF-Ensemble | 42.90 |
| TAPE-DeepSF-Prob | **43.03** |
| PSI-BLAST | 5.60 |

**Table 4: Fine Grained Classification**

Further, we see that the vectors trained from *TAPE* model capture the protein semantics much better than *DeepSF* model and hence report a significantly higher adjusted rand index.

- From Figures 7 and 8, we again see how *TAPE* vectors significantly outperform *DeepSF* regardless of the distance metric used.
- **Cosine** and **Euclidean** distances are found to perform well with the protein vectors. For coarse-grained clusters, taking the tSNE projections and then applying cosine similarity on 2-D vectors gives a high performance. Similarly, tSNE projections with Euclidean distance give the best results for fine-grained clustering.
- **TAPE-DeepSF-Max** achieves 50.42% overall accuracy which shows that TAPE indeed learns several important structural cues in proteins which the DeepSF model is not able to capture. This sets the upper bound on the maximum accuracy that can be achieved by using these 2 pretrained models.
- By a simple ensemble of the two models as done by **TAPE-DeepSF-Ensemble** and **TAPE-DeepSF-Prob**, we are able to achieve better results than the individual models.
- **TAPE-DeepSF-Joint** model does not perform as good as the individual models and this can be because the model may be getting confused while trying to map the embeddings from the different model spaces to a shared vector space.

In the next subsections we present some additional probing and analysis done in both unsupervised (clustering) and supervised learning paradigms.

## 7.1 Clustering: Qualitative Analysis

We plot the tSNE projections obtained from *TAPE* and *DeepSF* vectors and show the results in figures 9 and 10. We can observe that *TAPE* inherently captures the structural class semantics much better than *DeepSF* vectors and clearly segregates the red, purple, green, yellow, pink classes. Such a distinction is less evident in *DeepSF* results where blue, green, yellow, pink class points are more mixed.

## 7.2 Model Bias Analysis

We probe the *TAPE-Solo* and *DeepSF-Solo* models on the test set and try to look for biases they may have developed based on protein-fold, coarse-grained structural classes, family and superfamily labels of proteins. In the next subsections, we present our analysis.

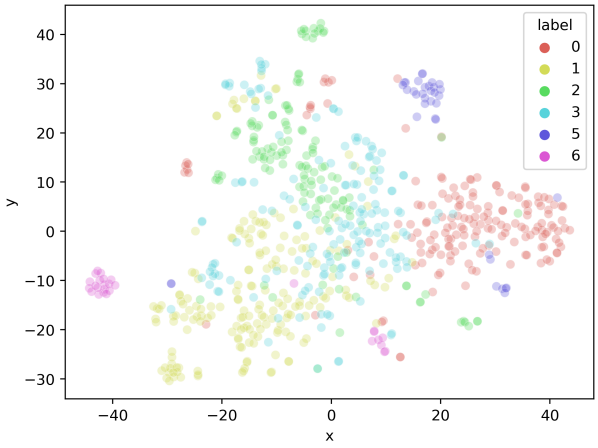


**Figure 9: tSNE projections on TAPE vectors for proteins in Test Set with 7 coarse structural classes**

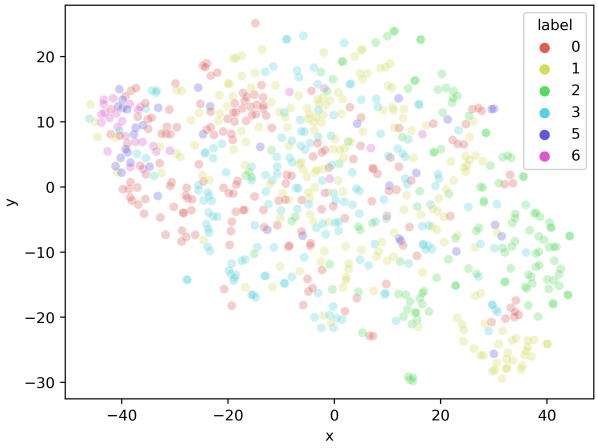


**Figure 10: tSNE projections on DeepSF vectors for proteins in Test Set with 7 coarse structural classes**

### 7.2.1 Protein Fold Bias

For fold-labels 28 and 30, CNN-based model is able to capture the structure well and transformer structure performs badly. On the other hand, for fold-label 454, the reverse is observed and transformer architecture significantly outperforms the CNN model. Table 5 shows some extreme classification cases.

| Protein Fold | DeepSF-Solo | TAPE-Solo |
|---|---|---|
| 73 | 89.2 | 89.2 |
| 80 | **57.1** | 0.0 |
| 454 | 20.0 | **95.0** |
| 28 | **100.0** | 16.7 |
| 71 | 10.0 | 0.0 |

**Table 5: Fold-Level Accuracy(%) Comparison (Test Set)**

### 7.2.2 Structural Class Bias

at the coarse-grained structural class level also, the *TAPE-Solo* and *DeepSF-Solo* models are found to behave differently. More specifically, *TAPE-Solo* which is pretrained on a very large Pfam protein dataset is found to work better than the *DeepSF-Solo* model for underrepresented structural classes. Table 6 compares the classification accuracy of the two models for each coarse-grained structural class.

| Structural Class | # Samples | DeepSF-Solo | TAPE-Solo |
|---|---|---|---|
| 0 | 168 | **32.7** | 23.8 |
| 1 | 190 | **37.9** | 20.5 |
| 2 | 134 | **54.5** | 28.4 |
| 3 | 162 | **44.4** | 21.6 |
| 4 | 0 | 0.0 | 0.0 |
| 5 | 36 | 38.9 | **58.3** |
| 6 | 28 | 28.6 | **53.6** |

**Table 6: Structural Class Level Accuracy(%) Comparison (Test Set)**

### 7.2.3 Protein Family Bias

family-Level biases developed by the two models are shown in Table 7. Proteins belonging to family 1332 and 1334 are better classified by *DeepSF-Solo*, while for 1440 and 1324, *TAPE-Solo* is better. For family 1399 and 1261 both perform badly.

| Structural Class | DeepSF-Solo | TAPE-Solo |
|---|---|---|
| 1332 | **69.2** | 0.0 |
| 1334 | **71.4** | 0.0 |
| 1261 | 12.5 | 0.0 |
| 1399 | 0.0 | 0.0 |
| 1440 | 20.0 | **80.0** |
| 1324 | 0.0 | **100.0** |

**Table 7: Family Level Accuracy(%) Comparison (Test Set)**

## 8. CONCLUSION AND FUTURE WORK

In conclusion, we observe that protein vector representations capture the semantics of proteins well. TAPE model and DeepSF model, both have some overlaps but also have some differences/biases they develop based on the underlying model architecture. Treating proteins as as a language just like our spoken languages has a great potential. Even on the task of homology detection, the current state-of-the-art result is **43%** depicting that there is still lots of scope for improvement.

From our observations, we see that joint training of TAPE and DeepSF model has not been found to be effective. This could be because the model during the training process gets confused while trying to map the two vectors (from TAPE and DeepSF) to the same joint vector space. In future, we would like to probe this further and devise some solutions so that the two models can be jointly trained. This has scope of improving further on the current state-of-the-art.

## 9. ACKNOWLEDGEMENTS

This work came as part of course project for CS466: Introduction to BioInformatics course. I'd like to thank the TAs and our course instructor for teaching us and guiding us throughout the course.

## 10. REFERENCES

[1] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, et al. The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432, 2019.

[4] N. K. Fox, S. E. Brenner, and J.-M. Chandonia. Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures. *Nucleic acids research*, 42(D1):D304–D309, 2014.

[5] E. Halperin, J. Buhler, R. Karp, R. Krauthgamer, and B. Westover. Detecting protein sequence conservation via metric embeddings. *Bioinformatics*, 19(suppl_1):i122–i129, 2003.

[6] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

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

[8] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[9] M. Linial, N. Linial, N. Tishby, and G. Yona. Global self-organization of all known protein sequences reveals inherent biological signatures. *Journal of molecular biology*, 268(2):539–556, 1997.

[10] L. v. d. Maaten and G. Hinton. Visualizing data using

t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[11] W. R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11(3):635–650, 1991.

[12] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[14] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.