Evaluating transfer learning for classification of proteins in bioinformatics

Rosario Vitale and Georgina Stegmayer

Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL, CONICET, Ciudad Universitaria UNL, (3000) Santa Fe, Argentina
{rvitale,gstegmayer}@sinc.unl.edu.ar

Abstract. This study presents a solution to significantly improve protein classification into families or domains using transfer learning. With more than 229 million proteins in UniProtKB, only 0.25% of them have been annotated and classified into over 17,000 possible families. Recently, deep learning (DL) models appeared for this task. However, DL models require large amounts of data for training, and most protein families have just a few examples. To tackle this issue, we propose the application of Transfer Learning (TL) to the classification problem. The TL approach involves self-supervised learning on large and unlabeled datasets to generate a numerical embedding for each data point. This representation learned can then be used with supervised learning on a small, labeled dataset for a specific classification task. The results achieved in this study indicate that using TL for protein families classification can reduce the prediction error by 55% compared to standard methods and by 32% compared to DL models with simple input representations such as one-hot encoding. This study demonstrates that transfer learning is an effective and promising technique to improve protein classification and annotation in large and yet un-annotated databases.

Keywords: Machine learning · Transfer learning · Classification · Protein family.

1 Introduction

The automatic computational annotation of the protein universe is still an unresolved challenge in bioinformatics. Functional annotation of proteins can be considered critical nowadays due to the rhythm of experimental data production [1]. For example, as of April 2023 there are more than 247,000,000 protein entries in the UniProtKB[1]; however, only 569,213 (less than 1%) of them have been reviewed and manually annotated by expert curators. This is a huge breach between sequencing and annotation capabilities. This gap exists due to the high speed of experimental data obtention and, on the opposite, the very low and time-consuming manual curation of results.

[1] https://www.uniprot.org/

The annotation of a protein involves its classification into a family, also named domain. The protein families database (Pfam[2]) is the most widely used repository of protein families. Pfam uses simple sequence similarity by BLAST [2] or manually curated seed alignments of homologous protein regions to generate profiles based on hidden Markov models (HMMs). The resulting models are a representation of each family and can be used to classify novel sequences [3]. Even though this approach is quite successful, there still remain around 25% of proteins that have not been annotated yet because they do not match an HMM profile. In many cases this happens because such a profile cannot be even built since there are only a few examples of a family, which are not sufficient for building a HMM model. Nowadays, the number of sequences in UniProt grows at a much faster rate than its Pfam coverage, introducing novel sequences that may belong to completely new families [4].

As a powerful alternative to profile-HMMs, deep learning (DL) models have recently appeared [5]. Those models are capable of inferring patterns shared across the family sequences, allowing autonomous domain annotation of completely new sequences. This is of high interest for the characterization of sequences that do not resemble anything already studied [6]. However, it is well known that DL techniques rely on large scale data to infer meaningful sequence patterns. This is a strong limitation for protein annotation since many Pfam families comprise just a small number of sequences. In this work we state that this issue can be solved with auto- supervised transfer learning (TL) by transferring Large Language Models (LLMs) representations of protein sequences already learned without requiring annotations from large-scale protein data [7].

In the last 5 years, several LLM for protein representation based on DL appeared, which given the raw sequence of a protein calculate a feature vector that is a unique representation of the protein, named embedding [8]. After that, a predictive model can efficiently learn the features of samples and perform the downstream prediction task by using these representations as input. This way, embedding models perform a process named transfer-learning of knowledge from one task to another [9].

Protein embedding has in fact recently become a new and highly active area of research [10]. A recent review [11] has performed a detailed comparison of protein sequence representation learning methods, explaining each approach and comparing them with an experimental benchmark on several bioinformatics tasks: (i) protein sequence similarity in the embedding space; (ii) protein family classification; and (iii) ontology-based annotation prediction. The review included 12 methods, and the comparative results have shown that ESM and ProtTrans were the best methods for most bioinformatics tasks evaluated.

In this work, we propose that ESM and ProtTrans, used in a TL schema, can effectively improve the protein family classification task in Pfam v32, even when used with simple and classical machine learning (ML) models, such as k-nearest neighbors (kNN) and multilayer perceptron (MLP). The results achieved in this study indicate that using TL for protein families classification can reduce the

[2] https://www.ebi.ac.uk/interpro/entry/pfam/

prediction error by 55% compared to standard HMM methods and by 32% compared to DL models with simple input representations such as one-hot encoding.

2 Transfer learning and protein representation methods

2.1 Transfer learning

Transfer learning (TL) (Fig. 1) is a ML technique where one model is first trained (left part) with a big unlabeled dataset in a self-supervised way, that is, not using annotations of any specific task, but predicting parts of the same data fed as input (e.g. masked small sub-sequences). This step is also named pre-training, and the result is a task-agnostic deep model (also named Large Language Model or LLM) and an output model associated with the pretext task for self-supervised learning, which is then discarded. In a second step (right part), the task-agnostic deep model is frozen and what was learned is "transferred" to another deep architecture in order to train a new task-specific model. Here another model is trained with supervised learning on a small dataset with labeled data for a specific task (e.g. protein family classification). In summary, TL refers to the situation where what has been learned in one setting is exploited to improve generalization in another one [12]. For proteins there are several already available task-agnostic deep models, named protein representation methods, which integrate different types of protein information in a compact representation usually named embeddings.

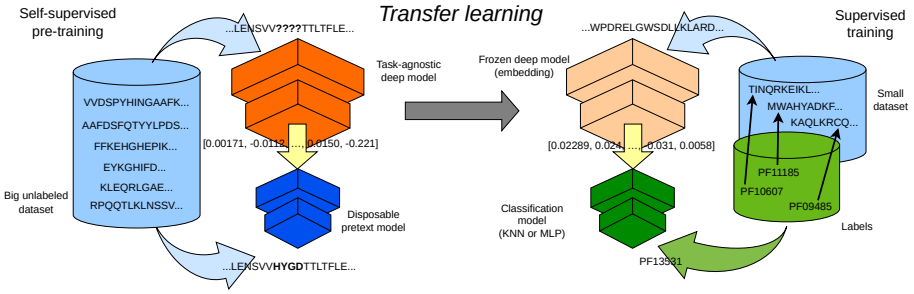

Fig. 1. Transfer learning is a machine learning technique where the knowledge gained by training a model on one general task is transferred to be reused in a second specific task. The first model on the left is trained on a big unlabeled dataset, in a self-supervised way. This process is known as pre-training, and the result is a task-agnostic deep model (orange). Through transfer learning, the first layers are frozen and transferred to another deep architecture (light orange). Then, the last layers of the new model are trained (dark green) with supervised learning on a small dataset with labeled data for a specific task, shown at the right.

2.2 Protein representation methods

Protein representation methods and protein embeddings are becoming known and required by the community. In fact UniProtKB now provides embeddings as part of the standard protein annotations (`https://www.uniprot.org/help/embeddings`). In recent years, a myriad of protein embedding methods has appeared [13]. A recent review [11] has demonstrated that the Evolutionary Scale Modeling (ESM) [6] and ProtTrans [14] are those most outstanding protein embeddings in terms of representational power. Those models were trained using more than 200 million (unaligned) sequences from UniProtKB and are based on Transformers, which have emerged as a powerful general-purpose model architecture for representation learning [15], out-performing deep recurrent and convolutional neural networks. Transformers were originally designed for natural language processing [16], where context within a text is used to predict masked (missing) words. The main hypothesis in this pretext task for self-supervised learning is that the semantics of words can be derived from their contexts. In this work, we have evaluated the two best protein representation models, ESM and ProtTrans, according to [11].

ESM [6]: this model was developed by Facebook Research. It uses a deep Transformer BERT [16] that processes sequences of amino acids as input. BERT was originally designed for Natural Language Processing (NLP) based on unsupervised learning where context within a text is used to predict missing words. The model makes an analogy between syllables in text and amino acids in protein sequences: it learns meaningful encodings for each residue in a self-supervised way, by masking some of the residues in the sequence and trying to predict them. This way, it builds an embedding per residue position that encodes the "meaning" of the residue in that context. Then, the per-residue representation can be collapsed to a per-protein embedding. After this, the learnt representation from UniProtKB, already trained and ready-to-use out of the box, can be "transferred" to be used in a specific downstream task. The model was trained using the masked language modeling objective, where each input sequence is corrupted by replacing a fraction of the amino acids with a special mask token, to predict the missing tokens from the corrupted sequence. ESM was trained on 220 million sequences in the UniProtKB database. In this comparison we have used ESM-1b, ESM-1v and ESM2 because those are the best performing instances of the method according to the authors.

ProtTrans [14]: this model was developed by Google. It was trained using several Transformer models, thus it is based on unsupervised learning. The authors trained two auto-regressive models (Transformer-XL and XLNet) and four auto-encoder models (BERT, Albert, Electra, T5) on data from UniRef50 and the BFD database [17] containing more than 2,000 million proteins and up to 393 billion amino acids. BERT was the first model in NLP which tried to reconstruct corrupted tokens, and is considered the de-facto standard for transfer learning in NLP. Albert reduced BERT's complexity by hard parameter sharing between its attention layers. Electra tries to improve the sampling-efficiency of the pre-training task by training two networks, a generator and a discriminator.

T5 consists of an encoder that projects a source language to an embedding space and a decoder that generates a translation to a target language based on the encoder's embedding. For ProtTrans, single amino acids were considered as input words; each protein sequence in a line represented the equivalent of "sentences"; an empty line was inserted between each protein sequence to indicate the "end of a document". The information learned by the protein learning models were the vector representations from the last hidden state of the Transformer. There are several ProtTrans instances available (ProtBert, ProtAlbert, ProtT5-XL, ProtXL and ProtElectra). We have used the best one as indicated by the authors in its paper.

3 Materials and experimental setup

3.1 Data

For the experimental comparison of models for protein families classification we used publicly available Pfam data as in [5]. Expertly curated sequences from the 17,929 families of Pfam v.32.0 were used to define a benchmark annotation task. Seed sequences from each family were split into challenging train and test sets by clustering them based on sequence similarity. The clustered split provides a benchmark task for annotation of protein sequences with remote homology, that is, sequences in the test set that have low similarity to the ones in the training set. This is useful as an estimation of how well a model will perform with new sequences that are quite different from the ones in the training data. The resulting benchmark has a hard test set of 21,293 sequences and a total of 1,339,083 sequences for training. We have obtained the classical one-hot representation, the ESM embeddings (ESM-1b, ESM-1v and ESM2) and ProtTrans (Bert-BDF and T5-XL) embeddings of all the data.

3.2 Experimental setup

We tested the following ML methods for the Pfam families classification task without TL (one-hot input representation or raw sequence in the case of HMM and BLAST) versus with TL (input embedding):

KNN: the classical k-nearest neighbor (kNN) classifier [18], from scikit learn [19], was used with $k = 1$ and Euclidean distance between embeddings for determining the neighborhoods. Higher number of k was also tried but with worse results.

KNN ensemble: an ensemble of 10 kNN classifiers, with $k = 1$. Each kNN was trained with an independent subset of the training dataset. The ensemble used a majority vote algorithm to determine the output class. Majority vote gets a list of the predictions of each classifier and finds the element that appears more than half the time as the most voted to determine the output class.

MLP: a multi-layer perceptron (MLP) neural network [20] was implemented using scikit learn, with 4 hidden layers of size 500, 100, 100 and 1000, respectively.

The MLP has an identity activation function for each layer, uses backpropagation with a cross entropy loss function and Adam solver for optimization. The model was trained for 25 epochs according to an early-stopping with patience = 10, that is until performance did not change for the last 10 epochs (see Supplementary Material), using partial fit due the large amount of data. Each partial fit call used 1% of the training data.

MLP ensemble: an ensemble of 10 MLP classifiers. Each MLP was trained with an independent subset of the training dataset. The ensemble used a majority vote algorithm, as explained before. The architecture of each MLP and the number of epochs was the same as described for the single MLP.

In the comparisons we have also included the DL models recently proposed for this task, ProtCNN and ProtENN [5]. ProtCNN receives a one-hot coded sequence and learns to automatically extract features to predict family membership. ProtENN is an ensemble of 19 ProtCNNs using a majority vote strategy, where each model was trained with different random parameter initializations.

4 Results

Table 1 shows, for each classification method (in rows), the error rate (calculated as the percentage difference between predicted family and golden standard family for each sequence) in the second column and the corresponding number of errors obtained by each method in the third column. The upper part of the table shows the results for the methods without TL. The first four rows reproduce the results reported in [5] for this same dataset and train/test partition: HMM and BLAST obtained from the raw sequence, and ProtCNN and ProtENN from a one-hot encoding input. We have compared the methods with the error rate because this is how performance is reported in that original comparative work. The fifth row shows the result of a single MLP trained with sequences encoded with one-hot. The bottom part of the table shows the results of the kNN, kNN ensemble, MLP and MLP ensemble classifiers using TL, that is, trained with inputs encoded with different embeddings (ESM-1b, ESM-1v, ESM2, ProtTrans Bert-BDF and ProtTrans T5-XL). The best results without TL and with TL are indicated in bold.

The results in Table 1 indicate that, without TL, the ensemble of DL models achieves the best result, with 12.2% of error, that is 2,590 errors in families classification from a total of 21,293 test sequences. Regarding the use of TL for classification, a single MLP+ProtTrans T5-XL obtained the best result (8,25% and 1,757 errors), followed by kNN+ProtTrans T5-XL (8,63% and 1,838 errors). Curiously, ensembles of models do not seem to improve the prediction of the single classifiers in this application domain. In the case of the MLP, this is due to the fact that each MLP was trained with an independent subset of the training dataset, the same way that the KNN model.

Comparing all the embeddings used in this study, it can be affirmed that using data embedded with ProtTrans T5-XL achieves the best performance in all models. In some cases, such as in comparison with ESM-1b, the error is even

diminished by half when ProtTrans T5-XL is used. If the best model not using TL (ProtENN) and the best model using TL (MLP + ProtTrans T5-XL) are compared, the last one is the one with the best global results, confirming the hypothesis of this study, which stated that protein family classification can be solved with auto-supervised TL by transferring Large Language Models (LLMs) representations of protein sequences already learned without requiring annotations from large-scale protein data.

If the standard method (HMM) is compared against the best performing method without TL (ProtENN) and the best performing method with TL (MLP + ProtTrans T5-XL), it can be stated that the last models improve the classification results by 33% and by an impressive 55%, respectively. It should be noted that the same single MLP without TL obtains a much worse performance, thus it can be clearly seen the improvement that the use of embeddings brings to this task.

Table 1. Pfam families classification results obtained without TL (upper) and with TL (bottom).

Method			**Error rate**	**Number of errors**
Without Transfer Learning	**HMM**		18.10%	3844
	BLAST		35.90%	7639
	ProtCNN		27.60%	5882
	ProtENN		**12.20%**	**2590**
	MLP		41.57%	8852
With Transfer Learning	**ESM-1b**	**KNN**	15.16%	3229
		KNN ensemble	23.39%	4981
		MLP	14.33%	3052
		MLP ensemble	24.96%	5314
	ESM-1v	**KNN**	21.33%	4541
		KNN ensemble	31.08%	6618
		MLP	18.59%	3959
		MLP ensemble	31.06%	6613
	ESM2	**KNN**	15.55%	3311
		KNN ensemble	26.90%	5728
		MLP	11.48%	2444
		MLP ensemble	21.63%	4605
	ProtTrans BERT-BDF	**KNN**	39.49%	8408
		KNN ensemble	55.74%	11868
		MLP	21.63%	4606
		MLP ensemble	35.36%	7529
	ProtTrans T5-XL	**KNN**	**8.63%**	**1838**
		KNN ensemble	15.92%	3390
		MLP	**8.25%**	**1757**
		MLP ensemble	15.92%	3389

Table 2 shows the performance detail of the best model of Table 1, TL+MLP model and the TL+MLP ensemble, this way trained with the full training dataset. The table shows error rate, number of errors, precision, recall and weighted F_1, which was used due to the high imbalance present in training data. Best performance is shown in bold. It can be seen now that the MLP ensemble has indeed obtained the best results, as it could be expected.

Table 2. Performance detail of the best performing method (TL+MLP) for Pfam families classification trained on the full dataset.

TL+MLP Model	Error rate	Number of errors	Prec	Rec	F1
1	9.05%	1926	0.94441	0.90955	0.91702
2	8.64%	1839	0.94353	0.91363	0.91957
3	8.97%	1911	0.94448	0.91025	0.91728
4	9.26%	1972	0.93868	0.90739	0.91220
5	8.88%	1890	0.94274	0.91124	0.91701
TL+MLP Ensemble	**6.48%**	**1379**	**0.95644**	**0.93524**	**0.93923**

5 Conclusions and future work

In this study, we employed transfer learning to enhance the classification of family proteins in a challenging partition of the full Pfam database, with low similarity between train and test sets, with machine learning classifiers. We found that, compared to models without transfer learning, even the most simple machine learning classifiers, such as kNN and MLP, with transfer learning achieved better results, that is the decrease in the error rate was remarkable. Moreover, these simpler classifiers outperformed even more complex models such as deep learning ones. These results suggest that transfer learning is a viable and effective solution for improving protein classification. In fact, instead of building one's own embedder for proteins, it is very useful to reuse all the computation time already spent by the available learnt representations. We conclude that using transfer learning with small sets of annotated sequences, even with very simple classifiers, is easy to implement and provides significant impact on final performance. In future work, it would be interesting to explore the combination of transfer learning with deep learning models to determine their impact on performance.

Acknowledgements. This work was supported by ANPCyT (PICT 2018 #3384) and UNL (CAI+D 2020 115).

References

1. The UniProt Consortium, Zhang, J.: UniProt: the Universal Protein Knowledgebase in 2023. Nucleic Acids Research. 51, D523–D531 (2023). https://doi.org/10.1093/nar/gkac1052
2. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology. 215, 403–410 (1990). https://doi.org/10.1016/S0022-2836(05)80360-2
3. Mistry, J., Coggill, P., Eberhardt, R.Y., Deiana, A., Giansanti, A., Finn, R.D., Bateman, A., Punta, M.: The challenge of increasing Pfam coverage of the human proteome. Database. 2013, (2013). https://doi.org/10.1093/database/bat023
4. Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G.A., Sonnhammer, E.L.L., Tosatto, S.C.E., Paladin, L., Raj, S., Richardson, L.J., Finn, R.D., Bateman, A.: Pfam: The protein families database in 2021. Nucleic Acids Research. 49, D412–D419 (2021). https://doi.org/10.1093/nar/gkaa913
5. Bileschi, M.L., Belanger, D., Bryant, D.H., Sanderson, T., Carter, B., Sculley, D., Bateman, A., DePristo, M.A., Colwell, L.J.: Using deep learning to annotate the protein universe. Nat Biotechnol. 40, 932–937 (2022). https://doi.org/10.1038/s41587-021-01179-w
6. Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C.L., Ma, J., Fergus, R.: Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc. Natl. Acad. Sci. U.S.A. 118, e2016239118 (2021). https://doi.org/10.1073/pnas.2016239118
7. Unsal, S., Atas, H., Albayrak, M., Turhan, K., Acar, A.C., Doğan, T.: Learning functional properties of proteins with language models. Nat Mach Intell. 4, 227–245 (2022). https://doi.org/10.1038/s42256-022-00457-9
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. (2013). https://doi.org/10.48550/ARXIV.1301.3781
9. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J Big Data. 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6
10. Dallago, C., Schütze, K., Heinzinger, M., Olenyi, T., Littmann, M., Lu, A.X., Yang, K.K., Min, S., Yoon, S., Morton, J.T., Rost, B.: Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets. Current Protocols. 1, (2021). https://doi.org/10.1002/cpz1.113
11. Fenoy, E., Edera, A.A., Stegmayer, G.: Transfer learning in proteins: evaluating novel protein learned representations for bioinformatics tasks. Briefings in Bioinformatics. 23, bbac232 (2022). https://doi.org/10.1093/bib/bbac232
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016).
13. Makrodimitris, S., Van Ham, R.C.H.J., Reinders, M.J.T.: Automatic Gene Function Prediction in the 2020's. Genes. 11, 1264 (2020). https://doi.org/10.3390/genes11111264
14. Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., Rost, B.: ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE Trans. Pattern Anal. Mach. Intell. 44, 7112–7127 (2022). https://doi.org/10.1109/TPAMI.2021.3095381
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. (2017). https://doi.org/10.48550/ARXIV.1706.03762

16. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018). https://doi.org/10.48550/ARXIV.1810.04805
17. Steinegger, M., Mirdita, M., Söding, J.: Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. Nat Methods. 16, 603–606 (2019). https://doi.org/10.1038/s41592-019-0437-4
18. Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction. Springer, New York, NY (2009).
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 12, 2825–2830 (2011).
20. Bishop, C.M.: Pattern recognition and machine learning. Springer, New York (2006).

Evaluating transfer learning for classification of proteins in bioinformatics

Rosario Vitale and Georgina Stegmayer

Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL, CONICET, Ciudad Universitaria UNL, (3000) Santa Fe, Argentina
{rvitale,gstegmayer}@sinc.unl.edu.ar

Supplementary Material

Table 1. Error rate obtained by MLP during training.

Ep	Esm1b	Esm1v	Esm2	ProtTrans Bert BFD	ProtTrans T5-XL	Without TL
1	21.52%	29.39%	19.04%	34.58%	14.29%	50.23%
2	18.56%	24.75%	16.93%	29.20%	11.78%	46.88%
3	17.68%	23.11%	14.75%	27.76%	11.40%	45.77%
4	16.85%	21.93%	14.72%	26.02%	10.66%	44.35%
5	16.73%	22.51%	14.20%	24.60%	10.11%	45.42%
6	16.98%	21.11%	13.89%	24.15%	9.47%	44.93%
7	16.46%	21.11%	13.91%	24.23%	9.43%	45.50%
8	15.97%	20.41%	13.04%	24.50%	9.36%	44.76%
9	14.95%	20.44%	12.99%	24.67%	9.04%	44.92%
10	15.55%	20.20%	12.51%	23.88%	9.14%	44.23%
11	14.81%	19.88%	12.91%	23.35%	9.26%	44.08%
12	14.92%	20.18%	12.63%	23.22%	8.99%	43.85%
13	14.97%	19.13%	11.91%	22.39%	8.81%	43.54%
14	14.45%	19.90%	12.99%	23.00%	8.78%	43.46%
15	14.81%	19.36%	12.61%	22.36%	8.56%	43.21%
16	15.12%	19.45%	11.91%	23.10%	8.51%	43.97%
17	15.01%	19.32%	12.50%	21.73%	8.66%	43.39%
18	15.07%	19.57%	12.06%	23.05%	8.25%	43.42%
19	14.69%	19.35%	11.63%	22.05%	8.87%	42.54%
20	15.40%	18.84%	11.48%	22.09%	8.57%	42.79%
21	14.42%	19.13%	12.18%	21.81%	8.56%	41.99%
22	14.98%	19.87%	11.67%	21.63%	8.54%	41.91%
23	14.77%	18.85%	11.84%	21.81%	8.45%	42.18%
24	14.33%	18.59%	11.65%	21.70%	8.75%	41.57%
25	14.34%	19.28%	11.74%	22.07%	8.60%	42.13%

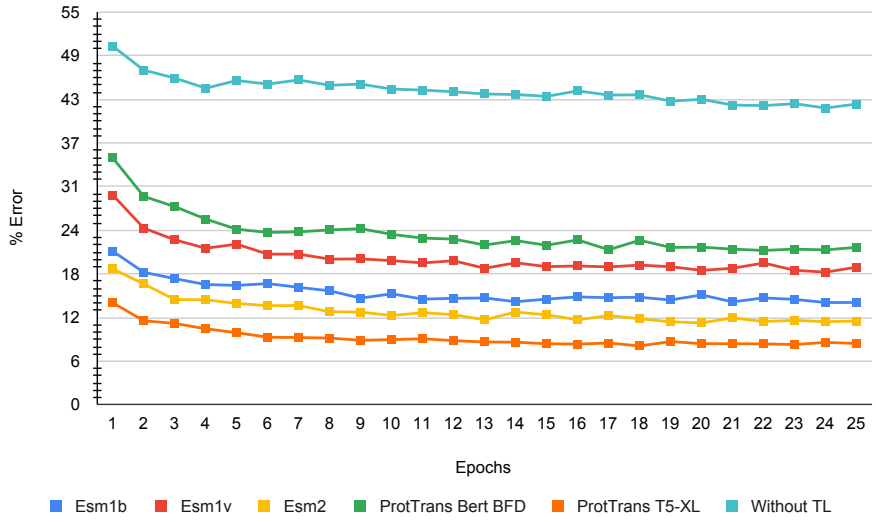

Fig. 1. Percentage of error rate per epoch for the single MLP model trained with different embeddings and without TL