Universal Deep Sequence Models for Protein Classification

Nils Strodthoff*, Patrick Wagner, Markus Wenzel, and Wojciech Samek*

***Abstract*—Inferring the properties of protein from its amino acid sequence is one of the key problems in bioinformatics. Most state-of-the-art approaches for protein classification tasks are tailored to specific classification tasks and rely on handcrafted features such as position-specific-scoring matrices from expensive database searches and show an astonishing performance on different tasks. We argue that a similar level of performance can be reached by leveraging the vast amount of unlabeled protein sequence data available from protein sequence databases using a generic architecture that is not tailored to the specific classification task under consideration. To this end, we put forward *UDSMProt*, a universal deep sequence model that is pretrained on a language modeling task on the Swiss-Prot database and finetuned on various protein classification tasks. For three different tasks, namely enzyme class prediction, gene ontology prediction and remote homology and fold detection, we demonstrate the feasibility of inferring protein properties and reaching state-of-the-art performance from the sequence alone.**

I. Introduction

Inferring protein properties from the underlying sequence of amino acids (primary structure) is a long-standing theme in bioinformatics and is of particular importance in the light of advances in sequencing technology and the vast number of proteins with mostly unknown properties. A rough estimate for this number is given by the size of the sparsely annotated TrEMBL dataset (158M) and should be set into perspective by comparison to the size of well-curated Swiss-Prot [1] dataset (560K) with much more complete annotation of protein properties. There is a large body of literature on methods to infer protein properties, most of which make use of additional handcrafted features in addition to the primary sequence alone [2]–[8]. These include experimentally determined functional annotations as contained in Swiss-Prot as well as features incorporating information from homologous (evolutionary related) proteins which are typically inferred from well-motivated but still heuristic methods such as the basic local alignment search tool (BLAST) [9], that searches a database for proteins that are homologous (evolutionary related) to a given query protein, via multiple sequence alignment.

Handcrafted features based on experimental results are obviously problematic for proteins with incomplete function annotation that have not been studied extensively to this point. Sequence-alignment methods suffer from a poor runtime performance. The complexity of PSI-BLAST [9], a popular variant of a sequence alignment algorithm, scales at least linearly with query and database size, where the latter is growing exponentially [1] in the case of Swiss-Prot and TrEMBL. In addition to unpleasant time complexity, several drawbacks remain for these approaches as they require functional annotations of homologous proteins and are therefore likely to fail for evolutionary distant or insufficiently annotated proteins [10]. Ultimately, self-supervised learning algorithms might be able to implicitly learn even more powerful representations to reach even higher levels of performance, while at the same time offering better scalability properties. In fact, we argue that the trade-off between initial costs for self-supervised pretraining on a large corpus and the variable costs for computing handcrafted features prior to classification will favor our proposal as the number of tasks and thus the number of samples increase.

For these reasons, algorithms operating on sequences of amino acids only are very promising for real-world applications and are currently on the agenda of many research institutions [11]–[13]. In the Machine Learning community, there has been a lot of interest in this direction motivated by the recent advances in Natural Language Processing (NLP) that have demonstrated the large prospects of using language model pretraining tasks to significantly improve the performance on downstream supervised classification tasks– a procedure that is particularly powerful when only small labeled datasets are available for the classification task. Proteomics represents an ideal field of application due to the availability of extremely large unlabeled databases such as Swiss-Prot or TrEMBL and the fact that small datasets are the common situation for most real world applications, where biochemical properties have to be determined experimentally one-by-one.

During the past few months there has been an increased interest in protein classification as possible application area for Deep Learning methods, see e.g. [11]–[15], and in particular NLP methods. In NLP, self-supervised approaches have shown tremendous prospects across a wide variety of tasks [16]–[22], which rely on leveraging implicit knowledge from large unlabeled corpora by pretraining using language modeling or related demasking tasks. This approach goes significantly beyond the wide use of pretrained word embeddings, where only the embedding layer is pretrained whereas the rest of the model is initialized randomly.

Protein classification tasks represent an tempting application domain for such techniques exploiting the analogy of amino acids as words and proteins and their domains as text paragraphs composed of sentences. In this setting, global protein classification tasks, such as enzyme class prediction, are analogous to text classification tasks (e.g. sentiment anal-

All authors are with Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany, e-mail: firstname.lastname@hhi.fraunhofer.de. Corresponding authors marked by *

ysis). Protein annotation tasks, such as secondary structure or phosphorylation site prediction, map to annotation tasks, such as part-of-speech tagging or named entity recognition. While this general analogy has been recognized and exploited already early on [23], self-supervised pretraining is a rather new technique in this field [12], [13]. Existing literature approaches in this direction [12], [13] show significant improvements of models trained using self-supervised pretraining compared to their counterparts trained from scratch on a variety of tasks and demonstrate that models leverage biologically sensible information from pretraining. However, none of them explicitly demonstrates that pretraining can bridge the gap to state-of-the art approaches for these problems that mostly rely on handcrafted features such as position-specific-scoring matrices (PSSM) derived via BLAST.

Our contributions in this paper are the following:

- We put forward a universal protein classification model that relies on finetuning a language model pretrained on Swiss-Prot augmented by custom classification layers (see section III). The success of this approach stresses the fact that many protein classification problems can potentially be addressed with a single task-independent architecture. The major advantage of our approach is that pretraining of a language model needs to be computed only once, hence accelerating future applications by reusing implicit knowledge already learned.
- We demonstrate the competitiveness of our approach for three classical protein classification tasks namely enzyme class (EC) prediction, gene ontology (GO) prediction and remote homology and fold detection (introduced in section II). In section IV we find that our approach performs on par with, or even outperforms state-of-the-art algorithms that typically make use of expensive PSSM features. It is the first time, to the best of our knowledge, that a model operating on the sequence alone was able to reach the level of performance of algorithms operating on PSSM features.
- We systematically investigate the dependence of the pretraining gain on the size of the training dataset for the downstream classification task and illustrate the particular advantages of our approach for small datasets both compared to models trained from scratch and to models that rely on PSSM features. This finding might be a crucial component to increase the quantitative accuracy of protein classification tasks for small datasets.

II. Tasks and Datasets

In this section, we briefly introduce each of the three protein classification tasks considered in this work along with the most prominent literature approaches to these problems before discussing the datasets used for our analysis.

A. Enzyme Class Prediction

Task: Enzyme prediction is a functional prediction task targeted to predict the Enzyme Commission number. The enzyme commission number is a hierarchical numerical classification scheme for enzymes based on the chemical reactions they catalyze. In particular, we consider EC prediction for level 0, i.e. predicting enzyme vs. non-enzyme, and level 1, i.e. predicting one of the six main enzyme classes.

State-of-the-art: A powerful EC classification algorithm of the pre-deep-learning-era was provided by *EzyPred* [2], which owed its success to the design of a hierarchical approach and to appropriate input features which are a combination of the functional (BLAST against a PFAM database) and evolutionary information (PSI-BLAST [9] against the Swiss-Prot database). For hierarchical classification (level 0 to level 2), a simple k-nearest-neighbor classifier (KNN) was trained in order to achieve convincing results. *EzyPred* was superseded by *DEEPre* [7] where deep learning was applied to raw sequence and homology data as input. Instead of training simple classifiers on highly engineered features, they trained feature representation and classification in an end-to-end fashion with a hybrid CNN-LSTM-approach. Recently, *ECPred* [8] also showed competitive results by building an ensemble of well-performing classifier (Subsequence Profile Map with PSSM [24], BLAST-kNN [9] and Pepstats-SVM using peptides statistics [25]). Nevertheless, drawbacks as described in section I remain, i.e. requiring functional annotations of homologous proteins, which is not guaranteed for evolutionary distant or insufficient annotated proteins.

Datasets: EC classification is a commonly used and conceptually simple protein classification task for which a large amount of labeled data is available. For these reasons, we decided to use EC classification as a test bed for detailed investigations of the effects of dataset size, similarity threshold and redundant sequences. The existing dataset provided by *DEEPre* did not meet the requirements of these investigations mainly due to the fact that only representative sequences per cluster instead of full datasets were provided. In addition to the original *DEEPre* dataset, we therefore recreated datasets for similarity threshold 40% and 50% by combining best practices from both [7] and [8]. The corresponding dataset with similarity threshold 40% relying on clusters from a prior CD-HIT [26] run is termed EC40. Compared to *DEEPre* we slightly adapted their dataset generation procedure by clustering Swiss-Prot all at once with CD-HIT, instead of clustering enzymes and non-enzymes separately, as this mimics the way UniRef50 clusters are constructed, albeit for similarity threshold 40% in this case. The dataset using a similarity threshold of 50% by using UniRef50-clusters is termed EC50. For both self-constructed datasets, we do not balance the number of enzymes and non-enzymes since we deem the ratio of enzymes and non-enzymes in Swiss-Prot a more appropriate proxy for the actual ratio observed in nature than artificially restricting the ratio to 0.5. We restrict ourselves for simplicity to proteins with a single EC-label. A detailed description of the dataset generation process and a table summarizing all datasets used for EC prediction are given in appendix.

B. Gene Ontology Prediction

Task: A much more general although closely related problem to enzyme prediction is gene ontology prediction. Gene ontology is an international bioinformatics initiative to unify a part of the vocabulary for representation of proteins attributes. It covers three domains, namely cellular components, molecular functions and biological processes. The nomenclature is organized into hierarchies ranging from coarse to fine-grained attributes. Here, we focus on the domain of molecular function prediction.

State-of-the-art: Similar to enzyme class prediction, the first proposed approaches in this field relied on handcrafted features like functionally discriminating residues (FDR) with PSSM [4] and classification models consisting of an array of Support Vector Machines [5]. Recently, deep learning approaches have raised the bar by using convolutional neural networks [27] and residual neural networks [15].

Dataset: For simplicity and easiest comparability with literature results, we focus on protein annotations belonging to the molecular function (MF) and completely disregard the other two main categories, cellular component and biological process. This is also the considered case of *DeeProtein* [15], with data from the CAFA3 challenge [28], which is a major attempt to establish a suitable evaluation framework for protein function prediction. We use the scripts provided by [15] to construct appropriate training and test sets. Specifically, we train on their Swiss-Prot training set, that was designed to have sequence similarities below 50% compared to the CAFA3 test set, resulting in a training set of 135468 unique sequences. In order to be able to compare directly to literature results [15], [27], we compare performance on a CAFA3 benchmark set comprising 575 sequences for a selection of 539 molecular function GO-terms that were predicted by *DeepGO*.

C. Remote Homology and Fold Detection

Task: Remote homology detection is one of the key problems in computational biology and refers to the classification of proteins into structural and functional classes, which is considered to be a key step for further functional and structural classification tasks. Here we consider remote homology detection in terms of the SCOP database [29], where all proteins are organized in four levels: class, fold, superfamily and family. Proteins in the same superfamily are homologous and proteins in the same superfamily but in different families are considered to be remotely homologous.

State-of-the-art: Remote homology detection has a rich history and we refrain from presenting a detailed discussion of literature approaches to the problem. The interested reader is referred to a recent review article on this topic [30]. We will compare to *ProDec-BLSTM* [6] with a bidirectional recurrent neural network operating on PSSM input features building on earlier work [31]. A classical baseline method is provided by *GPkernel* [3], who apply kernel-methods to sequence motifs.

Datasets: For remote homology detection, we make use of the SCOP 1.67 dataset as prepared by [31], which has become a standard benchmark dataset in the field. Here the problem is framed as a binary classification problem where one has to decide if a given protein is contained in the same superfamily or fold as a given reference protein. The superfamily/fold benchmark is composed of 102/85 separate datasets and we report the mean performance of all models across the whole set.

III. Models and Architectures

A. UDSMProt: Universal Deep Sequence Models for Protein Classification

We aim to address all three different classification problems introduced above within a single architecture that is universal in the sense that only the dimensionality of the output layer has to be adapted to the specific task, which facilitates the adaption of the approach to classification tasks beyond the three exemplary tasks considered in this work. For finetuning on the downstream classification tasks, all embedding weights and LSTM weights are initialized using the same set of weights obtained from language model pretraining. As we will demonstrate, this is a particularly powerful choice for small datasets. The central element that distinguishes approaches based on self-supervised pretraining (as our *UDSMProt*), from the algorithms used predominantly in the literature is the fact that it implicitly learns representations from unlabeled data. This representation can be leveraged for downstream classification task whereas the conventional approaches incorporate side-information in the form of precomputed features. The pretraining step has to be carried out only once, whereas precomputed features have to be recomputed for every single downstream classification dataset.

Our proposed method relies on an *AWD-LSTM* language model [32], which is, at its heart, a 3-layer LSTM augmented by different kinds of dropouts (embedding dropout, input dropout, weight dropout, hidden state dropout, output layer dropout). During language model training, the output layer is tied to the weights of the embedding layer. During finetuning for classification tasks, the output layer is replaced by a concatenation-pooling layer [17] followed by a fully-connected hidden layer and a final output layer. Specific model parameters are listed in Table I and were largely inspired by those from [17].

Table I: AWD-LSTM Parameters

Parameter	Value
Joint parameters	
Number of hidden units	1150
Number of layers	3
Embedding dimension	400
Backpropagation through time (bptt)	70
Gradient clipping	0.25
Weight decay	1e-7
Language-model-specific parameters	
Dropout (p_o,p_h,p_i,p_e,p_w)	0.5 · (0.25,0.1,0.2,0.02,0.15)
Classifier-specific parameters	
Dropout (p_o,p_h,p_i,p_e,p_w)	0.5 · (0.4,0.2,0.6,0.1,0.5)
Max. length (explicit backprop.)	1024
Number of hidden units (head)	50

The training procedure for transfer learning is largely inspired by *ULMFit* [17] and proceeds as follows, see Figure 1 for a schematic illustration: In a first step, we train a language model on Swiss-Prot data. We restrict to using Swiss-Prot, as opposed to the significantly larger TrEMBL database, for pretraining mainly for computational reasons. In particular, we are interested in establishing a solution that is also amenable for users with a smaller computational budget. Besides computational considerations, we also decided in favor of Swiss-

Prot, because we were interested in a direct comparison to BLAST features computed for the same reference database. In a second step, the language model's output layer is replaced by a concat-pooling-layer and two fully connected layers. A potential intermediate step where one finetunes the generic language model on the corpus underlying the classification step, as proposed by [17], did only show an improvement in terms of language model quality but did not result in an improved downstream classification performance. This step was therefore omitted for the results presented below.

When finetuning the classifier, we gradually unfreeze layer group by layer group (four in total) for optimization. We use discriminative learning rates during finetuning reducing the learning rate by a factor of 2 for each layer group compared to the previous. A single model is by construction only able to capture the context in a unidirectional manner, i.e. processing the input in the forward or backward direction. As simplest approach to incorporate both contexts into the final prediction, we train separate forward and backward models both for language modeling as well as for the finetuned classifiers. Then we report also the performance of the ensemble model obtained by averaging the output probabilities of both classifiers.

We use a 1-cycle learning rate schedule [33] during training for 30 epochs during the final finetuning step. Any kind of hyperparameter optimization was performed based on the model performance on a separate validation set, while we report performance on a separate test set. Our way of addressing the specific challenges of the remote homology datasets are described in Section IV-D. In all cases, we use binary/categorical crossentropy as loss function and the AdamW optimizer [34].

B. Convolutional Baseline Models

In order to relate our approach to the state-of-the-art, we will strictly compare our results against an appropriate baseline model. As already mentioned in Section II, the performance of literature approaches has been driven to a large extend by the inclusion of different kinds of handcrafted features rather than innovative model architectures or training procedures. The most beneficial input features throughout a variety of different classification tasks are obviously the position specific scoring matrices (PSSM) based on a multiple sequence alignment computed via position specific iterative BLAST (PSI-BLAST) [9]. BLAST is used to compare query sequences with a given database of already existing sequences, where the result is a list of local alignments solved with heuristics instead of using more time-consuming optimal local alignments with the Smith-Waterman-algorithm. PSI-BLAST is then used to find more distant relatives of a query protein, where a list of closely related proteins is created to get an initial general profile sequence. This profile sequence is used as a new query for the next iteration where a larger list of proteins is found for which again a profile sequence is computed. This process is repeated to a desired number of iterations. In our experiments we used the same parameters as reported in the literature [2], [7], [8], namely three iterations with `e_value` $= 0.001$, where `e_value` relates to the significance threshold for which an alignment is considered as significant.

While the raw sequences from Swiss-Prot contained 26 unique amino acids (20 standard and 6 non-standard amino acids), PSSM features are computed only for the 20 standard amino acids. The raw sequence of length L was then one-hot encoded into an $L \times 26$ matrix which is concatenated with the $L \times 20$ PSSM feature matrix yielding an $L \times 46$ input matrix overall. To make use of the full parallelization capabilities while retaining most information at the same time, we padded the sequences to a maximum length of 1024 residues.

For all following experiments we used a Convolutional Neural Network (CNN) consisting of seven layers, where each convolutional layer was followed by rectified linear unit (ReLU) and max pooling by a factor of 2. The number of filters across layers are: 1024, 512, 512, 512, 256, 256, 256 (with valid padding mode) each with filter size of 3, after the last layer the dimensionality was 6×256. The convolutional stage was followed by flatten/vectorization and three dense layers (512, 256, 128) each followed by dropout (with 25% dropout rate) and finally a softmax layer with six nodes, one for each class.

For all models, we minimized either binary or categorical crossentropy (for level 0 and level 1 respectively) with AdaMax, which is a variant of adaptive moment estimation (Adam) based on the infinity norm [35]. The hyperparameters follow those provided in the paper (learning rate 0.002, exponential decay rates $\beta_1 = 0.9, \beta_2 = 0.999$).

IV. Results

Since our proposed method covers multiple aspects ranging from self-supervised representation learning all the way down to finetuning on several tasks, we aim for a structured and comprehensive evaluation of all the parts mentioned above. Therefore, we first evaluate our proposed method for training a language model with respect to several constraints on the data in Section IV-A.

Afterwards, we compare and evaluate our methods for enzyme prediction in Section IV-B, where we start with an extensive analysis of our methods in Section IV-B1, followed by a comparison to state-of-the-art approaches in the literature in Section IV-B2. In Section IV-B3, we compare the performance with respect to the training size which highlights one of the major benefits, and in Section IV-B4, we evaluated the effect of data leakage issues which are probably present in the previous literature. To demonstrate the universality of our proposed approach, Section IV-C and Section IV-D evaluate our approach for gene ontology prediction and remote homology and fold detection. In all cases the proposed *UDSMProt* approach shows state-of-the-art performance reaching or even outperforming literature approaches for the respective tasks.

A. Language Modeling

We start by evaluating language model performance on proteomics data. This step is important for two reasons: First, language modeling on proteomics data is an interesting field of application for language models outside the classical domain of NLP. The results presented below are supposed to serve as a first benchmark in this direction addressing also the subtle

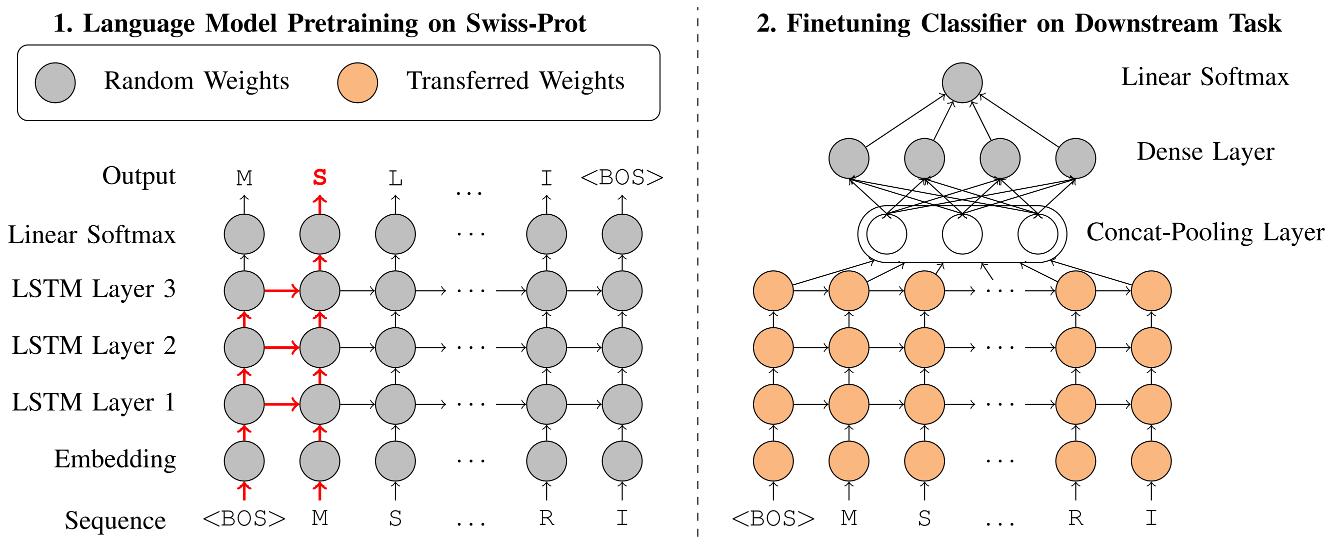

Figure 1: Schematic illustration of the training procedure, here for the amino acid sequence `<BOS>MSLR...RI`. The `<BOS>`-token marks the beginning of the sequence. The red arrows show the context for forward LM for predicting next character (`S`) given sequence `<BOS>M` of length 2. For finetuning on the downstream classification tasks, all embeddings weights and LSTM weights are initialized using the same set of weights obtained from language model pretraining. This has to be contrasted with the use of pretrained (amino acid) embeddings, where just the embedding weights are initialized in a structured way before the downstream finetuning step.

but crucial differences between language modeling evaluation in NLP and in proteomics. Second, as the language model objective is used as pretraining objective for downstream classification tasks, language model performance is instructive to evaluate the implicit knowledge about the structure of proteins that has been captured by the model.

We present language model results using *AWD-LSTM* language models trained on Swiss-Prot using 90% / 5% / 5% splits based on clusters for training, validation and test set, respectively. We train using redundant sequences and evaluate on a reduced dataset with a single representative sequence for each cluster. As performance metrics, we report the natural exponential of the test loss and the prediction accuracy. We tokenize on an amino acid level, the resulting vocabulary comprises the following additional tokens in addition to the 20+2 canonical amino acids: `X` (unknown), `B` (either `D` or `N`), `Z` (either `E` or `Q`), `<BOS>` (marks beginning of new protein). We stress that we do not specifically aim to optimize the language model performance, which could be easily improved using appropriate postprocessing techniques [36], as it only serves as pretraining objective in this context.

At this point a comment on the different training dataset sizes is appropriate. For random splits we disregard any cluster information and distribute samples according to the ratios 90% / 5% / 5%, which obviously results in the largest training dataset. In cases where cluster information is used the splits are performed by clusters, where we sort the clusters by the number of members and distribute them onto the three sets consecutively. Finally, we also report results for a dataset, where the train-test-split used for the language model respects those of a chosen downstream classification task, in this case level 1 EC prediction on the EC50 dataset described below. This construction avoids potential data leakage from using implicit knowledge about test and validation sets that is contained in the language model representations in the actual downstream classification task, see Section IV-B1 for a detailed discussion.

Table II: Language model performance on Swiss-Prot 2017_03 data. The downstream classification task is level 1 EC prediction on the EC50 dataset as described below.

		LM		Down-stream
Split	# Train	Perpl.	Acc.	Acc.
UniRef50	316K	11.37	0.254	0.956
UniRef50 (clean)	322K	11.75	0.244	0.954
Random(64% train)	316K	7.40	0.385	0.955
Random(fwd)	499K	6.88	**0.409**	0.960

The results on language modeling are compiled in Table II. As expected, the best language model is reached on random dataset splits with perplexities of around 7. It is the coverage of the whole dataset in terms of included clusters that is most important for language model performance. This is apparent from the comparison of (1) the language model trained on a random split, where we artificially restricted the training set to 64% of the original training set compared to (2) the UniRef50 runs. In both cases, the size of the training set is comparable but the former (1) reaches a perplexity comparable to that of the model trained on the full dataset with random splits, whereas the latter (2) only reaches a perplexity of around 11.

Even more important than the language model performance itself is, however, in our present context its impact on downstream classification tasks. This is illustrated exemplarily for the EC level 1 classification performance on the EC50 dataset as described below. Table II lists the downstream performance for all language models pretrained on the respective Swiss-Prot version. Interestingly, all fine-tuned classification models show a comparable performance and it is not possible to discriminate between different language models used for pretraining. The same pattern was observed across all considered classification tasks. In particular, it shows that the data leakage from inconsistent splits between pretraining and classification task is presumably small in the context of language modeling. We also experimented with the use of byte-pair-encoding to form subword units [37]. The language model metrics are obviously not comparable for different vocabulary sizes, so the downstream classification performance is the only meaningful metric to compare both approaches in this case. However, we did not see any significant performance improvements compared to the baseline models with single amino acids as tokens. In addition, if one tokenizes using subword units, sequence annotation tasks such as secondary structure prediction are less straightforward to handle.

Key insights from results presented in this section are the following: First, the results demonstrate that language modeling on amino acid sequence data is indeed meaningful and represents a potentially interesting domain of application for language model methods from NLP. Prediction accuracies of 40% (random split) or 25% (UniRef50 clusters) document a solid understanding of the general structure of proteins. However, the section also highlights crucial differences compared to language model evaluation in NLP. Whereas in NLP sequence similarity between train and test sequences is hardly considered, it has crucial implications in proteomics. Here, it is the dataset coverage in terms of clusters rather than the nominal size of the training set, which most crucially determines the language model performance. Second, however, the different language models show no significance differences in terms of downstream classification performance. This iterates a general insight from NLP in the sense that improved language model performance does not necessarily imply improved downstream task performance.

B. *Enzyme Class Prediction*

We start our analysis on downstream classification tasks with EC classification for the reason that it is a conceptually simple task for which a large number of annotated examples is available.

1) Effect of Similarity Threshold and Redundant Sequences: In order to investigate the benefits of the proposed approach in comparison to algorithms relying on alignment features, we based our initial analysis on the self-constructed EC40 and EC50 datasets, which are from the point of view of the dataset creation as fundamental as the datasets considered in the literature. This approach represents a very controlled experimental setup, where one can investigate the effect of the chosen similarity threshold, the impact of redundant sequences during training and potential sources of data leakage during pretraining in a very clean way.

We base our detailed analysis of the proposed method *UDSMProt* compared to a baseline algorithm operating on PSSM features on EC prediction tasks at level 0 (enzyme vs. non-enzyme) and level 1 (main enzyme class). In particular, we aim to investigate the impact of non-redundant sequences when training the baseline classifier and the impact of different similarity thresholds. It is a well-known effect that the difficulty of the classification problem scales inversely with the similarity threshold, as a higher similarity threshold leads to sequences in the test set that are potentially more similar to those seen during training. In the extreme case of a random split, i.e. by disregarding cluster information, the test set performance merely reflects the algorithm's capability to approximate the training set rather than the generalization performance when applied to unseen data. The failure to correctly incorporate the similarity threshold is one of the major pitfalls when applying NLP methods, because it represents a complication that does not exist for natural language in this form. Here we perform level 0 and level 1 prediction on two different datasets, namely EC40 (40%) and EC50 (50% similarity cutoff). Both datasets only differ in the similarity thresholds and the version of the underlying Swiss-Prot databases.

If not noted otherwise, CNN models are trained on representatives as this considerably reduces the computational burden for determining PSSM features and is in line with literature, whereas *UDSMProt* is conventionally trained using the full training set including redundant sequences, whereas the corresponding test and validation sets always contain only non-redundant sequences. For the EC50 dataset non-redundant sequences enlarge the size of the training set from 44,628 sequences to 113,931 sequences for level 1 and 170,535 instead of 86,087 sequences for level 0.

Table III: EC classification accuracy on the self-constructed EC40 and EC50 datasets described in Section IV-B1.

Method	EC40		EC50	
	Level 0	Level 1	Level 0	Level 1
CNN(seq;non-red.)	0.83	0.38	0.88	0.71
CNN(seq)	0.84	0.61	0.92	0.80
CNN(seq+PSSM;non-red.;clean)	0.91	0.84	0.95	0.94
CNN(seq+PSSM;non-red.;leak.)	**0.92**	0.85	0.95	0.95
UDSMProt(fwd;pretr.; non-red.)	0.815	0.786	0.934	0.935
UDSMProt(fwd;from scratch)	0.867	0.793	0.937	0.935
UDSMProt(fwd;pretr.)	0.894	0.840	0.949	0.960
UDSMProt(bwd;pretr.)	0.899	0.853	0.953	0.956
UDSMProt(fwd+bwd;pretr.)	0.909	**0.873**	**0.960**	**0.968**

In Table III, we compare the two classification algorithms *UDSMProt* and CNN that were introduced in Section III in terms of classification accuracy, which is the default metric considered in the literature for this task. There is a noticeable gap in performance across all experiments between CNN(seq; non-red.) and CNN(seq+PSSM; non-red.) which is a strong indication for the power of PSSM features. This gap can be reduced by the use of redundant sequences from training clusters (CNN(seq)) but still remains sizable. Most importantly, however, the gap can be closed by the use of language model pretraining. Disregarding the case of the EC40 dataset at level 0, the best-performing *UDSMProt* consistently outperforms the baseline algorithms that make use of PSSM features. Combining information from both forward and backward context consistently improves over models with unidirectional context. As another observation, pretraining leads to a consistent advantage compared to models trained from scratch that cannot be compensated by increasing the number of training epochs for the models trained from scratch. Finally, Table III illustrates that the *UDSMProt* classification models benefit from redundant training sequences for the downstream classification task. Comparing corresponding results from different similarity thresholds reveals the expected pattern, in the sense that lowering the similarity threshold complicates the classification task as test sequences show smaller overlap with sequences from the training set.

As already mentioned in Section IV-A, pretraining or precomputing features such as PSSMs on the full dataset disregarding the train-test splits for the downstream classification tasks is a possible source of data leakage that can lead to a systematic over-estimation of the model's generalization performance by implicitly leveraging information about the test set during the training phase. Here we investigate the issue also for the case of PSSM features as this issue was, to the best of our knowledge, not addressed in the literature so far. To this end, we compute two sets of PSSM features, one set computed based on the whole Swiss-Prot database (corresponding classification model: CNN(seq+PSSM; non-red.; leakage)) and a separate set based only on cluster members from the training data (corresponding classification model: CNN(seq+PSSM; non-red.; clean)). This is also the reason why all experiments are trained on the same train-test split without performing extensive cross-validation, because this would lead to tremendous computational costs for clean PSSM features since they would have been computed again for each fold. It turns out that the model with PSSM features computed on a consistent train-test-split always performs slightly worse than its counterpart that relies on PSSM features computed on the whole dataset. However, from a practical perspective, the effect of test data leakage remains small. In Section IV-B4, we provide a more extensive evaluation of the effect by varying the size of the training database that is used for calculating PSSM features.

To reiterate the main findings of the experiments carried out in this section, the most crucial observation is that language model pretraining can close the gap in performance between models operating on PSSM features compared to models operating on the sequences alone. The second main observation is that redundant sequences rather than cluster representants have a positive impact on the downstream classification training process, even though one might expect that this information was leveraged during language model training. The most obvious explanations for this observation are inhomogeneous clusters that contain samples with different labels that carry more finegrained information than a single label per cluster representant. Finally, the effect of data leakage from computing PSSM features on the whole dataset disregarding train-test splits turned out to be small, but should be kept in mind for future investigations.

2) *Comparison to Literature Benchmarks:*

In order to relate our proposed approach to state-of-the-art methods in literature, we conducted an experiment on two datasets provided with *ECPred* [8] and *DEEPre* [7]. The main purpose of this analysis is to justify our choice of the CNN baseline algorithm by demonstrating that it performs on par with state-of-the-art algorithms that do not make use of additional side-information for example in the form of PFAM features.

While for *DEEPre* the reported accuracies derived from a 5-fold-cross-validation are straightforward to compute, for *ECPred* however, we reproduced their evaluation scheme, which is tailored to binary one-vs-all-classifiers opposed to multi-class-classifiers. The scheme requires to evaluate binary classifiers that distinguish a particular EC class from other EC classes as well as non-enzymes. Finally, the mean F_1-score is reported across the six datasets. We adopt their evaluation procedure in order to be able to compare directly to their reported results. However, we decided to fit a seven-class categorical classifier (non-enzymes and six main enzyme classes) both on the concatenated training set for all EC classes. In our experiments, the performance of these classifiers was comparable or even better than the corresponding score obtained by training six independent classifiers on EC-class-specific training sets and the procedure is more in line with our approach. We would like to add that this statement applies only to level 0 and level 1, whereas hierarchical classifiers as used by *ECPred* show advantages for sparsely populated cases such as level 2 EC prediction, which is, however, not considered here. At this point we would like to stress that the evaluation procedure is tailored specifically to hierarchical classifiers rather inconvenient to apply for multi-class classifiers.

Table IV: EC classification accuracy on the published *DEEPre* and *ECPred* datasets described in Section IV-B2. *DEEPre* results were evaluated using 5-fold crossvalidation.

	Method	DEEPre (acc.)		ECPred (mean F_1)	
		Level 0	Level 1	Level 0	Level 1
	EzyPred [2]	0.91	0.90	-	-
	DEEPre [7]	0.96	0.95	-	-
	ECPred [8]	-	-	0.96	**0.96**
	DEEPre (seq+PSSM) [7]	0.88	0.82	-	-
	CNN (seq+PSSM)	**0.91**	**0.84**	0.97	0.94
UDSMProt	(fwd; pretr.)	0.855	0.810	0.954	0.928
	(bwd; pretr.)	0.864	0.826	0.968	0.934
	(fwd+bwd; pretr.)	0.874	**0.842**	0.970	0.940
	(fwd; pretr.; red.)	-	-	0.972	0.946
	(bwd; pretr.; red.)	-	-	0.973	0.945
	(fwd+bwd; pretr.; red.)	-	-	**0.976**	0.953

Table IV shows the results of this experiment. For completeness we list also results for *EzyPred*(seq+PSSM+PFAM) and *DEEPre*(seq+PSSM+PFAM), both of which make use of PFAM features in addition to PSSM features. Leaving aside the very unfavorable scaling with the dataset size [11] and possible issues with data leakage due to features computed using the full dataset, we do not consider methods relying on these features as such approach will fail when applied to proteins without functional annotations such that at least one-third of microbial proteins can not be annotated through alignments on given sequences [10]. Also note, that a convolutional model (as our baseline) seemed sufficient when compared to the hybrid model of *DEEPre* (using convolutional layers followed by a recurrent layer (LSTM)) as can been seen in Table IV where our baseline even surpassed the reported performances (91% vs. 88% for level 0 and 84% vs. 82% for level 1). Also for testing on *ECPred* our baseline approach yielded competitive results indicating a well-chosen baseline model. These results justify a posteriori our design choices for the CNN baseline model.

Turning to the performance of the proposed *UDSMProt*, we find a solid prediction performance reaching state-of-the-art performance reported in the literature for algorithms operating on PSSM features. Considering the results of the previous section, the results on the *DEEPre* dataset represent only the lower bound for the achievable performance as it profits considerably from redundant training sequences, which could, however, not be reconstructed in this case. Considering the sizable performance gaps between training on redundant and non-redundant datasets in Table III, it is even more remarkable that the model already reaches state-of-the-art performance when trained on non-redundant sequences. For *ECPred* we report both the performance for training on the original training set as well as the performance on an augmented training set comprising all corresponding Uniref50 cluster members as shown in the three bottom lines in Table IV. In terms of level 0 performance the proposed approach outperforms *ECPred* and it shows competitive performance at level 1.

To summarize, given the results on the benchmark datasets, we see it justified to claim state-of-the-art performance for the CNN baseline model compared to literature approaches disregarding those that incoorporate PFAM features. This allows us to to use this model as baseline model with state-of-the-art performance for the following investigations. The proposed *UDSMProt*-model is very competitive on both literature datasets in particular in the light of missing redundant training sequences for the case of the *DEEPre* dataset.

3) Impact of Dataset Size: In this section, we aim to demonstrate the particular advantages of the proposed *UDSMProt*-approach in the regime of small dataset sizes. To investigate this effect in a clean experimental setup, we conducted an experiment with consecutively decreasing training set sizes, while keeping test and validation sets fixed.

For this experiment, we used the EC50 dataset as described in the appendix with numbers per class as shown in Table VII and trained a level 1 classifier for each split. We compared our proposed approach (AWD-LSTM with pretraining and from scratch) with baseline models (CNN with PSSM features and CNN on redundant sequences only) for seven different training set sizes measured in terms of clusters compared to the number of clusters in the original training set. The hyperparameters were kept fixed to those of the run with full training data. Therefore the results have to be taken with a grain of salt as different architectures might exhibit a different sensitivity to hyperparameter choices upon varying the size of the training dataset, i.e. models trained at smaller dataset sizes might profit differently from additional hyperparameter tuning.

The results from Figure 2 show an interesting pattern: The bidirectional *UDSMProt* model always outperforms the CNN baseline model and most interestingly the gap between the two models increases for small dataset sizes, which suggests the representations learned during language model finetuning represent a more effective baseline for finetuning than using PSSMs as fixed input features. As a second observation, also the gap to the models trained from scratch widens. Reducing the number of training clusters by 50% only leads to a decrease in model performance by 3%, whereas the performance of the model trained from scratch drops by 8%.

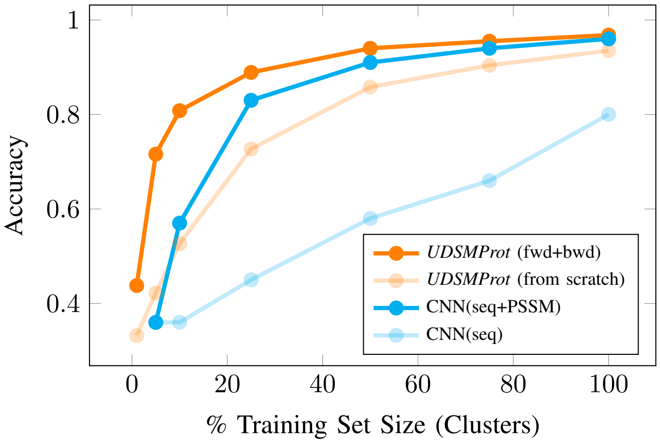

Figure 2: Dependence of the EC classification accuracy (level 1; EC50 dataset) on the size of the training dataset, see Section IV-B3 for a detailed discussion.

To summarize, both above observations represent strong arguments for applications of *UDSMProt* in particular to small datasets. Our results suggest to make language model pretraining a standard procedure in these cases.

4) Impact of BLAST Database Size: In this section, we investigate the data leakage effect discussed in Section IV-A and Section IV-B1 in more detail with focus on PSSM features. The aim is to illustrate that the overstimation of the model generalization performance by computing PSSM features on the whole dataset is not only a theoretical issue but has practical implication for downstream classification performance at least in the cases where the corresponding BLAST database is small.

For pretraining using language modeling this issue did not have significant impact on the downstream performance. This is obviously a desirable property as it would otherwise require to pretrain on large datasets with train-test splits consistent with the downstream task, which defeats the purpose of using pretraining as a universal step unspecific to the choice of the downstream task. However, data leakage is always a potential issue in this context and deserves further research from our perspective to understand the practical implications for different classification tasks.

To highlight the importance of using an appropriate train-test-split also for the database on which PSI-BLAST is performed, we conducted the following experiment, where we compared two sets of features (as already described in Section IV-B1) both trained with the same model and hyperparameters: (1) PSSM features based on the whole Swiss-Prot database (including test sequences) and (2) PSSM features on database consisting only of sequences from the training clusters. While experimenting, we observed that the effect is consistent but barely measurable for large training data, but as the training data (and therefore the BLAST database) get smaller, the effect becomes more apparent. We considered four training sizes: 10%, 30%, 50% and 80% training size, while the test size stayed 20% (the rest of the sequences were neglected for this experiment. For each experiment, we trained a model for level 1 EC prediction as described in Section III-B and shown in Section IV-B1, where we used an early stopping criterion based on the accuracy on a validation set. Here, we used the EC40 dataset as described in Table VII.

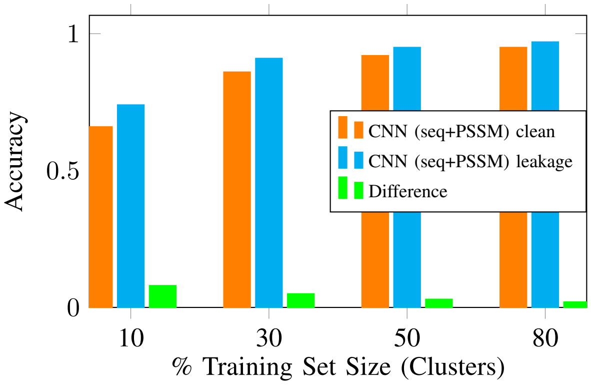

Figure 3: Dependence of EC classification accuracy on the size of the dataset used to computed PSSM features compared to a model exploiting PSSM features computed on the full dataset.

Figure 3 shows the results of this experiment. It can be seen that the effect (difference between clean and leakage) is getting more pronounced when the training dataset gets smaller. This might be a obvious fact from a machine learning point of view, but to the best of our knowledge, this has never been shown in the context of PSSM features for proteomics. From this finding, we can conclude that the results from previous literature are not overestimated strongly. Nevertheless, a fundamental difference remains when dealing with smaller datasets. In the case of small data, considering e.g. 10% of the EC40 training data, the gap in performance between a clean and a leaky procedure is 8% in accuracy, which is far from negligible. In addition, we restricted our analysis to EC classification whereas the effect might very well also depend on the considered downstream task.

C. Gene Ontology Prediction

To stress the universality of the proposed approach, we also present results for gene ontology prediction, which is a multi-label classification task as already introduced in Section II-B.

To ensure maximum comparability, we used the implementation provided by [15] to construct train and test datasets. We grouped duplicate sequences across multiple species with slightly different sets of annotations by building a grand union on those in order to ensure that every possible annotations is considered. We train on their Swiss-Prot training set, that was designed to have sequence similarities below 50% compared to the CAFA3 test set, resulting in a training set of 135468 unique sequences. In order to compare directly to literature results [15], [27], we compare performance on a CAFA3 benchmark set comprising 575 sequences based on a selection of 539 GO-terms that were predicted by DeepGO. Also the evaluation of the final scores from the raw model predictions was carried out using the provided evaluation scripts [15], [27]. The reported metrics are based on those reported in the CAFA3 challenge [27], [38], namely a protein-centric maximum F-measure $F_{\max}$ along with a corresponding average recall and a term-centric AUC. As baseline model, we trained the same model as used in previous experiments, but since the output is 539-dimensional, with slightly broader fully connected layers (i.e. three layers with 512 dimensions each, each followed by rectified linear unit (ReLU) and dropout with rate 0.25). In addition, since this task is framed as a multi-label classification task, we instead optimized piece wise binary crossentropy again with AdaMax. Unless noted otherwise, we trained our models using Swiss-Prot data as prepared by [15] and evaluated on a CAFA3 benchmark set in order to compare to literature results provided by [15], [27].

Table V: Gene ontology (molecular function) prediction performance evaluated on the DeepGO [27] benchmark set, see Section IV-C for details.

Method	AUC	$F_{\max}$	Recall
DeeProtein(Swiss-prot) [15]	0.89	0.51	0.47
DeeProtein(CAFA3) [15]	0.88	0.50	0.42
DeepGO [27]	**0.90**	0.48	0.40
FFPred3 [5]	0.86	0.38	0.40
GoFDR [4]	0.84	0.52	0.36
CNN(seq)	0.89	0.55	0.48
UDSMProt(fwd;from scratch)	0.876	0.574	0.502
UDSMProt(fwd;pretr.)	0.886	0.593	0.531
UDSMProt(bwd;pretr.)	0.884	0.589	0.535
UDSMProt(fwd+bwd;pretr.)	0.886	**0.598**	**0.541**

The results in Table V show that *UDSMProt* model as well as the CNN-baseline outperform the existing approaches in terms of $F_{\max}$-score and recall, while only the pretrained *UDSMProt* reach the performance level of *DeeProtein* in terms of AUC. As for EC prediction, the best-performing *UDSMProt*-model is the forward-backward-ensemble model that exploits bidirectional context. The result presented in this section substantiate our claims regarding the universality of transferring implicit knowledge to task-specific requirements.

D. Remote Homology and Fold Detection

As second demonstration of the universality of our approach, we consider remote homology detection tasks. The corresponding datasets are with a few hundred training examples situated clearly in the small dataset regime investigated in Section IV-B3 and substantiate the claims made in this section in a real-world setting.

The remote homology and fold detection tasks are challenging for two reasons: The datasets are rather small and the task comprises 102 or respectively 85 different datasets that would in principle require a separate set of hyperparameters. To keep the process as simple as possible, we decided to keep a global set of hyperparameters for all datasets of a given task. The procedure is as follows: As no validation is provided for the original datasets, we split the training data into a training and a validation set based on CD-HIT clusters (threshold 0.5). We optimize hyperparameters using the mean AUC for all datasets of a given task measured on the validation set. Most importantly, this involves fixing a (in this case constant) learning rate that is appropriate across all datasets. Using these hyperparameter settings, we perform model selection based on the validation set AUC, i.e. for each individual dataset, we select the model at the epoch with the highest validation AUC. We evaluate the test set AUC for these models and report the mean test set metrics. The standard metrics considered in this context are AUC and AUC50, where the latter corresponds to the (normalized) partial area under the ROC curve integrated up to the first 50 false positives, which allows for a better characterization of the classifier in the domain of small false positive rates that is most relevant for practical applications than the overall discriminative power of the classifier as quantified by AUC.

Table VI: Remote homology and fold detection performance on the SCOP 1.67 benchmark dataset.

Method	Superfamily-level		Fold-level	
	AUC	AUC_{50}	AUC	AUC_{50}
GPkernel [3]	0.902	0.591	0.844	0.514
LSTM_protein [31]	0.942	0.773	0.821	0.571
ProDec-BLSTM [6]	0.969	0.849	-	-
UDSMProt(fwd;from scratch)	0.706	0.552	0.734	0.653
UDSMProt(fwd;pretr.)	0.957	0.880	0.834	0.734
UDSMProt(bwd;pretr.)	0.969	0.912	0.839	0.757
UDSMProt(fwd+bwd;pretr.)	**0.972**	**0.914**	**0.862**	**0.776**

The results are compiled in Table VI. Both for homology and fold detection according to most metrics, the *UDSMProt* model trained from scratch performs worse than the original LSTM model [31]. This is most likely due to the fact that the *UDSMProt* model is considerably larger than the latter model and most datasets are fairly small with a few hundreds training examples per dataset. This deficiency is overcome with the use of language model pretraining, where both unidirectional models perform better than the LSTM baseline model. This observation is in line with the experiments in Section IV-B3 that demonstrates the particular effectiveness of the proposed approach for small datasets. The best-performing model from the literature, *ProDec-BLSTM*, is a bidirectional LSTM operating on sequence as well as PSSM features. Interestingly, reaching its performance in terms of overall AUC required the inclusion of bidirectional context i.e. the forward-backward ensemble model. The proposed method also clearly outperforms classical methods such as *GPkernel* [3] both on the fold and the superfamily level. The excellent results on remote homology and fold detection support our claims on the universality of the approach as well as the particular advantages in the regime of small dataset sizes.

V. Summary and Outlook

In this work, we investigated the prospects of self-supervised pretraining for protein classification tasks leveraging the recent advances in NLP in this direction. Protein classification represents an ideal test bed for NLP methods.

For our methodological investigations, we focused on an enzyme prediction task for which a comparably large set of annotations is available. Here, we demonstrate that self-supervised pretraining, in our case language model pretraining for an AWD-LSTM model as in ULMFit [17], can indeed close the gap to state-of-the-art algorithms that rely on BLAST features obtained from database searches that scale unfavorably with dataset size, or in some cases even outperforms them. Self-supervised approaches show particular advantages for small datasets, which is the generic situation in proteomics. This was investigated in a controlled setup by artificially reducing the size of the training dataset. To substantiate our claims about the universality of the approach in the sense of transferring implicit knowledge to task-specific requirements, we also presented results for gene ontology prediction and remote homology detection outperforming existing approaches that reached state-of-the-art performance in the case of GO prediction or even outperformed existing approaches as for remote homology detection.

Differently from typical NLP tasks, the dataset creation and the evaluation procedure has to be carried out with particular care, as small differences in the procedure can have large impact on the difficulty of the classification problem. This applies in particular to a well-defined way of handling the similarity threshold, i.e. dealing with homologous sequences that differ only by a few amino acids when splitting into train and test sets. These factors urge for the creation of appropriate benchmark datasets that convert raw data from an exemplary subset of the many existing protein classification tasks into benchmark datasets in a transparent manner that allow for a rigorous testing of Machine Learning algorithms in this setting.

Given the insights gained from the three classification tasks, we can draw the following general conclusions for generic protein classification tasks:

1) Considering the fact that both the baseline CNN as well as *UDSMProt* were able to reach state-of-the-art results suggests that problem-specific architectures are less important than the training procedure (*UDSMProt*) or the input features (PSSM), at least for models that are powerful enough. This allows to design task-independent, universal classification algorithms that can be applied without much manual intervention to unseen classification tasks.
2) Bidirectional contexts are important, which is reflected by the fact that in all cases forward-backward-ensemble reached the best performance and in most cases improved the performance of unidirectional models considerably. Ensembling forward and backward models is in fact the simplest – although at the same time a quite ineffective – way of capturing bidirectional context as there is no possibility for interactions of forward-looking and backward-looking parts below the final classification layer. From our perspective, this represents an opportunity for approaches such as BERT [19] or XLNet [22], which are able to capture the bidirectional context directly. This might be particularly important for more complicated protein classification tasks that go beyond the prediction of a single global label such

as sequence annotation tasks like secondary structure or phosphorylation site prediction.

3) Redundant sequences are a valuable source of information also for downstream classification tasks. This fact is in tension with the standard practice in bioinformatics, where in many cases only representants without the corresponding cluster assignments are presented. To ensure comparability, benchmarking datasets should always include full information to reproduce the cluster assignments used during dataset creation, which would allow at least retrospectively to reconstruct the complete dataset from a given set of representants.
4) Data leakage arising from inconsistent train-test splits between pretraining and classification e.g. by precomputing features on the full Swiss-Prot database without excluding downstream test clusters is a possible source of systematic overestimation of the model's generalization performance. From our experiments on EC prediction tasks, its effect was found to be small in particular for large pretraining datasets such as Swiss-Prot, but it should be kept in mind for future investigations.

Leveraging large amount of unlabeled data in the form of large, in parts very well-curated protein databases by the use of modern NLP methods, represents a new paradigm in the domain of proteomics. It will be interesting to see how this process continues with the rapidly evolving algorithmic advances in the field of NLP, considering in particular approaches like BERT [19], see also [12] for first applications in the domain of proteomics, and XLNet [22]. This is particularly relevant for more complicated protein classification tasks that go beyond the prediction of a single global label, such as sequence annotation tasks like secondary structure or phosphorylation site prediction, which even more require bidirectional contexts that cannot be captured by traditional language models. Apart from the huge prospects in terms of quantitative prediction performance, the recent advances in the field of explainable AI research open exciting new avenues for deeper insights into the inner structure of proteins themselves, see also [15] for first applications in this direction.

ACKNOWLEDGEMENTS

The authors thank J. Vielhaben for discussions and work on related topics. This work was supported by the Bundesministerium für Bildung und Forschung (BMBF) through the Berlin Big Data Center under Grant 01IS14013A and the Berlin Center for Machine Learning under Grant 01IS180371. *UDSMProt* was implemented using Pytorch [39] and fast.ai [40] and CNN baseline models were implemented in Keras [41].

REFERENCES

[1] The UniProt Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, no. D1, pp. D506–D515, 11 2018. [Online]. Available: https://doi.org/10.1093/nar/gky1049

[2] H.-B. Shen and K.-C. Chou, "EzyPred: a top–down approach for predicting enzyme functional classes and subclasses," *Biochemical and biophysical research communications*, vol. 364, no. 1, pp. 53–59, 2007.

[3] T. Håndstad, A. J. Hestnes, and P. Sætrom, "Motif kernel generated by genetic programming improves remote homology and fold detection," *BMC Bioinformatics*, vol. 8, no. 1, p. 23, Jan 2007. [Online]. Available: https://doi.org/10.1186/1471-2105-8-23

[4] Q. Gong, W. Ning, and W. Tian, "GoFDR: a sequence alignment based method for predicting protein functions," *Methods*, vol. 93, pp. 3–14, 2016.

[5] D. Cozzetto, F. Minneci, H. Currant, and D. T. Jones, "FFPred 3: feature-based function prediction for all Gene Ontology domains," *Scientific reports*, vol. 6, p. 31865, 2016.

[6] S. Li, J. Chen, and B. Liu, "Protein remote homology detection based on bidirectional long short-term memory," *BMC Bioinformatics*, vol. 18, no. 1, p. 443, Oct 2017. [Online]. Available: https://doi.org/10.1186/s12859-017-1842-2

[7] Y. Li, S. Wang, R. Umarov, B. Xie, M. Fan, L. Li, and X. Gao, "DEEPre: sequence-based enzyme EC number prediction by deep learning," *Bioinformatics*, vol. 34, no. 5, pp. 760–769, 2018. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btx680

[8] A. Dalkiran, A. S. Rifaioglu, M. J. Martin, R. Cetin-Atalay, V. Atalay, and T. Doğan, "ECPred: a tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature," *BMC Bioinformatics*, vol. 19, no. 1, p. 334, Sep 2018. [Online]. Available: https://doi.org/10.1186/s12859-018-2368-y

[9] T. Madden, "The BLAST sequence analysis tool," in *The NCBI Handbook [Internet]. 2nd edition*. National Center for Biotechnology Information (US), 2013.

[10] M. N. Price, K. M. Wetmore, R. J. Waters, M. Callaghan, J. Ray, H. Liu, J. V. Kuehl, R. A. Melnyk, J. S. Lamson, Y. Suh *et al.*, "Mutant phenotypes for thousands of bacterial genes of unknown function," *Nature*, vol. 557, no. 7706, p. 503, 2018.

[11] M. L. Bileschi, D. Belanger, D. Bryant, T. Sanderson, B. Carter, D. Sculley, M. A. DePristo, and L. J. Colwell, "Using Deep Learning to Annotate the Protein Universe," *bioRxiv*, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/05/06/626507

[12] A. Rives, S. Goyal, J. Meier, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences," *bioRxiv*, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/04/29/622803

[13] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating Protein Transfer Learning with TAPE," *bioRxiv*, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/06/20/676825

[14] M. AlQuraishi, "AlphaFold at CASP13," *Bioinformatics*, 05 2019. [Online]. Available: https://doi.org/10.1093/bioinformatics/btz422

[15] J. U. zu Belzen, T. Bürgel, S. Holderbach, F. Bubeck, L. Adam, C. Gandor, M. Klein, J. Mathony, P. Pfuderer, L. Platz *et al.*, "Leveraging implicit knowledge in neural networks for functional dissection and engineering of proteins," *Nature Machine Intelligence*, vol. 1, no. 5, p. 225, 2019.

[16] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[17] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2018, pp. 328–339.

[18] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *Technical report, OpenAI Blog*, 2018.

[19] 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. [Online]. Available: https://www.arxiv.org/abs/1810.04805

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *Technical report, OpenAI Blog*, 2019.

[21] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MASS: Masked Sequence to Sequence Pre-training for Language Generation," *arXiv preprint arXiv:1905.02450*, 2019. [Online]. Available: https://www.arxiv.org/abs/1905.02450

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," 2019. [Online]. Available: https://www.arxiv.org/abs/1906.08237

[23] E. Asgari and M. R. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics," *PloS one*, vol. 10, no. 11, p. e0141287, 2015.

[24] O. S. Sarac, Ö. Gürsoy-Yüzügüllü, R. Cetin-Atalay, and V. Atalay, "Subsequence-based feature map for protein function classification," *Computational biology and chemistry*, vol. 32, no. 2, pp. 122–130, 2008.

[25] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: the European molecular biology open software suite," 2000.

[26] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, "CD-HIT: accelerated for clustering the next-generation sequencing data," *Bioinformatics*, vol. 28, no. 23, pp. 3150–3152, 2012.

[27] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2017.

[28] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur *et al.*, "A large-scale evaluation of computational protein function prediction," *Nature methods*, vol. 10, no. 3, p. 221, 2013.

[29] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures," *Journal of molecular biology*, vol. 247, no. 4, pp. 536–540, 1995.

[30] J. Chen, M. Guo, X. Wang, and B. Liu, "A comprehensive review and comparison of different computational methods for protein remote homology detection," *Briefings in bioinformatics*, vol. 19, no. 2, pp. 231–244, 2016.

[31] S. Hochreiter, M. Heusel, and K. Obermayer, "Fast model-based protein homology detection without alignment," *Bioinformatics*, vol. 23, no. 14, pp. 1728–1736, may 2007. [Online]. Available: https://doi.org/10.1093/bioinformatics/btm247

[32] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *arXiv preprint arXiv:1708.02182*, 2017. [Online]. Available: https://www.arxiv.org/abs/1708.02182

[33] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018. [Online]. Available: https://www.arxiv.org/abs/1803.09820

[34] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," *ICLR*, 2019.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[36] E. Grave, A. Joulin, and N. Usunier, "Improving neural language models with a continuous cache," *arXiv preprint arXiv:1612.04426*, 2016. [Online]. Available: https://www.arxiv.org/abs/1612.04426

[37] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015. [Online]. Available: https://www.arxiv.org/abs/1508.07909

[38] W. T. Clark and P. Radivojac, "Information-theoretic evaluation of predicted ontological annotations," *Bioinformatics*, vol. 29, no. 13, pp. i53–i61, 06 2013. [Online]. Available: https://doi.org/10.1093/bioinformatics/btt228

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *31st Conference on Neural Information Processing Systems (NIPS) Workshop Autodiff*, 2017.

[40] J. P. Howard *et al.*, "fast.ai," http://fast.ai, 2018.

[41] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

Appendix A
Enzyme Class Prediction Dataset Construction

The EC50 and EC40 EC classification datasets are constructed according to the following procedure:

1) Acquire Swiss-Prot (2016_08 for EC40 and 2017_03 for EC50)
2) Cluster Swiss-Prot with CD-HIT (40% similarity cutoff) for EC40 or acquire UniRef50 (2017_03) for EC50 and apply it to Swiss-Prot
3) Remove non-enzymes which have not enough experimental evidence in order to avoid misleading false negatives (annotated evidence is greater or equal to 4)
4) Remove proteins which are annotated as fragments
5) Remove enzymes with multiple enzymatic annotations in order to obtain a single-label classification problem
6) Filter proteins by length to include only proteins with more than 50 and less than 5000 amino acids
7) Split by clusters into 80% training and 10% validation and 10% test set.
8) For test and validation set, we select only representatives (from CD-HIT or UniRef50 clustering) or alternatively the first member of the cluster in the case where the representative was excluded by filtering rules. Optionally a similar reduction is applied to the training set to obtain a set of non-redundant sequences.
9) Use the predefined cluster assignments of the previous step and similarly distribute all remaining Swiss-Prot clusters onto training, validation and test dataset according to split ratios 90%, 5% and 5%, respectively, to obtain a clean train-test-split on the whole Swiss-Prot dataset consistent with chosen clusters for the downstream classification task. Again, we keep only representatives for validation and test set.

The last step is important for suitable database creation for PSI-BLAST for the following experiments in Section IV-B4 and Section IV-B1. One thing can already be anticipated, as a result, for significantly more sequences from the test dataset PSI-BLAST returned empty queries resulting in less informative features for the test set.

Table VII summarizes all datasets used for our experiments, namely the original data from [7] and [8] in the first two columns, and our replica in the last two columns. Note that we did not balance the datasets according to the number of enzymes and non-enzymes.

Table VII: EC Prediction Datasets Overview

EC class	DEEPre [7]	ECPred [8]	EC40	EC50
Oxidoreductase	3341	8242	3781	8244
Transferase	8517	20133	9137	20139
Hydrolase	5917	16018	7987	16020
Lyase	1532	3475	1401	3475
Isomerase	1193	2883	1160	2834
Ligase	1666	4429	2165	4429
Enzymes	22166	55180	25631	55141
Non-Enzymes	22142	25333	32387	49799
Total	44336	80513	58018	104940