ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago,
Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer,
Martin Steinegger, Debsindhu Bhowmik and Burkhard Rost

Abstract—Computational biology and bioinformatics provide vast data gold-mines from protein sequences, ideal for Language Models taken from NLP. These LMs reach for new prediction frontiers at low inference costs. Here, we trained two auto-regressive models (Transformer-XL, XLNet) and four auto-encoder models (BERT, Albert, Electra, T5) on data from UniRef and BFD containing up to 393 billion amino acids. The LMs were trained on the Summit supercomputer using 5616 GPUs and TPU Pod up-to 1024 cores. Dimensionality reduction revealed that the raw protein LM-*embeddings* from unlabeled data captured some biophysical features of protein sequences. We validated the advantage of using the *embeddings* as exclusive input for several subsequent tasks. The first was a per-residue prediction of protein secondary structure (3-state accuracy Q3=81%-87%); the second were per-protein predictions of protein sub-cellular localization (ten-state accuracy: Q10=81%) and membrane vs. water-soluble (2-state accuracy Q2=91%). For the per-residue predictions the transfer of the most informative embeddings (ProtT5) for the first time outperformed the state-of-the-art without using evolutionary information thereby bypassing expensive database searches. Taken together, the results implied that protein LMs learned some of the *grammar* of the *language of life*. To facilitate future work, we released our models at https://github.com/agemagician/ProtTrans.

Index Terms—Computational Biology, High Performance Computing, Machine Learning, Language Modeling, Deep Learning

1 Introduction

Deep Learning (DL) has recently been advancing hand-in-hand with *High-Performance Computing* (HPC) to achieve new scientific breakthroughs in both fields. More powerful supercomputers [1], [2] and advanced libraries [3], [4], [5], [6], [7] enable the training of more complex models on bigger data sets using advanced processing units (incl. Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs)).

Through contextualized Language Models (LMs) [8], [9], Natural Language Processing (NLP) has been benefiting substantially from advances in HPC. In particular *Transformers* [10] have reached state-of-the-art (*SOA*) performance for several tasks [11], [12]. Limitations in annotations do not constrain LMs: the self-supervised training exclusively relies upon the sequential order of the input, e.g., by reconstructing corrupted tokens given the surrounding sequence. After training, we can extract some information learned by the LMs, referred to as *embeddings*. *Transfer-learning* refers to the idea of using such embeddings as input for subsequently trained supervised models. These two steps outsource the computationally demanding LM pre-training to the HPC infrastructure, leaving the computationally less demanding inference to commodity hardware.

Proteins are the machinery of life, built from 20 different basic chemical building blocks (called *amino acids*). Like beads, those amino acids are strung up in one-dimensional (**1D**) sequences (the beads are referred to as *residues* once connected). These 1D sequences adopt unique three-dimensional (**3D**) shapes (referred to as protein *3D structure*) [13], and the 3D structures perform specific function(s) (often simplified as *sequence determines structure determines function*). We know many orders of magnitude more protein amino acid sequences than experimental protein structures (*sequence-structure gap*) [14]. Knowing protein structure helps to understand function. Closing, more generally, the sequence-annotation gap through prediction methods based on artificial intelligence (AI) is one of the crucial challenges for computational biology and bioinformatics. Tapping into the vast wealth of unlabeled data through transfer-learning may become crucial to bridging these gaps.

Top prediction methods in computational biology [15], [16], [17], [18], [19], [20] combine machine learning (ML) and *evolutionary information* (EI), first established as the winning strategy to predict protein secondary structure [21], [22] in two steps. First, search for a family of related proteins summarized as multiple sequence alignment (MSA) and extract the evolutionary information contained in this align-

- *A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi and B. Rost affiliated with TUM (Technical University of Munich) Department of Informatics, Bioinformatics & Computational Biology - i12, Boltzmannstr. 3, 85748 Garching/Munich, Germany.*
- *Y. Wang affiliated with Med AI Technology (Wu Xi) Ltd. , Ma Shan, Mei Liang Road, 88, 2nd floor (west), Bin Hu District, Wu Xi, Jiang Su Province, China.*
- *L. Jones affiliated with Google AI, Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA.*
- *T. Gibbs, T. Feher and C. Angerer affiliated with NVIDIA, 2788 San Tomas Expy, Santa Clara, CA 95051, Vereinigte Staaten, USA.*
- *M. Steinegger affiliated with School of Biological Sciences, Seoul National University, Seoul, 08826, South Korea.*
- *D. Bhowmik affiliated with Oak Ridge National Laboratory (ORNL), 1 Bethel Valley Rd, Oak Ridge, TN 37830, Vereinigte Staaten.*
- *A. Elnaggar and M. Heinzinger contributed equally to this work.*
- *Corresponding author: ahmed.elnaggar [at] tum.de, tel: +49-289-17-811 (email rost: assistant [@] rostlab.org)*
- *The official GitHub repository: https://github.com/agemagician/ProtTrans*

ment. Second, feed the EI into the ML through supervised learning implicit structural or functional constraints. When predicting for proteins without experimental annotations, such methods only use experimental information implicitly captured in the trained model. Since all other information originates from the knowledge of sequences, such methods need no additional information as input other than the EI which is amply available giving the exploding databases of bio-sequences [23], [24]. However, there are several prices to pay for EI. Firstly, when predicting for entire proteomes (all proteins in an organism), compiling the EI for all is computationally expensive [25]. Secondly, EI is not available for all proteins (intrinsically disordered proteins [26] or *dark proteome* [27]). Thirdly, the improvement is best when the EI is most diverse [28], [29]. Fourthly, predictions based on EI somehow average over an entire family, possibly falling short of distinguishing differences between two different proteins in the same family. The latest, and arguably largest leaps ever in terms of protein structure prediction, namely AlphaFold2, roots on an advanced combination of EI and ML [30]. Although that method predicts protein 3D structures at unprecedented levels of accuracy, AlphaFold2 models are many order of magnitude more computationally expensive than the creation of EI.

The leap of NLP through advanced LMs has been successfully generalized toward understanding the *language of life* through advanced LMs trained on proteins [31], [32], [33], [34], [35], [36], [37], [38], [39]. In analogy to NLP, these approaches interpret an entire protein sequence as a sentence and its constituents – amino acids – as single words. Protein sequences are constrained to adopt particular 3D structures optimized for accomplishing particular functions. These constraints mirror the rules of grammar and meaning in NLP. Since LMs extract features directly from single protein sequences, they might reach performance of the SOA without using EI.

In this project, dubbed *ProtTrans*, we pursued two objectives. Firstly, we explored the limits of up-scaling language models trained on proteins as well as protein sequence databases used for training. Secondly, we compared the effects of auto-regressive and auto-encoding pre-training upon the success of the subsequent supervised training, and compared all LMs trained here to existing state-of-the-art (SOA) solutions using evolutionary information (EI) [40].

2 METHODS

2.1 Data for protein Language Models (LMs)

In this work, we assessed the impact of database size on performance through three data sets (Table 1, SOM Fig. 10): Uniref50 [41], UniRef100 [41], and BFD [24], [42]. The latter merged UniProt [23] and proteins translated from multiple metagenomic sequencing projects, making it the largest collection of protein sequences available at the time of writing even after removal of duplicates from the original BFD. Overall, BFD was about eight times larger than the largest data sets used previously for protein LMs [34]. Despite the 8-fold increase in data, the number of tokens increased only five-fold (Table 1), because UniRef100 sequences were longer than those in BFD (1.6-fold). Without a clear mapping for LMs from NLP to proteins, i.e., the concept of words can be related to single amino acids, a window of amino acids (k-mer motifs [43]) or functional units (domains [44]), we decided to interpret single amino acids as input tokens/words. Thereby, protein databases contain several orders of magnitude more tokens than corpora used in NLP, e.g., Google's Billion Word data set [45] is one of the biggest for NLP with about 829 million tokens (words), i.e. about 500-times fewer than BFD with 393 billion tokens. Interpreting domains as words, would cut the number of tokens in BFD roughly by a factor of 100 (average domain length [46]) still leaving 5-times more tokens in BFD than the Billion Word corpus. Uniref50, UniRef100 and BFD were tokenized with a single space (indicating word-boundaries) between each token. Each protein sequence was stored on a separate line, with lines/proteins representing the equivalent of "sentences". Additionally, an empty line was inserted between each protein sequence in order to indicate the "end of a document"; however, this is only essential for models with auxiliary task (Bert and Albert). Non-generic or unresolved amino acids ([BOUZ]) were mapped to *unknown* (X). For training ProtTXL and ProtT5, the data was transformed to pytorch and tensorflow tensors, respectively on the fly. For ProtBert, ProtAlbert, ProtXLNet and ProtElectra, the data was pre-processed and stored as tensorflow records. Given tensorflow records with terabytes, data sets had to be chunked into 6000 files for thousands of parallel workers.

Data LM	*UniRef50*	*UniRef100*	*BFD*
Number proteins [in m]	45	216	2,122
Number of amino acids [in b]	14	88	393
Disk space [in GB]	26	150	572

TABLE 1: Data Protein LM - UniRef50 and UniRef100 cluster the UniProt database at 50% and 100% pairwise sequence identity (100% implying that duplicates are removed) [41]; BFD combines UniProt with metagenomic data keeping only one copy for duplicates [24], [42]. Units: number of proteins in millions (m), of amino acids in billions (b), and of disk space in GB (uncompressed storage as text).

2.2 *Embeddings* for supervised training

We extracted the information learned by the protein LMs through *embeddings*, i.e., vector representations from the last hidden state of the protein LM (Fig. 1). In the transfer-learning step these embeddings served as input to subsequent supervised training. Although we mostly relied on previously published data sets to ease comparisons to other methods, for the supervised training, we also added a novel test set to refine the evaluation.

Per-residue prediction/single tokens: to predict properties of single tokens (here: single amino acids, dubbed residues when joined in proteins), we used the training set published with NetSurfP-2.0 [15] describing secondary structure in 3- and 8-states (class distribution for all data sets in SOM Tables 6, 5). We also included other public test data sets, namely CB513 [47]), TS115 [48], and CASP12 [49]. Each of those has severe limitations (CASP12: too small, CB513 and TS115 redundant and outdated). Therefore, we added a new test set using only proteins published after the release of NetSurfP-2.0 (after Jan 1, 2019). We included proteins from the PDB [50] with resolutions $\leq$ 2.5Å and $\geq$ 20 residues. MMSeqs2 [51] with highest sensitivity (*-s* 7.5) removed proteins with >20% PIDE to either the training

set or to itself. On top, PISCES [52] removed any protein considered by its procedure to have >20% PIDE. These filters reduced the number of new proteins (chains) from 18k to 364 (dubbed set *NEW364*).

Per-protein prediction/embedding pooling: For the prediction of features for entire proteins (analogous to the classification of whole sentences in NLP), the DeepLoc [16] data set was used to classify proteins into (i) membrane-bound vs. water-soluble and (ii) ten classes of subcellular localization (also referred to as cellular compartments).

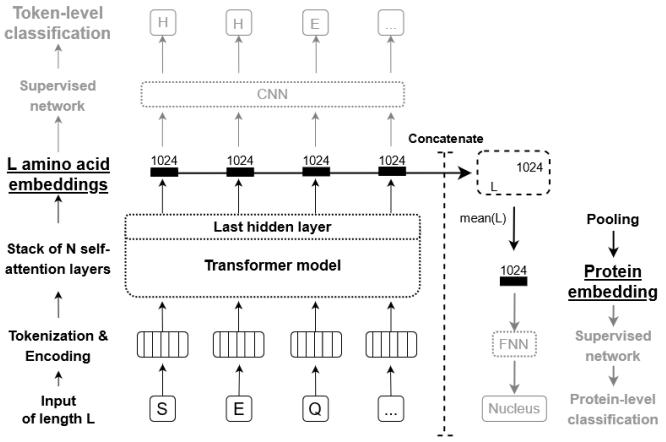


Fig. 1: Feature extraction overview - We give a general overview on how ProtTrans models can be used to derive features (embeddings) for arbitrary protein sequences either on the level of single amino acids or whole proteins and how they can be used for classification tasks on both levels. First, an example protein sequence "SEQ" is tokenized and positional encoding is added. The resulting vectors are passed through any of our ProtTrans models to create context-aware embeddings for each input token, i.e. each amino acid. Here, we used the last hidden state of the Transformer's attention stack as input for downstream prediction methods. Those embeddings can either be used directly as input for prediction tasks on the level of individual tokens, e.g. a CNN can be used to predict an amino acid's secondary structure. Alternatively, those embeddings can be concatenated and pooled along the length-dimension to get fixed-size embedding irrespective of the input length, i.e., global average pooling is applied. The resulting protein-level embedding can be used as input for predicting aspects of proteins, e.g., a FNN can be used to predict a protein's cellular localization.

2.3 Data for unsupervised evaluation of embeddings

We also assessed the information captured by the embeddings extracted from the protein LMs by projecting the high-dimensional representations down to two dimensions (2D) using t-SNE [53]. Toward this end, we took annotations from several sources. First, a non-redundant (PIDE<40%) version of the SCOPe database [54] (release 2.07 with 14,323 proteins). Second, we mapped proteins into the three major domains of life (*archaea*, *bacteria*, or *eukarya*) or to viruses (removing all proteins with missing classifications). The number of iterations for the t-SNE projections was set to 3,000 and the perplexity to 30 for all plots with the exception of the amino acid plot for which we used a perplexity of 5. All visualizations used the same random seed (42).

2.4 Step 1: Protein LMs extract embeddings

We trained six successful LMs in NLP (T5 [55], Electra [56], BERT [57], Albert [58], Transformer-XL [59] and XLNet [11]) on protein sequences. *BERT* was the first bidirectional 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 which allows to increase the number of attention heads (64 chosen here). *Electra* tries to improve the sampling-efficiency of the pre-training task by training two networks, a generator and a discriminator. Instead of only reconstructing corrupted input tokens, the generator (BERT) reconstructs masked tokens, potentially creating plausible alternatives, and the discriminator (Electra) detects which tokens were masked. This enriches the training signal as the loss can be computed over all tokens instead of the subset of corrupted tokens (usually only 15%). *T5* uses the original transformer architecture proposed for sequence translation, which 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. Only later, models used either the encoder (BERT, Albert, Electra) or the decoder (*TransformerXL, XLNet*), but T5 showed that this simplification might come at a certain prize as it reaches state-of-the-art results in multiple NLP benchmarks. Additionally, it provides the flexibility to apply different training methodologies and different masking strategies, e.g., T5 allows to reconstruct spans of tokens instead of single tokens.

As self-attention is a set-operation and thus order-independent, Transformers require explicit positional encoding. Models trained with sinusoidal position signal like BERT, Albert or Electra, can process only sequences shorter or equal to the length of the positional encoding which has to be set before training. Due to the huge memory requirement of Transformer-models, this parameter is usually set to a value lower than the longest proteins, e.g., Titin with 33k residues. Here, we trained models that were affected by this limitations (ProtBERT, ProtAlbert, ProtElectra) first on proteins of length ≤ 512, then on proteins ≤ 1024. Only setting the length of the positional encoding to 40k after pre-training allowed the models to process protein sequences up to a length of 40k. In contrast to this, TransformerXL introduced a memory that allows it to process sequences of arbitrary length. Still, the model cuts sequences into fragments but allows for flow of information between fragments for longer proteins by re-using hidden states of fragments which have already been processed. While its memory is uni-directional as fragments are processed sequentially, TransformerXL captures only uni-directional context within one memory fragment (auto-regressive) while XLNet, which uses a similar memory mechanism to process sequences of arbitrary length, allows to gather bidirectional context within one memory fragment.

In contrast to this, T5 learns a positional encoding for each attention head that is shared across all layers. This way, the model learned to combine the relative offset between residue pairs of lower layers, enabling the model to make predictions beyond the actual length of the positional encoding. No auxiliary tasks like BERT's next-sentence prediction were used for any model described here.

ProtTXL, ProtBert, ProtXLNet, ProtAlbert and ProtElectra were trained on UniRef100, ProtT5 on UniRef50, and ProtTXL, ProtBert & ProtT5, on BFD (Table 2). Largely, we

Hyperparameter	ProtTXL		ProtBert		ProtXLNet	ProtAlbert	ProtElectra	ProtT5-XL		ProtT5-XXL	
Dataset	BFD100	UniRef100	BFD100	UniRef100	UniRef100	UniRef100	UniRef100	UniRef50	BFD100	UniRef50	BFD100
Number of Layers	32	30	30		30	12	30	24		24	
Hidden Layers Size	1024		1024		1024	4096	1024	1024		1024	
Hidden Layers Intermediate Size	4096		4096		4096	16384	4096	16384		65536	
Number of Heads	14	16	16		16	64	16	32		128	
Positional Encoding Limits	-		40K		-	40K	40K	-		-	
Dropout	0.15		0.0		0.1	0.0	0.0	0.1		0.1	0.0
Target Length	512		512/2048		512	512/2048	512/1024	512		512	
Memory Length	512		-		384	-	-	-		-	
Masking Probability	-		15%		-	15%	25%	15%		15%	
Local Batch Size	8	5	32/6	30/5	2	21/2	18/7	8	4	8	4
Global Batch Size	44928	22464	32768/6144	15360/2560	1024	10752/1024	9216/3584	2048	4096	2048	4096
Optimizer	Lamb		Lamb		Adam	Lamb	Lamb	AdaFactor		AdaFactor	
Learning Rate	0.0005	0.002	0.002		0.00001	0.002	0.002	0.01		0.01	
Weight Decay	0.0	0.01	0.01		0.01	0.01	0.01	0.0		0.0	
Training Steps	40.7K	31.3K	800K/200K	300K/100K	847K	150K/150K	400K/400K	991K	1.2M	343K	920K
Warm-up Steps	13.6K	5.5K	140K/20K	40K/0K	20K	40K/5K	40K/40K	10K		10K	
Mixed Precision	FP16 Model Weight Fp32 Master Weight		None		None	None	None	None		None	
Number of Parameters	562M	409M	420M		409M	224M	420M	3B		11B	
System	Summit	Summit	TPU Pod		TPU Pod	TPU Pod	TPU Pod	TPU Pod		TPU Pod	
Number of Nodes	936		128	64	64	64	64	32	128	32	128
Number of GPUs/TPUs	5616		1024	512	512	512	512	256	1024	256	1024

TABLE 2: Large-scale Deep Learning: the table shows the configurations for pre-training the protein LMs introduced here (ProtTXL, ProtBert, ProtXLNet, ProtAlbert, ProtElectra, ProtT5) using either Summit, a TPU Pod v2 or a TPU Pod v3.

transferred configurations successfully from NLP to protein sequences [36], [39], [60], with the exception of the number of layers that was increased to optimize memory utilization.

ProtTXL: The Transformer-XL[1] was trained using both UniRef100 and BFD-100 datasets (referred to as *ProtTXL* and *ProtTXL-BFD*, respectively; Table 2). Both models used a dropout rate of 15%, a memory length of 512 tokens and using mixed precision. The number of layers, number of heads, batch size, learning rate, weight decay, training steps and warm-up steps were adjusted according to training set size as well as GPU utilization. The number of warm-up steps was set to cover at least one epoch for each data set. We tested initial learning rates between 0.001 and 0.005 which were increased linearly at every training step over the warm-up period. To avoid model divergence during training, the learning rate had to be (i) reduced along with the warm-up steps (for BFD), or (ii) increased for both (for Uniref100). Even after increasing the warm-up steps to two epochs, the maximum learning rate remained at 0.0025 for both data sets. Beyond this point, the training diverged. Using weight decay to regularize the network increased the GPU memory usage as it required to compute the norm of all weight vectors on our models, thus reducing the batch size. ProtTXL-BFD was trained for 40k steps in total, with 13.6k warm-up steps using a learning rate of 0.0005, while ProtTXL was trained for 31k steps with 5k warm-up steps using a learning rate of 0.002. The Lamb optimizer was able to handle the resulting batch sizes of 44k and 22k for ProtTXL-BFD and ProtTXL, respectively, without divergence.

ProtBert: BERT[2] was trained using both UniRef100 and BFD-100 datasets (referred to as *ProtBert* and *ProtBert-BFD*, respectively; Table 2). Compared to the original BERT publication, the number of layers was increased. Unlike Transformer-XL which was trained on Nvidia GPUs, mixed-precision was not used to train other models because those were trained on TPUs. Similar to the BERT version trained in the Lamb paper [61], ProtBert was first trained for 300k steps on sequences with a maximum length of 512 and then for another 100k steps on sequences with a length of a maximum length of 2k. While ProtBert-BFD was trained for 800k steps, then for another 200k steps for sequences with maximum length of 512 and 2k, respectively. This allows the model to first extract useful features from shorter sequences while using a larger batch size, rendering training on longer sequences more efficient.

ProtAlbert: We trained Albert[3] on UniRef100 (*ProtAlbert*; Table 2). We used the configuration from the official GitHub repository for Albert (version: xxlarge v2). For Albert the number of layers is increased through the number of times, Albert stacks its single layer. Compared to the original publication, we achieved increasing the global batch size from 4096 to 10752 on the same hardware. The reason for this counter-intuitive effect is the reduced vocabulary size in proteins: the entire diversity of the protein universe is realized by 20 different amino acids, compared to tens of thousands of different words. Similar to ProtBert, ProtAlbert was first trained for 150k steps on sequences with a maximum length of 512 and then for another 150k steps on sequences with a maximum length of 2k.

ProtXLNet: XLNet[4] was trained on UniRef100 (*ProtXLNet*) using the original NLP configuration [11] (Table 2) except for the number of layers that was increased to 30 layers which reduced the global batch size to 1024. Due to the relatively small batch-size, we used the original optimizer: Adam with a learning rate of 0.00001. The model was trained through more steps, i.e. 20k warm-up and 847k steps to compensate for the smaller batch-size of this model.

ProtElectra: Electra[5] consists of two models, a generator and discriminator (same number of layers, generator 25% of the discriminator's hidden layer size, hidden layer intermediate size, and number of heads). We copied Electra's NLP configuration with two changes: increasing the number

1. https://github.com/NVIDIA/DeepLearningExamples/
2. https://github.com/google-research/bert
3. https://github.com/google-research/albert
4. https://github.com/zihangdai/xlnet
5. https://github.com/google-research/electra

of layers to 30 and using Lamb optimizer. Again, we split the training into two phases: the first for proteins $\leq$ 512 residues (400k steps at 9k global batch size), the second for proteins $\leq$ 1024 (400k steps at 3.5k global batch size). While ProtTXL, ProtBert, ProtAlbert and ProtXLNet relied on pre-computed tensorflow records as input, Electra allowed to mask sequences on the fly, allowing the model to see different masking patterns during each epoch.

ProtT5: Unlike the previous LMs, T5[6] uses an encoder and decoder [10]. We trained two model sizes, one with 3B (T5-XL) and one with 11B parameters (T5-XXL). T5-XL was trained using 8-way model parallelism, while T5-XXL was trained using 32-way model parallelism. First, T5-XL and T5-XXL were trained on BFD for 1.2M and 920k steps respectively (*ProtT5-XL-BFD*, *ProtT5-XXL-BFD*). In a second step, ProtT5-XL-BFD and ProtT5-XXL-BFD were fine-tuned on UniRef50 for 991k and 343k steps respectively (*ProtT5-XL-U50*, *ProtT5-XXL-U50*). Contrary to the original T5 model which masks spans of multiple tokens, we adopted BERT's denoising objective to corrupt and reconstruct single tokens using a masking probability of 15%. All T5 models used the AdaFactor optimizer with inverse square root learning rate schedule for pre-training. Like ProtElectra, T5 masks each sequence on the fly. In our hands, the encoder outperformed the decoder on all benchmarks significantly and running the model in half-precision during inference instead of full-precision had no effect on performance but allowed to run the model on on a single Nvidia TitanV (12GB vRAM). Thus, we dropped the decoder from further analysis which cuts model size by half during inference. For completeness, we made weights for encoder and decoder publicly available.

2.5 Step 2: Transfer learning of supervised models

To best analyze the impact of transfer learning, we deliberately kept the supervised models using the embeddings from the protein LMs as input minimal. In particular, compared to SOA solutions such as NetSurfP-2.0, all our experiments used the pre-trained LMs as feature extractors without fine-tuning, i.e. without gradient back-propagating to the LMs. Throughout, we extracted the embeddings from the last hidden state of the pre-trained LMs as described in detail elsewhere [32]. To briefly summarize (Fig. 1): we applied tasks on two different levels, namely on the level of single tokens (per-residue) and whole sentences through pooling (per-protein) predictions. For the **per-residue prediction**, we input the embeddings into a two-layer convolutional neural network (CNN). The first CCN layer compressed the embeddings to 32 dimensions using a window size of 7. The compressed representation was fed into two different CNNs (each with window size 7). One learned to predict secondary structure in 3-states, the other in 8-states. The network was trained on both outputs simultaneously by adding their losses (multi-task learning). For ProtBERT-BFD embeddings we additionally trained three other models: logistic regression, FNN and LSTM. Similar to the CNN, the two-layer FNN first compressed the output of the language model down to 32 dimensions which the second FNN-layer used to predict 3- and 8-states simultaneously. The bi-directional LSTM compressed the embeddings down to 16 dimensions. Concatenating both directions, the resulting 32 dimensional representation was used by a FNN layer to predict 3- or 8-states. As the CNN performed best (SOM Table 10), we used CNNs throughout. For the **per-protein prediction**, we also extracted the embeddings from the last layer of the protein LMs. However, then we pooled the representations over the length-dimension resulting in a fixed-size representation for all proteins. Using ProtBERT-BFD embeddings, we compared alternative pooling strategies (SOM Table 10) and chose mean-pooling for all further experiments. The resulting vector was used as an input to a single feed forward layer with 32 neurons which compressed information before making the final predictions for both per-protein tasks, i.e., the prediction of subcellular localization and the differentiation between membrane-bound and water-soluble proteins, simultaneously (multi-task learning).

2.6 Hardware

HPC hardware is advancing both through infrastructure of supercomputers, such as Fugaku [62], Summit [1] or the SuperMUC-NG [63], and through its components, such as TPU pods [2], specifically designed to ease large scale neural network training for users. Concurrent software improvements in form of more efficient libraries such as Horovod [6] allow executing general purpose code on large distributed clusters with minor code changes. In this section we give details on the hard- and software used for training language models on large protein sequence databases.

ORNL Summit & Rhea: The Oak Ridge National Laboratory (ORNL) provides several clusters for researchers who need computational resources not provided by research facilities such as universities. Here, we used *Summit* and *Rhea*. Summit was used to train the deep learning models, while Rhea was used for the pre-processing of data sets including the distributed generation of tensorflow records.

Summit is the world's second fastest computer, consisting of approximately 4618 nodes. Each node has two IBM POWER9 processors and six NVIDIA Volta V100 with 16GB of memory each [1]. Every POWER9 processor is connected via dual NVLINK bricks, each capable of a 25GB/s transfer rate in both directions. A single node has 0.5 TB of DDR4 main memory and 1.6TB of non-volatile memory that can be used as a burst buffer. Summit is divided into racks with each rack having 18 nodes. In all of our experiments we reserved 936 nodes for training. As having nodes on the same rack decreases the communication overhead, we reserved entire racks.

The smaller cluster (Rhea) contains two partitions: Rhea and GPU. The Rhea partition has 512 node, each with 128 GB of memory and two Intel® Xeon® E5-2650. The GPU partition has only 9 nodes, each with 1 TB of memory and two Intel® Xeon® E5-2695. Reha reduced the time needed for creating tensorflow records for the BFD dataset from 7.5 months (!) to fewer than two days, by converting the original sequential script to distributed processing using MPI. The generation script used two nodes of the GPU partition, with a total of 112 parallel threads.

Google TPU Pod: In 2016, Google introduced tensor processing unit (TPU) as its application-specific integrated

6. https://github.com/google-research/text-to-text-transfer-transformer

circuit optimized for training neural networks. TPUs can be accessed through Google Cloud. Training the protein LMs used both older TPU generation (V2) with 256 cores, and the latest TPU generation (V3) with 512 and 1024 cores. These cores are divided into hosts with each host having access to 8 cores. Consequently, we had access to 32, 64 and 128 hosts for V2/V3-256, V3-512 and V3-1024, and each core had 8 GiB and 16 GiB of high-bandwidth memory for V2 and V3. Training on the TPUs required access to a virtual machine on Google Cloud and storage on Google Bucket [64].

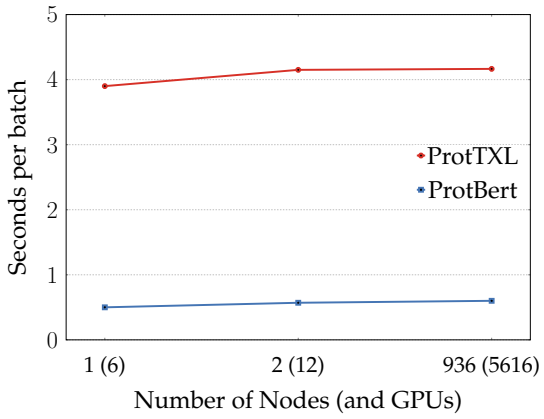


Fig. 2: Large Scale Dataset Training: The figure shows the overhead of increasing the number of nodes/gpus for both ProtTXL (blue; low) and ProtBert (red; high). The overhead increases slightly from 1 to 2 nodes but remains constant even when scaling up to 936 nodes with a total of 5616 GPUs. Having a low overhead means the model has a near-linear scale-up across thousands of GPUs, upper-bounded by the theoretical scale-up.

2.7 Software

Summit integrates several pre-configured modules which include the most popular libraries and tools required for simulation, deep learning, distributed training and other purposes. We used the IBM Watson Machine Learning module versions 1.6.0 and 1.6.2 for our deep learning training. In contrast to this, the Google Cloud server, which we used for the TPU Pod training, had to be configured manually because only the operating system was installed.

Pytorch was used to train ProtTXL, tensorflow to train ProtBert, ProtAlbert, ProtXLNet, ProtElectra and ProtT5. Both libraries used the Horovod framework [6] to train the models on distributed clusters such as Summit. Horovod supports distributed GPU training with minimal change in the code. It supports different backends including MPI, NCCL and IBM PowerAI distributed deep learning (DDL). We tested all three backends and found DDL to be the fastest for our training purpose on Summit. The time needed to finish a single batch with ProtTXL-BFD increased from one to two nodes due to the communication overhead (Fig. 2). After two nodes the communication overhead plateaued, even when scaling up to 936 nodes with 5616 GPUs. Summit has integrated DDL in their Watson Machine Learning module which comes with most DDL libraries including pytorch, tensorflow, apex, DDL and horovod. However, Summit has only a license for using DDL up to 954 nodes. Contrary to Summit, training on TPU Pods did not require any changes in the Tensorflow code to use either a single TPU host or to distribute workload among multiple TPU hosts.

Mixed precision allows to fit larger models and batch sizes into GPU memory by using 16-bit precision only or a mix of 16-bit and 32-bit precision. Nvidia's APEX library [65] was used for mixed precision training of ProtTXL, because APEX supports pytorch. APEX supports four types of mixed precision and model weights storing: 1) Pure 32-bit precision; this is the regular training without using mixed precision. 2) Pure 16-bit precision, all the model weights will be stored in 16-bit rather than 32-bit. 3) Mixed Precision, for different layer types depends on previously tested whitelist/blacklist by Nvidia; some weights will be stored in 32-bit while others in 16-bit format. 4) Almost FP16, storing all model weights at 16 Bit precision; exception: batch-normalization layers, while keeping a master copy of the model's weights in 32-Bit. Using pure 16-bit training leads to a big part of activation gradient values becoming zeros, leading to divergence during training. This problem is solved using Almost FP16 because there is a master copy of the model's weights in 32-Bit. As ProtTXL training became instable when training with 16 Bit precision, we switched to almost half precision training. We did not use mixed-precision for models trained on TPUs.

Another optimization technique/library crucial for our training on Summit was IBM's large model support (LMS) [66]. Similar to gradient checkpointing [67], LMS virtually extends the GPU memory by outsourcing parts of the model from GPU to main memory. This allows training models larger than the GPU memory. The obvious drawback of LMS is the increase in training time due to shuttling data between CPU and GPU and back. However, the reduced memory consumption of the model allows to increase the batch size, potentially compensating for the communication overhead. Compared to gradient checkpointing, LMS provides easier integration into existing code by operating directly on a computational graph defined by users and automatically adds swap-in and swap-out nodes for transferring tensors from GPU memory to main memory and vice versa. We have tested LMS on ProtTXL as well as ProtBert (Figure 2). As Pytorch and tensorflow have different strategies to integrate LMS, we also compared the effect of LMS on batch-size, model size and training time using the two different libraries. ProtTXL was used to evaluate the effect of Pytorch's implementation of LMS while ProtBert was trained for a few steps BFD using Summit to evaluate tensorflow's implementation of LMS. Training ProtBert for a few steps was sufficient to assess the effect of LMS on batch-size, model size as well as an estimate of training time. In the end, we used LMS only for ProtTXL to strike a balance between model size and training time. The number of LM parameters could be increased by about 15.6% for ProtTXL-BFD and to 6.6% for ProtBert (3a). Additionally, we could increase the batch size by 700% for ProtTXL-BFD (Figures 3b and 3c). The NV-Link between CPU and GPU on Summit-nodes, reduced the training time for ProtTXL by 60%while it increased by 72% for ProtBert (Figure 3d).

3 Results

3.1 Step 1: Unsupervised protein LMs informative

Embeddings extract constraints about protein function and structure learned by the protein LMs in the first self-

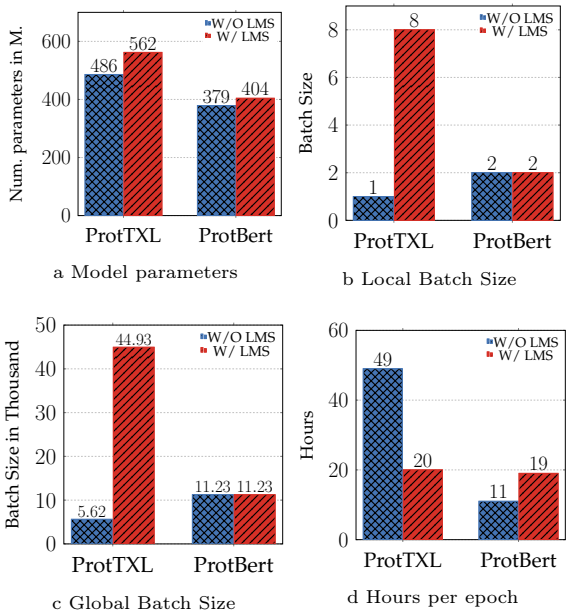


Fig. 3: Large Scale Deep Learning Training: The figures show the effect of enabling (red bars) or disabling (blue bars) large model support (LMS) on both, model size as well as batch size, when we tested ProtTXL or ProtBert on Nvidia V-100 16GB GPUs. It highlights the difference between applying LMS using PyTorch (ProtTXL) or tensorflow (ProtBert). Panel (a) shows the effect of using LMS on the maximum model size that can fit in a single V-100 memory. Panels (b,c) compare the effect of LMS on the maximum local (b) and global batch size (c) that can fit in the GPU. The number of hours required to finish a single epoch using 936 nodes, each with 6 GPUs when LMS being enabled is shown in (d).

supervised step of pre-training on raw protein sequences. Using t-SNE [53], we visualized this information by projecting the embeddings onto 2D and annotated structural, functional or evolutionary features. Using attention maps, we analyzed the DNA-binding zinc-finger motif well conserved in evolution.

Capturing biophysical features of amino acids. Applying t-SNE to the uncontextualized token embedding layer visualized information extracted by the LMs for individual amino acids independent of their context (residues next to it). As previously established for another protein LM [39], the t-SNE projections (e.g. ProtT5-XL-U50 SOM Fig. 14A or ProtBert-BFD SOM Fig. 15A) suggested that all LMs captured essential biophysical amino acid features, including charge, polarity, size, hydrophobicity, even to the level of aliphatic ([AILMV]) vs. aromatic ([WFY]).

We compared the embedding projection with a randomly initialized model of identical architecture to ascertain that the observed effects did not originate from coincidental signals originating from projecting high-dimensional data (Fig. 4A) or some inductive bias of neural networks [68]. The random projection clearly did not carry biophysical information, while the embeddings projection did.

Capturing protein structure classes. To assess which aspects of protein structure were captured by the unsupervised LMs, we averaged over the length-dimension of the representations derived from the last layer of each model (see Fig. 1 for a sketch) to derive fixed size representations for each protein in the database. We annotated structural class through the SCOPe database [54] (Methods). ProtBert-BFD and especially ProtT5-XL-U50 embeddings visually separated the proteins best (details in SOM Figs. 14-19). Although sequence length was not explicitly encoded and our pooling squeezed proteins into a fixed vector size, all models separated small from long proteins (brown, e.g. ProtT5-XL-U50 SOM Fig. 14D). All models also distinguished between water-soluble and transmembrane proteins (light blue, e.g. ProtT5-XL-U50 SOM Fig. 14D) and, to some extent, between proteins according to their secondary structure composition(e.g. all-alpha (dark blue) vs. all-beta (dark green) ProtT5-XL-U50 Fig. 14D). While having much higher entropy, even the random clustered small proteins from long proteins (brown, Fig. 4B).

Capturing domains of life and viruses. The analysis distinguished three domains of life: *archaea*, *bacteria*, and *eukarya*, along with *viruses* typically not considered as life. We used the same proteins and per-protein pooling as for the SCOPe analysis. All protein LMs captured some organism-specific aspects (e.g. ProtT5-XL-U50 SOM Fig. 14E). Eukarya and bacteria clustered better than viruses and archaea. Comparing different LMs revealed the same trend as for protein structure classes: ProtTXL (SOM 19E) and ProtBert (SOM 16E) produced higher entropy clusters while ProtAlbert (SOM 17E), ProtXLNet (SOM 18E), ProtBERT-BFD (SOM Fig. 15E) and ProtT5-XL-U50 (SOM Fig. 14E) produce visually easier separable clusters.

Capturing protein function in conserved motifs. A similar overall per-protein analysis as for structural classes and domains of life also suggested some clustering according to protein function as proxied by enzymatic activity (EC-numbers [69] and subcellular localization (SOM -1.2 Protein LMs unsupervised). We focused in more detail on the attention mechanism [70] at the core of each Transformer model [10] providing some limited understanding of the AI [71], [72]. We visualized [73] the attention weights of ProtAlbert to analyze the structural motif of a zinc-finger binding domain (SOM Fig. 11) crucial for DNA- and RNA-binding and conserved across diverse organisms. The right part of ProtAlbert' attention heads (SOM Fig. 11; line thickness resembles attention weight) learned to focus mostly on the four residues involved in zinc-binding (residues highlighted in the left part of SOM Fig. 11) which is essential for function.

3.2 Step 2: Embeddings good input to predict

The acid test for proving that the embeddings from protein LMs extracted important constraints is to exclusively use embeddings as input to supervised training of features reflecting structure and function. We proxied this through predictions on two different level, namely on the per-residue or token level (secondary structure) and on the per-protein or sentence level through pooling over entire proteins (localization, and classification into membrane/non-membrane proteins). Protein LMs remained unchanged, i.e. both approaches (per-residue/per-protein) used only embeddings derived from the hidden state of the last attention layer of each protein LM (Fig. 1) without gradient backpropagation to the LM, i.e., LMs were only used as static feature extractors.

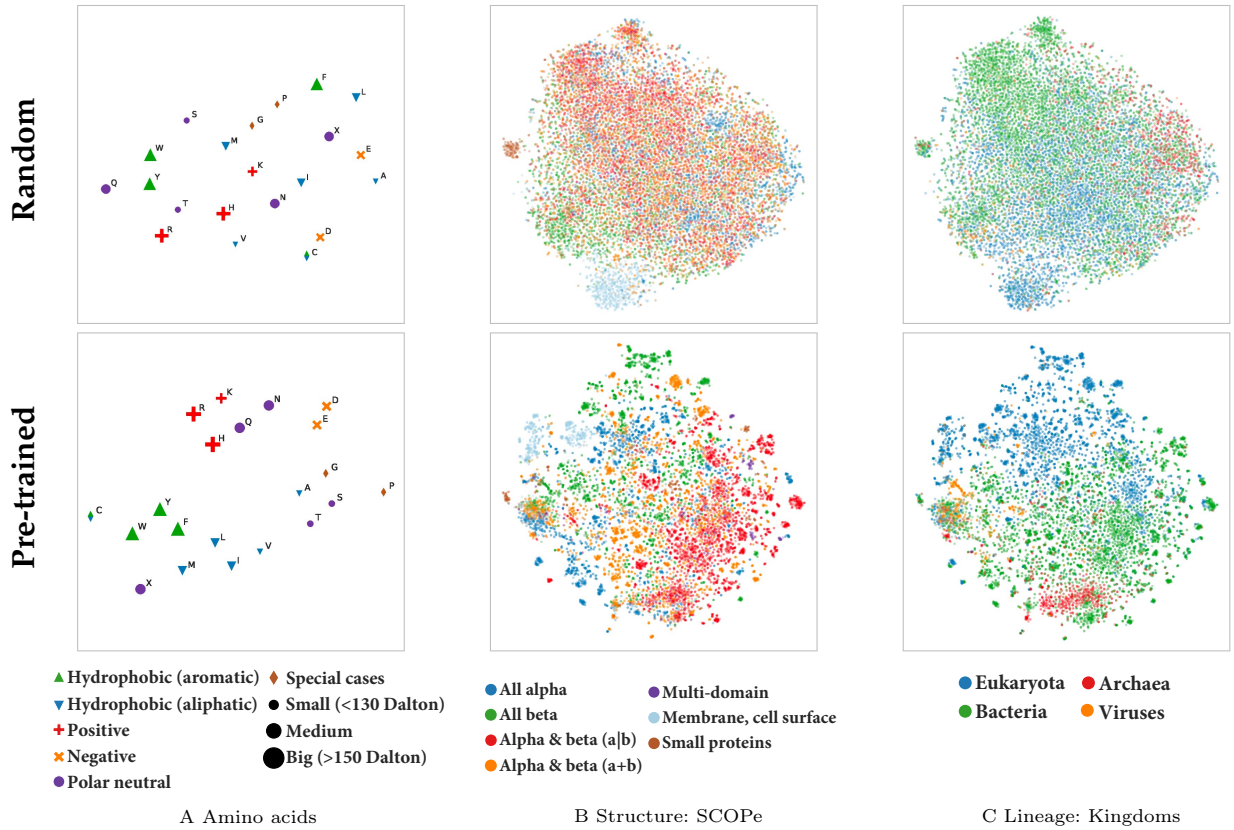


Fig. 4: Protein LMs learned constraints. t-SNE projections visualized information extracted by the unsupervised protein LMs (here best-performing ProtT5-U50; upper row: before training (Random), and lower row: after pre-training on BFD & UniRef50. (A) The left-most column highlights single amino acids by biophysical features. (B) The middle column annotates protein structural class (taken from SCOPe). (C) The right-most column distinguishes proteins according to the kingdom of life in which it is native. Although the random projections on top may suggest some adequacy of the clusters, the trained models shown on the lower row clearly stood out. Incidentally, the comparison of the two also highlighted the potential pitfalls of using t-SNE projections from many dimensions onto 2D: although random, the human might see some correct patterns even in the top row. Most impressive might be the fine-grained distinctions of biophysical features of amino acids (A), however, more surprising are the classifications of entire proteins according to structural class (B) and organism (C). For these, we created embeddings through global average pooling over the representations extracted from the last layer of ProtT5-U50 (average over protein length, i.e. per-protein embeddings; Fig. 1).

3.2.1 *Per-residue secondary structure prediction*

To ease comparability, we evaluated all models on standard performance measures (Q3/Q8: three/eight-state per-residue accuracy, i.e. percentage of residues predicted correctly in either of the 3/8 secondary structure states) and on standard data sets (CASP12, TS115, CB513). To increase the validity of the comparisons, we added a novel, non-redundant test set (dubbed *NEW364*, Methods). For simplicity, we only presented values for Q3 on CASP12 and NEW364 (TS115 and CB513 contain substantial redundancy; Q8 results brought little novelty; all details in SOM Tables 9, 8). As error estimates failed to capture the performance variation between NEW364 and CASP12, we used CASP12 as lower- and NEW364 as upper-limit.

Comparing supervised architectures: We input embeddings from ProtBERT-BFD into four different models for supervised training (Methods): logistic regression (LogReg), FNN, CNN and LSTM. LogReg provided an advanced *baseline* (Q3(LogReg)=74.3-79.3, where the spread is from the lower level for the set CASP12 and the upper for the set NEW364; SOM Table 7). LSTMs and CNNs performed alike and better than LogReg (Q3(CNN)=76.1-81.1% vs. Q3(LSTM)76.1-80.9%). As CNNs are computationally more efficient, we focused on those in the following.

Comparing protein LMs: Trained on UniRef100 (Table 1), ProtBert outperformed other models trained on the same corpus (Tables 3, 8). For ProtTXL and ProtBert we could analyze the influence of database size upon performance: 10-times larger BFD (Table 1) helped ProtBert slightly (ΔQ3: +1.1%) but made ProtTXL worse (ΔQ3: -0.6%; Table 3 and SOM Tables 9, 8). The gain was larger when fine-tuning the two ProtT5 versions (XL and XXL) by first training on BFD and then refining on UniRef50. Consistently, all models fine-tuned on UniRef50 outperformed the versions trained only on BFD (Fig. 5, Table 3, SOM Table 8). Although these gains were consistently numerically higher, the statistical significance remained within the 68% confidence interval (maximal difference: 1.1% compared to one standard error of $\pm$0.5%).

Embeddings reach state-of-the-art (SOA): All models (ProtTXL, ProtBert, ProtAlbert, ProtXLNet, ProtElectra, ProtT5) and all databases (BFD, UniRef50/UniRef100) tested improved significantly over context-free feature extractors such as word2vec-based approaches (DeepProtVec in Fig. 5 and SOM Table 8). Both ProtTXL versions fell short compared to an existing ELMo/LSTM-based solution (DeepSe-

Dataset	*CASP12*	*NEW364*
DeepProtVec	62.9	64.7
ProtTXL*	71.5	72.8
ProtTXL-BFD*	71.7	72.2
DeepSeqVec	73.0	76.0
ProtXLNet*	73.7	77.3
ProtElectra*	73,9	78,1
ProtAlbert*	74.6	78.5
ProtBert*	75.0	80.1
ProtBert-BFD*	75.8	81.1
ESM-1b	76.9	82.6
ProtT5-XXL-BFD*	77.7	81.6
ProtT5-XL-BFD*	77.5	82.0
ProtT5-XXL-U50*	79.2	83.3
ProtT5-XL-U50*	81.4	**84.8**
NetSurfP-2.0	**82.0**	84.3

TABLE 3: The three-state accuracy (Q3) for the per-residue/token-level secondary structure prediction (percentage of residues correctly predicted in either of 3 states: helix, strand, or other) for all protein LMs trained here (marked by star) along with other LMs, namely one word2vec-based approach (DeepProtVec), one LSTM (DeepSeqVec), one transformer (ESM-1b) and one of the current state-of-the-art methods (NetSurfP-2.0) that uses evolutionary information (EI)/multiple sequence alignments (MSAs). Values were compiled for two datasets: one because it is a standard in the field (CASP12, results for two other standard data sets - TS115 and CB513 - in Table 9), the other because it is larger and less redundant (dubbed NEW364 introduced here). Standard errors were computed using bootstrapping: CASP12=±1.6%, NEW364=±0.5%. Highest values in each column marked in bold-face.

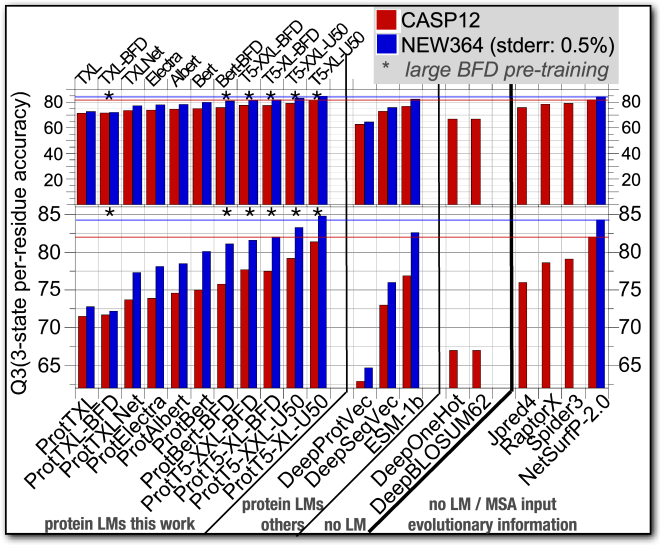


Fig. 5: Per-residue (token-level) performance for secondary structure prediction: CASP12 (red) and NEW364 (blue) constitute two test sets. Protein LMs trained here are shown in the left panel of the figure. Additions of BFD mark pre-training on the largest database BFD, U50 mark pre-training with BFD and refining with UniRef50. We included protein LMs described elsewhere (marked as: protein LMs others, namely ESM-1b [72], DeepProtVec and DeepSeqVec [32]). All embeddings were input to the same CNN architecture. Two approaches used amino acids instead of embeddings as input (marked as: no LMs: DeepOneHot [32] - one-hot encoding - and DeepBLOSUM62 [32] - input BLOSUM62 [74] substitution matrix), as well as, to the current state-of-the-art (SOA) method NetSurfP-2.0 [15], and Jpred4 [75], RaptorX [76], [77], Spider3 [78]. The rightmost four methods use MSA as input (marked as: MSA input evolutionary information). While only rotT5-XL-U50 reached the SOA without using MSAs, several protein LMs outperformed other methods using MSA. All protein LMs other than the context-free DeepProtVec improved significantly over methods using only amino acid information as input. One interpretation of the difference between the two data sets is that CASP12 provided a lower and NEW364 an upper limit. The top row shows the complete range from 0-100, while the lower row zooms into the range of differences relevant here.

qVec [32]) while all other Transformer-models outperformed DeepSeqVec. Embeddings extracted from another large Transformer (ESMB-1b [72]), improved over all our non-ProtT5 models (Figs. 5 and SOM Table 8)). Most solutions using only embeddings as input were outperformed by the state-of-the-art method NetSurfP-2.0 [15] using evolutionary information (Fig. 5 and SOM Tables 9, 8). However, ProtT5-XL-U50 reached nearly identical performance without ever using multiple sequence alignments (MSA). Analyzing the average Q3 per protein of both models for set NEW364 in more detail (SOM Fig. 12), revealed that 57% of the proteins were predicted with higher Q3 by ProtT5-XL-U50 (CASP12 was too small for such a differential analysis).

LMs shine for small families: The size of protein families follows the expected power-law/Zipf-distribution (few families have many members, most have fewer [80]). To simplify: families with fewer members carry less evolutionary information (EI) than those with more. One proxy for this is the number of effective sequences (Neff), i.e., the number of sequences in an MSA clustered at 62% PIDE [79], [81]. We analyzed the effect of Neff by comparing NetSurfP-2.0 (using MSAs/EI) to ProtT5-XL-U50 (not using MSAs) using four subsets of NEW364 using different Neff cutoffs, i.e., the subset of proteins without any hit (Neff=1, 12 proteins), less than 10 hits (Neff<=10, 49 proteins) and Neff>10 (314 proteins) (Fig. 6). Details on the MSA generation are given in SOM. ProtT5XL-U50 improved most over NetSurfP-2.0 for the smallest families (Neff=1).

More samples better performance of protein LMs: despite substantial differences in training corpus, hyperparameter choices and transformer model peculiarities, the LMs trained here exhibited a similar trend: correlating performance and the number of samples presented during the first step training of the protein LMs (*pre-training*). Toward this end, we computed the *number of samples* as the product of the *number of steps* and the *global batch size* (Fig. 7; Spearman's ρ=0.62). In particular, comparing the two largest models trained by us (ProtT5-XL and ProtT5-XXL) suggested that seeing more samples during pre-training might be more beneficial than increasing model size.

3.2.2 *Per-protein localization & membrane prediction*

To investigate per-protein (sentence-level) predictions of protein function, we trained FNNs on sub-cellular localization (also referred to as *cellular compartment*) in ten classes and on the binary classification of membrane vs. non-membrane (also referred to as *globular*) proteins. Levels of ten-state (Q10 for localization) and two-state (Q2 for membrane/globular) measured performance. Toward this end, we derived per-residue embeddings from the last hidden layer of the protein LM and pooled over the length-dimension/entire protein (Fig. 1).

Mean-pooling performed best: Using ProtBERT-BFD embeddings we compared four different pooling strategies

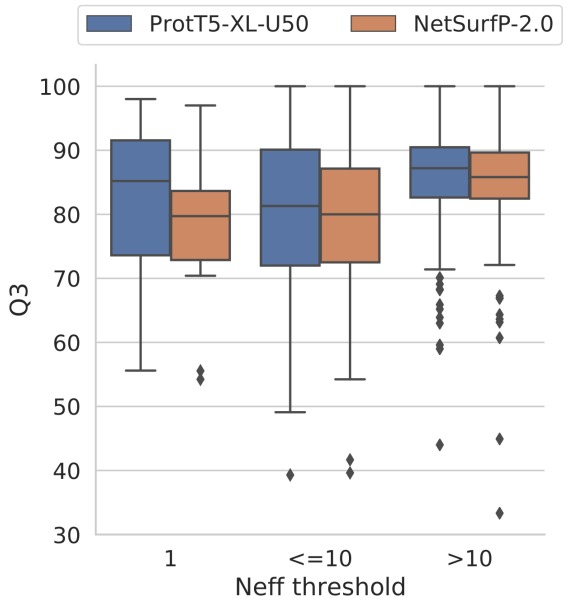


Fig. 6: Effect of MSA size. We used our new test set (NEW364) to analyze the effect of the size of an MSA upon secondary structure prediction (Q3) for the two top methods (both reaching Q3=84.3%): NetSurfP-2.0 (using MSA) and ProtT5-XL-U50 (not using MSA). As proxy for MSA size served Neff, the number of effective sequences [79] (clustered at 62% PIDE): leftmost bars: MSAs with Neff=1, middle: $Neff \& \leq 10$, right: Neff>10. As expected ProtT5-XL-U50 tended to reach higher levels than NetSurfP-2.0 for smaller families. Less expected was the almost on par performance for larger families.

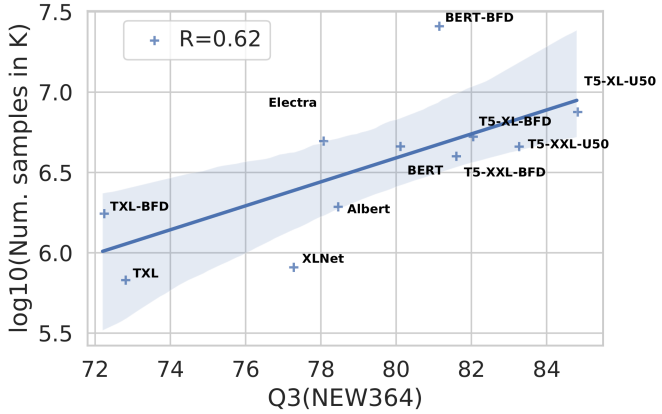


Fig. 7: Number of samples seen during pre-training correlates with performance - We compared 3-state secondary structure prediction performance (Q3) on NEW364 for all LMs trained here against the number of samples seen during pre-training (training steps times global batch-size in K). For simplicity, we dropped the prefix "Prot" from the models in this plot. Despite the peculiarities of each of the Transformers trained here, Spearman's ρ of 0.62 indicates that there might be a common trend towards seeing more samples during pre-training.

for collapsing per-residue (token-level) embeddings, the dimensions of which differ for proteins of different length, into representations of fixed length. These were min-, max-, and mean-pooling, as well as, the concatenation of those three (*concat*). The first two (min/max) performed almost fourteen percentage points worse for localization (Q10) and about three for membrane/other (Q2) compared to the others (mean/*concat*, Table 10). While mean-pooling and *concat* performed similarly for the classifcation task

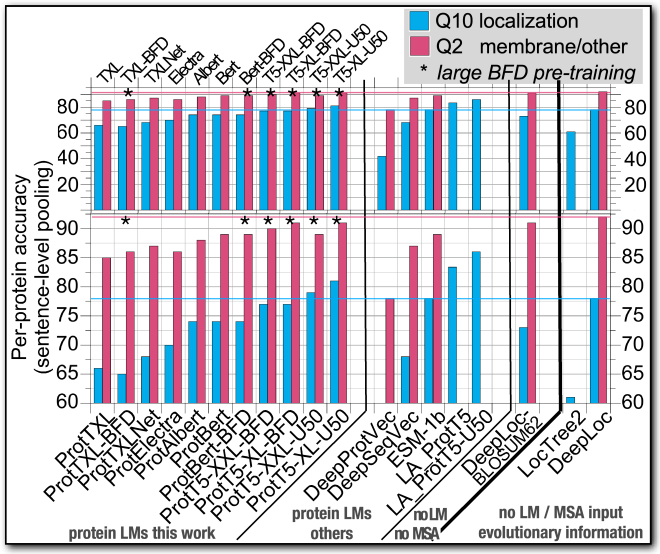


Fig. 8: Per-protein (sentence-level) performance: The prediction of localization in 10 states (lower bars in cyan: Q10: percentage of proteins with 1 of 10 classes correctly predicted) and the classification of membrane/other (higher bars in magenta: Q2: percentage of proteins correctly classified in either of two classes). Embeddings were derived from protein LMs by mean-pooling, i.e. averaging over the length of the entire protein (Fig. 1). Abbreviations as in Table 5 except for one method using neither LMs nor MSA (no LM no MSA: DeepLoc-BLOSUM62 [16]), and two methods using MSAs (MSA input evolutionary information): the current state-of-the-art (SOA) method (performance marked by horizontal thin lines in magenta and cyan) DeepLoc [16], and LocTree2 [82]. Almost all LMs outperformed LocTree2 and a version of DeepLoc not using MSAs (DeepLoc-BLOSUM62). Only, ProtT5-XXL-U50 and ProtT5-XL-U50 outperformed the SOA. A recent method optimized localization prediction from embeddings (ProtT5) through a light-attention mechanism; it clearly outperformed the SOA without using MSAs (LA_ProtT5 & LA_ProtT5-U50 [83]). The top row shows the complete range from 0-100, while the lower row zooms into the range of differences relevant here.

(membrane/other), mean-pooling outperformed *concat* for localization by about ten percentage points (Table 10). In the following, we used only mean-pooling to benchmark the per-protein/sentence-level predictions.

Comparison of LMs: the per-protein prediction of localization in 10 states largely confirmed the trend observed for per-residue secondary structure prediction (Q3/Q8) in several ways: All LMs introduced in this work (marked by * in Table 4) clearly outperformed the un-contextualized word2vec-based approaches (DeepProtVec; Fig. 8, Table 4). Except for ProtTXL and ProtXLNet, all transformers trained here outperformed the previous ELMo/LSTM-based solution (DeepSeqVec). Increasing the corpus for pre-training the protein LMs 10-fold appeared to have little effect (Prot* vs. Prot*-BFD in Fig. 8 and Table 4). In contrast, fine-tuning ProtT5 models already trained on BFD using UniRef50 improved (Prot*/Prot*-BFD vs. Prot*-U50 in Fig. 8 and Table 4). Although most embedding-based approaches were outperformed by the state-of-the-art method (*DeepLoc*) which uses multiple sequence alignments (MSAs) as input, both ProtT5 models trained on UniRef50 outperformed DeepLoc without using MSAs: Q10, Fig. 8 and Table 4.

Similar for membrane/other: Results for the classification into membrane/other (Q2; Table 4), largely confirmed those

Dataset	*Q10: Localization*	*Q2: Membrane/other*
DeepProtVec	42	78
ProtTXL*	66	85
ProtTXL-BFD*	65	86
DeepSeqVec	68	87
ProtXLNet*	68	87
ProtElectra*	70	86
ProtAlbert*	74	88
ProtBert*	74	89
ProtBert-BFD*	74	89
ESM-1b	78	89
ProtT5-XXL-BFD*	77	90
ProtT5-XL-BFD*	77	91
ProtT5-XXL-U50*	79	89
ProtT5-XL-U50*	**81**	91
DeepLoc	78	**92**

TABLE 4: Per-protein prediction of protein function: Given is the performance for two tasks that proxy the prediction of aspects of protein function, namely the prediction of subcellular localization (Localization) in ten states (Q10) and the classification of proteins into membrane-bound/other (Membrane/other) in two states (Q2). Values mark all protein LMs introduced here (marked by star) along with other LMs, namely one word2vec-based approach (DeepProtVec), one LSTM-based (DeepSeqVec), one transformer-based (ESM-1b) and the current state-of-the-art method (DeepLoc) that, unlike all other methods shown, used multiple sequence alignments (MSAs)/evolutionary information (EI) for the values shown. All values based on a standard, public data set [16]. Highest values in each column marked in bold-face.

obtained for localization (Q10) and secondary structure (Q3/Q8): (1) ProtT5 LMs fine-tuned on UniRef50 performed best without MSAs, (2) the 10-fold larger pre-training BFD had no noticeable effect, (3) our best protein LMs outperformed existing transformer LMs (ESM-1b) (Fig. 8). In contrast to localization and secondary structure, the fine-tuning appeared not to increase performance (Table 4) and both ProtT5 remained 1-2 percentage points below DeepLoc.

3.3 Fast and reliable predictions from embeddings

We compared the time needed to generate representations for EI-based prediction methods and protein language models by generating MSAs/embeddings for each protein in the human proteome (20,353) proteins with a median sequence length of 415 residues). We used the fastest method available, namely MMseqs2 [51], with parameters established by NetSurfP-2.0 to generate MSAs from two databases (UniRef90 with 113M and UniRef100 with 216M proteins; see SOM for more technical details). MMseqs2 was about 16 to 28-times slower than the fastest LMs (ProtElectra and ProtBert), and about 4 to 6-times slower than our best model (ProtT5-XL; Fig. 9). ProtT5-XL, required on average 0.12 seconds to generate embeddings for a human protein, completing the entire human proteome (all proteins in an organism) in only 40 minutes. We also investigated the cross-effect of sequence length and batch-size (SOM Table 11) on the inference speed of different protein LMs. When using a single Nvidia Quadro RTX 8000 with half precision on varying batch-sizes (1,16,32) as well as sequence lengths (128, 256, 512), ProtBert and ProtElectra provided the fastest inference with an average of 0.007 seconds per protein when using a batch size of 32, followed by ProtT5-XL and ProtAlbert (0.025s). The batch-size of most models could have been increased on the same hardware but was limited to allow a direct comparison between all models, due to large memory requirements for ProtT5-XXL. The script for running this benchmark is freely available[7].

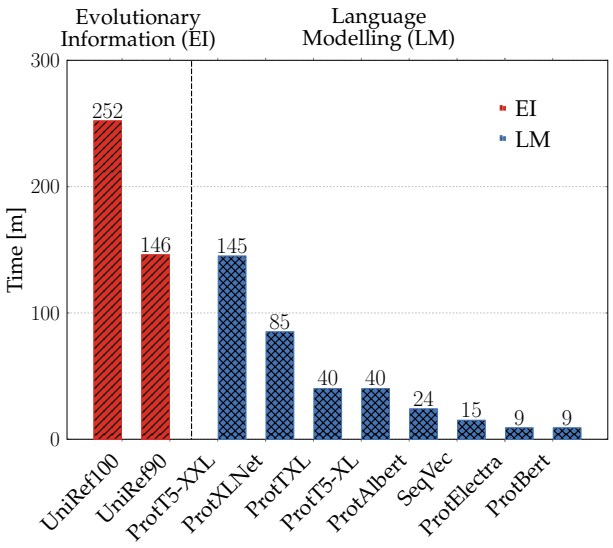


Fig. 9: Inference Speed Comparison: The time required to generate protein representations for the human proteome (20.353 proteins) is compared using either our protein LMs or mmseqs2 (protein sequence search tool [51] used to generate evolutionary information; NetSurfP-2.0 [15] parameters are used). Here, we used mmseqs2 (red bar) to search each protein in the human proteome against two large protein database (UniRef90 and UniRef100 with 113M and 216M proteins, respectively). Only embedding or search time is reported, i.e. no pre-processing or pre-training was measured. mmseqs2 was run on an Intel Skylake Gold 6248 processor with 40 threads, SSD and 377GB main memory, while protein LMs were run on a single Nvidia Quadro RTX 8000 with 48GB memory using half precision and dynamic batch size depending on sequence length (blue bar).

4 Discussion

4.1 *Substantial computing resources needed to cope*

HPC Supercomputers such as Summit [1] and Google's cloud TPU Pod [2], combined with optimized libraries such as IBM DDL [7] and Horovod [6] set the stage for training LMs with billions of free parameters on corpora with terabytes of data in hours or days. Increasing model size improves performance for some NLP applications [12], although the massive data challenges the communication between thousands of nodes and divergence between large batches during training. Here, we presented some solutions to overcome these challenges for training protein LMs by fully utilizing 20% of Summit for TransformerXL [59], as well as, by using one TPU Pod V3-512 for Bert [57], Electra [56], Albert [58] and XLNet [11], and a mix of TPU Pod V3-256, V3-512, and V3-1024 for ProtT5-XL and ProtT5-XXL [55]. This translated into the parallel use of 5616 GPUs on Summit or 256/512/1024 TPU cores on a TPU Pod, while avoiding training divergence with specialized optimizers such as LAMB [61] up to a global batch size of 44K samples (here: protein sequences).

7. https://github.com/agemagician/ProtTrans/tree/master/Benchmark

4.2 Training protein LMs longer most important

Better protein LM means higher performance when using it as input. For most of our work, we proxied the degree to which the pre-trained protein language models (LMs) extracted information through the performance of the second step supervised tasks. Consequently, we considered that the *protein LM improved (or was better)* when the supervised task using this LM as input reached higher performance.

BFD: Largest database for pre-training: We trained our protein LMs on the largest protein database ever used for this purpose, namely BFD [42], more than an order of magnitude larger than UniProt [23], the standard in the field. Although bigger did not equate better in terms of prediction performance on supervised tasks, some of the protein LMs appeared to improve through pre-training on more data (UniRef100 vs BFD, Table 1). Nevertheless, the top performance increase appeared somehow limited given the 10-fold larger data set (e.g. ΔQ3(ProtBert-BFD/UniProt)=1.3%). Instead, the pre-training fine-tuning protocol by which we first trained on the larger but more noisy (more mistakes in protein sequences) and redundant BFD and then continued pre-training using the smaller, less redundant UniRef50 improved performance significantly for both ProtT5 versions (ΔQ3(ProtT5-XL-BFD/U50)=2.8% and ΔQ3(ProtT5-XXL-BFD/U50)=1.4%). Possibly, the refined models were less biased toward large protein families over-represented in redundant databases. The improvement through refined pre-training for ProtT5-XL (3B parameters) exceeded that for ProtT5-XXL (11B parameters), presumably, because it saw more samples when continuing pre-training for a similar amount of time (limited by resources).

This highlighted a remarkable trend common across an immense diversity of protein LMs and corpus: the performance of supervised downstream tasks using the embeddings from pre-trained protein LMs as input increased with the number of samples presented during the LM pre-training (Fig. 7; Spearman's ρ=0.62). We could not observe a similarly consistent trend for model size. However, this might be attributed to some trade-off between both: training for more steps might also require a model with sufficient capacity to absorb the information of the corpus. For instance, while ProtBERT-BFD (420M parameters) saw around 27B proteins during pre-training, it fell short compared to ProtT5-XL-BFD (3B parameters) which saw only around 5B proteins (Figs. 5, Table 3, and SOM Table 8). This finding appeared to confirm results from NLP suggesting that larger models absorb information faster and need less training time to achieve similar performance [84]. However, the comparison between, e.g., ProtT5-XL and ProtT5-XXL suggested a possible cap to this trend as larger models see fewer samples in the same amount of computing power. The clear correlation between performance and samples seen during pre-training combined with the need for sufficient model size spotlighted the crucial role of substantial computing resources (HPC, TPUs, and GPUs): big data needs large models need loads of computational resources.

4.3 Protein LMs learned global constraints

Some rudimentary information about how proteins are formed, shaped, and function has been learned by the protein LMs during pre-training because all models (ProtT5, ProtBert, ProtElectra, ProtAlbert, ProtTXL, ProtXLNet) extracted valuable information as revealed by visualizing embeddings without further supervised training on labeled data. The comparison to a random LMs highlighted two important aspect. Firstly, how easy it is to mis-interpret patterns when projecting from high-dimensional spaces upon 2D: although the randomly initialized protein LMs contained no information, some annotations might have suggested the opposite (top row in Fig. 4 learned NO information). Secondly, the protein LMs did extract important global constraints relevant for protein structure and function (lower row in Fig. 4). This span from the most local (individual token level) biophysical features of the amino acid building blocks (e.g. hydrophobicity, charge, and size, Fig. 4A), over global classifications of proteins according to structural classes (Fig. 4B), to the macroscopic level of the domains of life (Fig. 4C). Global structural properties (e.g. overall secondary structure content, Fig. 4B) and global biochemical properties (e.g. membrane-boundness, SOM Fig. 14B) appeared most distinctive. In contrast, local features relying on short motifs were less separated (EC-numbers: Fig. 14F, localization: Fig. 14C) but still clustered, e.g., for secreted (extracellular) proteins or hydrolases.

On a more fine-grained level, the visual analysis of the attention mechanism at the core of each of the transformer models trained here, confirmed that the protein LMs even picked up more subtle signals of short functional motifs. Specifically, one of the attention heads of ProtAlbert zoomed mostly into the four residues most important for the coordination of zinc-binding (SOM Fig. 11). Although limited in scope [71], such an analysis provides some explanation about the inner workings of the Transformer models without needing large sets of experimental annotations (labels). On top, the resulting *interpretations of the AI/ML* might be less biased than experimental annotations. For instance, databases with annotations of protein function such as Swiss-Prot [85] and of protein structure such as PDB [50] are extremely biased by what today's experimental techniques can handle [80], [86], [87].

4.4 Protein LMs top without MSAs

The t-SNE and UMAP analyses suggested that the protein LMs had extracted some level of *understanding of the language of life*. However, prediction is the acid test for understanding. To pass this test, we extracted the embeddings learned by the protein LMs directly as input to predict aspects of protein structure (per-residue/token-level prediction of secondary structure) and protein function (per-protein/sentence-level prediction of localization and membrane/other). Overall, the results obtained from the second step of using embeddings from LMs as input to supervised models confirmed [32] that evolutionary information (EI, i.e. methods using multiple sequence alignments MSAs) scientifically and statistically significantly outperformed most LMs not using such information except for ProtT5-XL (on all per-residue and per-protein tasks, Figs. 5, 8 and SOM Tables 9, 8). ProtT5-XL eliminated this gap from embeddings-only input: on some tasks/data sets, it outperformed the current state-of-the-art MSA-based method, on others it remained

slightly behind. Newer protein LMs using *context* improved over both previous LM-based approaches [32] (8-9 percentage points in Q3), other transformers [72] (2-4 percentage points in Q3), and over non-contextualized word2vec-type approaches [88], [89], [90] (18-22 percentage points in Q3). The performance ranges for using two different data sets (CASP12 and NEW364) highlight a different problem. While it is clear that we need to redundancy-reduce evaluations sets with respect to themselves and all data used for development, it is less clear how to exactly do this. In focusing on CASP12 and NEW364, we approached two different assumption. CASP12 is best described as measuring how well predictions will be for proteins with very different structures. A comprehensive rigorous realization of data sets following this perspective has recently been published [91]. NEW364, on the other hand, builds on the assumption that the maximal redundancy is defined by *sequence similar to protein in the PDB*. In this sense, we interpreted results for CASP12 as a lower and those for NEW364 as an upper limit. Either way, the most important task is to constantly create up-to-date sets with enough non-redundant proteins never used for development by any of the methods assessed.

Protein LMs so powerful that even simple baseline are effective: While pre-training is computationally demanding, training supervised models using embeddings as input requires much fewer resources. For instance, the logistic regression trained on top of ProtBERT-BFD was already competitive with substantially more complex CNNs or LSTMs in predicting secondary structure (SOM Table 7). In another example, a parameter-free nearest neighbor lookup using distances from protein LM embeddings sufficed to outperform homology based inference for predicting protein function [92]. This suggested that protein LMs are particularly suitable when the experimental annotations (labels) available are very limited and hinder training of large supervised networks. In fact, none of the supervised solutions presented here that reached the SOA came anywhere near in complexity (number of free parameters) to that of the EI-based methods they reached. Here, we focused on the development of protein LMs and used performance on supervised tasks primarily as a proof of principle without optimizing particular supervised solutions. Others have already begun beating EI-based methods not using protein LMs by custom-designing such solutions [83] possibly even through end-to-end systems [30], [38]. Combinations of evolutionary information and embeddings might bring the most accurate methods. However, such a merger would sacrifice the advantage of protein LMs relying only on single protein sequences which is crucial for large scale analysis as well as certain tasks like single amino acid variant (SAV) effect prediction.

Bi-directionality crucial for protein LMs: In NLP uni-directional (auto-regressive) and bi-directional (auto-encoding) models perform *on par* [12], [93]. In contrast, the bi-directional context appeared crucial to model aspects of the language of life. While auto-encoding models such as Albert [58] utilize context to both sides during loss calculation, auto-regressive models such as TransformerXL [59] consider only context to one side. Performance increased substantially from uni-directional ProtTXL to bi-directional ProtXLNet (Fig. 5, Table 3, and SOM Table 8). This might be compensated for by first pre-training on sequences and their reverse and then concatenating the output of uni-directional LMs applied on both directions. While this does not allow the LM to use bi-directional context during training, it allows supervised networks to combine context derived independently from both sides. For instance, ELMo [8] concatenates the embeddings derived from a forward and a backward LSTM. The protein LM version of ELMo (SeqVec) outperformed the uni-directional ProtTXL but not the bi-directional ProtXLNet. The difference in model size (SeqVec=93M vs. ProtXLNet=409M) and in pre-training data (SeqVec=30M vs. ProtAlbert=224M) might explain some of this effect. Nevertheless, pure uni-directionality as used in TransformerXL appeared detrimental for modeling protein sequences.

4.5 Have protein LMs reached a ceiling?

Applying techniques from NLP to proteins opens new opportunities to extract information from proteins in a self-supervised, data-driven way. New protein representations may complement existing solutions, most successful when combining evolutionary information[8] and machine learning [21], [22], [40], [94]. Here we showed for the first time that embeddings from protein LMs input to relatively simple supervised learning models can reach similar levels of performance without using EI and without optimizing the supervised training pipeline much. However, the gain in inference speed for protein LMs compared to traditional models using evolutionary information is so significant that large-scale predictions become, for the first time since 30 years, feasible on commodity hardware. For instance, the best-performing model ProtT5-XL-U50 can run on a Nvidia TitanV with 12GB vRAM (see Methods for details). Nevertheless, given the experiments described here and in previous work [32], [33], [34], [35], [36], [37], [39], we might expect an upper limit for what protein LMs can learn when using masked language modeling (or auto-regressive pre-training) exclusively. Although this work explicitly addressed the possibility of reaching such a limit, we could not conclusively provide an answer. We could establish three findings. (1) Less noisy and less redundant corpora (e.g. UniRef50) improved over larger but more noisy and redundant corpora (e.g. BFD). (2) In our perspective of limited resources, it was most important to use the resources for long-enough training because the number of samples seen during pre-training correlated with the prediction performance of downstream tasks. Ultimately, this seemed to originate from a trade-off between sufficient model size and sample throughput. (3) The bi-directional outperformed the uni-directional models tested. However, given the advances of protein LMs over the course of the reviewing of this work, we have seen no evidence for having reached a limit for protein LMs, yet.

Many open questions: Answers to the following questions might advance the status-quo. **(1)** Would the addition of auxiliary tasks such as next-sentence or sentence-order

8. Throughout this work, we used evolutionary information (EI) as synonymous for *using multiple sequence alignments (MSAs)*. Whether protein LMs do not implicitly extract EI will have to be proven in separate publications.

prediction offered by BERT or Albert suit protein sequences? A suggestion might be the usage of structure information [95] or evolutionary relationship [35], [96]. **(2)** Might the efficiency of transformers protein LM training improve through sparse transformers [97] or attention optimized with locality-sensitive hashing (LSH) [98] as introduced recently by the Reformer model [99] or more recent work of linear transformers [100]? **(3)** Which data set prepossessing, reduction and training batch sampling should optimally used for better results? **(4)** How much will it improve to tailor the supervised training pipeline to particular tasks? We treated secondary structure or localization prediction more as proxies to showcase the success of protein LMs than as an independent end. **(5)** Will the combination of EI and AI [96] bring the best protein predictions of the future, or will the advantages of single-protein predictions (speed, precision) win out? In fact, single-protein predictions also have the advantage of being more precise in that they do not provide *some implicit average over a protein family*.

Overall, our results established that the combination of HPC solutions for training protein LMs and subsequent training of supervised prediction methods scaled up to the largest data sets ever used in the field. Only the combination of these different domains allowed us to demonstrate that protein LMs can reach up to the same performance of the state-of-the-art of methods combining EI and AI without ever exploiting multiple sequence alignments.

5 Conclusion

Here, we introduced many novel protein language models (LMs) and proved that embeddings extracted from the last LM layers captured constraints relevant for protein structure and function. Although neither the usage of the largest ever database for a protein LMs (BFD), nor that of very large models generated the most informative embeddings, pre-training sufficiently long on considerable diversity made a difference, and more recent LMs performed best. Using embeddings as exclusive input to relatively small-size CNN/FNN models without much optimization yielded methods that appeared competitive in predicting secondary structure, localization and in classifying proteins into membrane/other. In fact, for the first time, new small-size supervised solutions based on LMs embedding input reached levels of performance challenging the state-of-the-art (SOA) methods based on multiple sequence alignment (MSA) input. In contrast, the models presented here never used MSAs. This could save immense expenses when routinely applying embedding-based protein predictions to large data sets, but it also opens a path toward protein-specific rather than family-averaged predictions. Ultimately, joining the strengths of three different, yet complementary, fields (HPC, NLP and computational biology) affected the advance. Self-supervised pre-training combined with transfer-learning tapped into the gold-mines of unlabeled data opening the door for completely novel perspectives (and solutions) on existing problems.

6 Availability

We made all protein LMs trained here publicly available at our ProtTrans repository "https://github.com/agemagician/ProtTrans/". This repository also holds jupyter python notebooks with various tutorials, e.g., on how to extract embeddings or visualize attention using freely available online resources (Google Colab).

Acknowledgments

The authors thank primarily Tim Karl (TUM) and Jian Kong (TUM) for invaluable help with hard- and software; Inga Weise and Aline Schmidt (both TUM) for support with many other aspects of this work; Florian Matthes (TUM) for his generous support and encouragement. Thanks for crucial support and feedback from NVIDIA, in particular to Ulrich Michaelis, Ada Sedova, Geetika Gupta, Axel Koehler, Frederic Pariente, Jonathan Lefman, and Thomas Bradley. Thanks to many at ORNL without whom no aspect of this work could have been realized; particular thanks to John Gounley, Hong-Jun Yoon, Georgia Tourassi, Bill, Brian, Junqi, Graham and Verónica (ORNL Summit). Furthermore, special thanks to Jack Wells (ORNL) for opening the door to kicking off this project. From IBM, we thank Nicolas Castet and Bryant Nelson for their help to fix issues and enhance the performance of IBM PowerAI. From Google, we are deeply grateful to Jamie Kinney, Alex Schroeder, Nicole DeSantis, Andrew Stein, Vishal Mishra, Eleazar Ortiz, Nora Limbourg, Cristian Mezzanotte and all TFRC Team for helping to setup a project on Google Cloud and solving Google cloud issues. No ProtTrans model were easily publicly available without support from the Hugging Face team; including Patrick von Platen, Julien Chaumond, and Clement Delangue. Special thanks to Konstantin Schütze for helping with grant writing and providing early results for the structure prediction task. Furthermore, thanks to both Adam Roberts and Colin Raffel for help with the T5 model. We are grateful to the editor and the anonymous reviewers for essential criticism, especially, for suggesting to compare t-SNEs to randomly initialized models.

This work was supported by a grant from Software Campus 2.0 (TUM) through the German Ministry for Research and Education (BMBF), a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF), and by a grant from the Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA with the donation of 2 Titan GPUs used for the development phase. We also thank the Leibniz Rechenzentrum (LRZ) for providing access to DGX-1(V100) for the testing phase. Martin Steinegger acknowledges support from the National Research Foundation of Korea grant [2019R1A6A1A10073437, NRF-2020M3A9G7103933]; New Faculty Startup Fund and the Creative-Pioneering Researchers Program through Seoul National University.

Last not least, this research used resources of the Oak Ridge National Laboratory (ORNL) Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725, and resources of TPU pods under TensorFlow Research Cloud grant. Furthermore, the Rostlab gladly acknowledges support from Google Cloud and Google Cloud Research Credits program to fund this project under Covid19 HPC Consortium grant.

REFERENCES

[1] J. Wells, B. Bland *et al.*, "Announcing Supercomputer Summit," Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States), Tech. Rep., Jun. 2016.

[2] N. P. Jouppi, C. Young *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. Toronto, ON, Canada: Association for Computing Machinery, Jun. 2017, pp. 1–12.

[3] M. Abadi, A. Agarwal *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467 [cs]*, Mar. 2016.

[4] A. Paszke, S. Gross *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle *et al.*, Eds. Curran Associates, Inc., 2019, pp. 8026–8037.

[5] D. Kirk, "NVIDIA cuda software and gpu parallel computing architecture," in *Proceedings of the 6th International Symposium on Memory Management*, ser. ISMM '07. Montreal, Quebec, Canada: Association for Computing Machinery, Oct. 2007, pp. 103–104.

[6] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," *arXiv:1802.05799 [cs, stat]*, Feb. 2018.

[7] M. Cho, U. Finkler *et al.*, "PowerAI DDL," *arXiv:1708.02188 [cs]*, Aug. 2017.

[8] M. E. Peters, M. Neumann *et al.*, "Deep contextualized word representations," *arXiv:1802.05365 [cs]*, Mar. 2018.

[9] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," *arXiv:1801.06146 [cs, stat]*, May 2018.

[10] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[11] Z. Yang, Z. Dai *et al.*, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," *arXiv:1906.08237 [cs]*, Jan. 2020.

[12] M. Shoeybi, M. Patwary *et al.*, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," *arXiv:1909.08053 [cs]*, Mar. 2020.

[13] C. B. Anfinsen and E. Haber, "Studies on the reduction and reformation of protein disulfide bonds," *Journal of Biological Chemistry*, vol. 236, no. 5, pp. 1361–1363, 1961.

[14] B. Rost and C. Sander, "Bridging the protein sequence-structure gap by structure predictions," *Annual Review of Biophysics and Biomolecular Structure*, vol. 25, pp. 113–136, 1996.

[15] M. S. Klausen, M. C. Jespersen *et al.*, "NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 6, pp. 520–527, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.25674.

[16] J. J. Almagro Armenteros, C. K. Sønderby *et al.*, "DeepLoc: Prediction of protein subcellular localization using deep learning," *Bioinformatics*, vol. 33, no. 21, pp. 3387–3395, Nov. 2017.

[17] J. Yang, I. Anishchenko *et al.*, "Improved protein structure prediction using predicted interresidue orientations," *Proceedings of the National Academy of Sciences*, vol. 117, no. 3, pp. 1496–1503, Jan. 2020.

[18] A. Kulandaisamy, J. Zaucha *et al.*, "Pred-MutHTP: Prediction of disease-causing and neutral mutations in human transmembrane proteins," *Human Mutation*, vol. 41, no. 3, pp. 581–590, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.23961.

[19] M. Schelling, T. A. Hopf, and B. Rost, "Evolutionary couplings and sequence variation effect predict protein binding sites," *Proteins: Structure, Function, and Bioinformatics*, vol. 86, no. 10, pp. 1064–1074, 2018, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.25585.

[20] M. Bernhofer, E. Kloppmann *et al.*, "TMSEG: Novel prediction of transmembrane helices," *Proteins: Structure, Function, and Bioinformatics*, vol. 84, no. 11, pp. 1706–1716, 2016, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.25155.

[21] B. Rost and C. Sander, "Improved prediction of protein secondary structure by use of sequence profiles and neural networks," *Proceedings of the National Academy of Sciences*, vol. 90, pp. 7558–7562, 1993.

[22] ——, "Prediction of protein secondary structure at better than 70% accuracy," *Journal of Molecular Biology*, vol. 232, pp. 584–599, 1993.

[23] T. U. Consortium, "UniProt: A worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, no. D1, pp. D506–D515, Jan. 2019.

[24] M. Steinegger, M. Mirdita, and J. Söding, "Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold," *Nature methods*, vol. 16, no. 7, pp. 603–606, 2019.

[25] M. Bernhofer, C. Dallago *et al.*, "Predictprotein-predicting protein structure and function for 29 years," *Nucleic Acids Research*, 2021.

[26] P. Radivojac, Z. Obradovic *et al.*, "Protein flexibility and intrinsic disorder," *Protein Science*, vol. 13, no. 1, pp. 71–80, 2004, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1110/ps.03128904.

[27] N. Perdigão, J. Heinrich *et al.*, "Unexpected features of the dark proteome," *Proceedings of the National Academy of Sciences*, vol. 112, no. 52, pp. 15 898–15 903, 2015.

[28] T. A. Hopf, L. J. Colwell *et al.*, "Three-dimensional structures of membrane proteins from genomic sequencing," *Cell*, vol. 149, no. 7, pp. 1607–1621, 2012.

[29] B. Rost and A. Valencia, "Pitfalls of protein sequence analysis," *Current Opinion in Biotechnology*, vol. 7, no. 4, pp. 457–461, 1996.

[30] J. John, E. Richard *et al.*, "High accuracy protein structure prediction using deep learning," in *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 2020. [Online]. Available: $https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf$

[31] J. Ingraham, V. Garg *et al.*, "Generative Models for Graph-Based Protein Design," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle *et al.*, Eds. Curran Associates, Inc., 2019, pp. 15 820–15 831.

[32] M. Heinzinger, A. Elnaggar *et al.*, "Modeling aspects of the language of life through transfer-learning protein sequences," *BMC Bioinformatics*, vol. 20, no. 1, p. 723, Dec. 2019.

[33] E. C. Alley, G. Khimulya *et al.*, "Unified rational protein engineering with sequence-based deep representation learning," *Nature Methods*, vol. 16, no. 12, pp. 1315–1322, Dec. 2019.

[34] A. Madani, B. McCann *et al.*, "ProGen: Language Modeling for Protein Generation," *bioRxiv*, p. 2020.03.07.982272, Mar. 2020.

[35] S. Min, S. Park *et al.*, "Pre-Training of Deep Bidirectional Protein Sequence Representations with Structural Information," *arXiv:1912.05625 [cs, q-bio, stat]*, Feb. 2020.

[36] R. Rao, N. Bhattacharya *et al.*, "Evaluating Protein Transfer Learning with TAPE," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle *et al.*, Eds. Curran Associates, Inc., 2019, pp. 9689–9701.

[37] J. J. A. Armenteros, A. R. Johansen *et al.*, "Language modelling for biological sequences – curated datasets and baselines," *bioRxiv*, p. 2020.03.09.983585, Mar. 2020.

[38] M. AlQuraishi, "End-to-End Differentiable Learning of Protein Structure," *Cell Systems*, vol. 8, no. 4, pp. 292–301.e3, Apr. 2019.

[39] A. Rives, S. Goyal *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *bioRxiv*, p. 622803, May 2019.

[40] B. Rost and C. Sander, "Combining evolutionary information and neural networks to predict protein secondary structure," *Proteins: Structure, Function, and Genetics*, vol. 19, pp. 55–72, 1994.

[41] B. E. Suzek, Y. Wang *et al.*, "UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics*, vol. 31, no. 6, pp. 926–932, Mar. 2015.

[42] M. Steinegger and J. Söding, "Clustering huge protein sequence sets in linear time," *Nature Communications*, vol. 9, no. 1, pp. 1–8, Jun. 2018.

[43] E. Asgari, A. C. McHardy, and M. R. Mofrad, "Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (dimotif) and sequence embedding (protvecx)," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[44] L. Coin, A. Bateman, and R. Durbin, "Enhanced protein domain discovery by using language modeling techniques from speech recognition," *Proceedings of the National Academy of Sciences*, vol. 100, no. 8, pp. 4516–4520, 2003.

[45] C. Chelba, T. Mikolov *et al.*, "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling," *arXiv:1312.3005 [cs]*, Mar. 2014.

[46] M. M. Lin and A. H. Zewail, "Hydrophobic forces and the length limit of foldable protein domains," *Proceedings of the National Academy of Sciences*, vol. 109, no. 25, pp. 9851–9856, 2012.

[47] Y. Yang, J. Gao *et al.*, "Sixty-five years of the long march in protein secondary structure prediction: The final stretch?" *Briefings in bioinformatics*, vol. 19, no. 3, pp. 482–494, 2018.

[48] J. A. Cuff and G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction,"

Proteins: Structure, Function, and Bioinformatics, vol. 34, no. 4, pp. 508–519, 1999.

[49] L. A. Abriata, G. E. Tamò *et al.*, "Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods," *Proteins: Structure, Function, and Bioinformatics*, vol. 86, pp. 97–112, 2018.

[50] H. M. Berman, J. Westbrook *et al.*, "The Protein Data Bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, Jan. 2000.

[51] M. Steinegger and J. Söding, "Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets," *Nature biotechnology*, vol. 35, no. 11, pp. 1026–1028, 2017.

[52] G. Wang and R. L. Dunbrack Jr, "PISCES: A protein sequence culling server," *Bioinformatics*, vol. 19, no. 12, pp. 1589–1591, 2003.

[53] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[54] J.-M. Chandonia, N. K. Fox, and S. E. Brenner, "SCOPe: Classification of large macromolecular structures in the structural classification of proteins—extended database," *Nucleic Acids Research*, vol. 47, no. D1, pp. D475–D481, Jan. 2019.

[55] C. Raffel, N. Shazeer *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[56] K. Clark, M.-T. Luong *et al.*, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[57] J. Devlin, M.-W. Chang *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019.

[58] Z. Lan, M. Chen *et al.*, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *arXiv:1909.11942 [cs]*, Feb. 2020.

[59] Z. Dai, Z. Yang *et al.*, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," *arXiv:1901.02860 [cs, stat]*, Jun. 2019.

[60] A. Nambiar, M. E. Heflin *et al.*, "Transforming the language of life: Transformer neural networks for protein prediction tasks," *BioRxiv*, 2020.

[61] Y. You, J. Li *et al.*, "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," in *International Conference on Learning Representations*, Sep. 2019.

[62] F. Limited, "Press release announcing Supercomputer Fugaku," RIKEN, Tech. Rep., Dec. 2019.

[63] N. Hammer, F. Jamitzky *et al.*, "Extreme Scale-out SuperMUC Phase 2 - lessons learned," *arXiv:1609.01507 [astro-ph, physics:physics]*, Sep. 2016.

[64] "Google TPU," https://cloud.google.com/tpu/docs/system-architecture, Jun. 2020.

[65] "Nvidia Apex," https://github.com/NVIDIA/apex, Mar. 2020.

[66] T. D. Le, H. Imai *et al.*, "TFLMS: Large Model Support in TensorFlow by Graph Rewriting," *arXiv:1807.02037 [cs, stat]*, Oct. 2019.

[67] J. Feng and D. Huang, "Optimal Gradient Checkpoint Search for Arbitrary Computation Graphs," *arXiv:1808.00079 [cs, stat]*, Sep. 2019.

[68] K. Jarrett, K. Kavukcuoglu *et al.*, "What is the best multi-stage architecture for object recognition?" in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 2146–2153.

[69] A. Bairoch, "The ENZYME database in 2000," *Nucleic acids research*, vol. 28, no. 1, pp. 304–305, 2000.

[70] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv:1409.0473 [cs, stat]*, May 2016.

[71] S. Vashishth, S. Upadhyay *et al.*, "Attention interpretability across nlp tasks," *arXiv preprint arXiv:1909.11218*, 2019.

[72] R. M. Rao, J. Meier *et al.*, "Transformer protein language models are unsupervised structure learners," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.12.15.422761v1

[73] J. Vig, "A multiscale visualization of attention in the transformer model," 2019.

[74] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks." *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10 915–10 919, Nov. 1992. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.89.22.10915

[75] A. Drozdetskiy, C. Cole *et al.*, "Jpred4: a protein secondary structure prediction server," *Nucleic acids research*, vol. 43, no. W1, pp. W389–W394, 2015.

[76] S. Wang, W. Li *et al.*, "Raptorx-property: a web server for protein structure property prediction," *Nucleic acids research*, vol. 44, no. W1, pp. W430–W435, 2016.

[77] S. Wang, J. Peng *et al.*, "Protein secondary structure prediction using deep convolutional neural fields," *Scientific reports*, vol. 6, p. 18962, 2016.

[78] R. Heffernan, Y. Yang *et al.*, "Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility," *Bioinformatics*, vol. 33, no. 18, pp. 2842–2849, 2017.

[79] D. S. Marks, L. J. Colwell *et al.*, "Protein 3D Structure Computed from Evolutionary Sequence Variation," *PLOS ONE*, vol. 6, no. 12, p. e28766, Dec. 2011.

[80] B. H. Dessailly, R. Nair *et al.*, "Psi-2: structural genomics to cover protein domain family space," *Structure*, vol. 17, no. 6, pp. 869–881, 2009.

[81] T. Kosciolek and D. T. Jones, "Accurate contact predictions using covariation techniques and machine learning," *Proteins: Structure, Function, and Bioinformatics*, vol. 84, pp. 145–151, 2016.

[82] T. Goldberg, T. Hamp, and B. Rost, "LocTree2 predicts localization for all domains of life," *Bioinformatics*, vol. 28, no. 18, pp. i458–i465, Sep. 2012.

[83] H. Stärk, C. Dallago *et al.*, "Light attention predicts protein location from the language of life," *bioRxiv*, 2021.

[84] T. B. Brown, B. Mann *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[85] A. Bairoch and B. Boeckmann, "The swiss-prot protein sequence data bank," *Nucleic acids research*, vol. 19, no. Suppl, p. 2247, 1991.

[86] P. Radivojac, Z. Obradovic *et al.*, "Protein flexibility and intrinsic disorder," *Protein Science*, vol. 13, no. 1, pp. 71–80, 2004.

[87] A. Schafferhans, S. I. O'Donoghue *et al.*, "Dark proteins important for cellular function," *Proteomics*, vol. 18, no. 21-22, p. 1800227, 2018.

[88] T. Mikolov, I. Sutskever *et al.*, "Distributed Representations of Words and Phrases and their Compositionality," *arXiv:1310.4546 [cs, stat]*, Oct. 2013.

[89] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[90] A. Joulin, E. Grave *et al.*, "FastText.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[91] M. AlQuraishi, "Proteinnet: a standardized data set for machine learning of protein structure," *BMC bioinformatics*, vol. 20, no. 1, pp. 1–10, 2019.

[92] M. Littmann, M. Heinzinger *et al.*, "Embeddings from deep learning transfer go annotations beyond homology," *Scientific reports*, vol. 11, no. 1, pp. 1–14, 2021.

[93] S. Rajbhandari, J. Rasley *et al.*, "ZeRO: Memory Optimization Towards Training A Trillion Parameter Models," *arXiv:1910.02054 [cs, stat]*, Oct. 2019.

[94] B. Rost, "PHD: predicting one-dimensional protein structure by profile based neural networks," *Methods in Enzymology*, vol. 266, pp. 525–539, 1996.

[95] T. Bepler and B. Berger, "Learning protein sequence embeddings using information from structure," *arXiv:1902.08661 [cs, q-bio, stat]*, Oct. 2019.

[96] R. Rao, J. Liu *et al.*, "Msa transformer," *bioRxiv*, 2021.

[97] R. Child, S. Gray *et al.*, "Generating Long Sequences with Sparse Transformers," *arXiv:1904.10509 [cs, stat]*, Apr. 2019.

[98] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[99] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," in *International Conference on Learning Representations*, Sep. 2019.

[100] M. Zaheer, G. Guruganesh *et al.*, "Big bird: Transformers for longer sequences," *arXiv preprint arXiv:2007.14062*, 2020.

[101] M. Elrod-Erickson, T. E. Benson, and C. O. Pabo, "High-resolution structures of variant zif268–dna complexes: implications for understanding zinc finger–dna recognition," *Structure*, vol. 6, no. 4, pp. 451–464, 1998.

Ahmed Elnaggar is a PhD candidate at the Technical University of Munich. His main focus of research is self-supervised learning on various modalities (Text, Protein, Source code, Images and speech) using high performance computing.

Michael Heinzinger is a PhD candidate in the Rostlab at TUM in Munich/Garching. His recent research focuses on learning, evaluating and understanding representations for protein sequences from unlabeled data with the goal to empower peers with the computational tools necessary to unravel more fundamental biological truths.

Christian Dallago performs research at the interface of Biology, Machine Learning and Software Engineering with the goal of improving human health through intelligent machines.

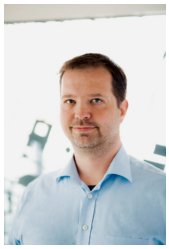
Ghalia Rehawi is a PhD candidate at Helmholtz Zentrum München. She completed her Master of Science (Msc), in the field of informatics, from the Technical University of Munich. She is interested in the application of machine and deep learning techniques in genome and transcriptome analysis.

Yu Wang studied AI at Katholieke Universiteit Leuven in Belgium. He later moved to Munich, Germany to join MIPS, Helmholtz Zentrum München, where he got his Ph.D. in genomics and bioinformatics from Technical University Munich in 2011. He is currently CTO of Med AI Technology (Wu Xi) Ltd., working on transforming healthcare with AI in China.

Llion Jones is a senior software engineer at Google and has been at Google for over 9 years. Started as a YouTube engineer before moving into machine learning. Was on the original team of researchers who developed the now popular Transformer model where he worked on the initial code base and on the attention visualizations.

Tom Gibbs manages Developer Relations for NVIDIA's Supercomputing Business Unit. His primary focus areas are AI for Science, the Convergence of Simulation and Experiment, Quantum Computing and Classical Simulation of High Energy Physics. He has over 40 years of experience in large scale simulation and modeling with an emphasis on grand challenge science problems.

Tamas Feher is an AI developer technology engineer at NVIDIA. His work is focused on accelerating deep learning and machine learning workloads on GPUs. Tamas holds a PhD from the University of Greifswald.



Christoph Angerer is a senior manager within the Autonomous Driving team at NVIDIA. Christoph's team is concerned with designing, implementing, and optimizing AI-based solutions for advanced learning and automation. Christoph holds a PhD from the ETH Zurich, Switzerland.

Martin Steinegger is an Assistant Professor in the biology department at the Seoul National University. His group develops novel computational methods that combine big data algorithms and machine learning to gain insights into unexplored microbial communities.

Debsindhu Bhowmik is a Computational Scientist in the Computational Sciences & Engineering Division and Health Data Sciences Institute at Oak Ridge National Laboratory. His current focus is in understanding complex biological and genetic phenomena and studying disordered systems by implementing new generation large scale simulation blended with Deep learning and scattering techniques.

Burkhard Rost chairs *Comp Biol & Bioinformatics* at TUM Munich. He rooted the leap through combining evolutionary information and machine learning and the launch of *PredictProtein* as first Internet prediction server. Over 30 years, the *Rostlab* contributed influential methods for protein prediction, headed the International Society for Computational Biology (ISCB) and has been dedicated to teaching and raising diversity and gender balance.