ProtTrans: Toward Understanding the Language of Life 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 (LMs) taken from Natural Language Processing (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 protein LMs (pLMs) were trained on the Summit supercomputer using 5616 GPUs and TPU Pod up-to 1024 cores. Dimensionality reduction revealed that the raw pLM-*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: (1) a per-residue (per-token) prediction of protein secondary structure (3-state accuracy Q3=81%-87%); (2) per-protein (pooling) predictions of protein sub-cellular location (ten-state accuracy: Q10=81%) and membrane versus water-soluble (2-state accuracy Q2=91%). For secondary structure, the most informative embeddings (ProtT5) for the first time outperformed the state-of-the-art without multiple sequence alignments (MSAs) or evolutionary information thereby bypassing expensive database searches. Taken together, the results implied that pLMs learned some of the *grammar* of the *language of life*. All our models are available through 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 biochemical 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 these perform specific function(s) (simply put: 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 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 is becoming crucial to bridging these gaps.

Top prediction methods in computational biology [15], [16], [17], [18], [19], [20] combine machine learning (ML)

- *Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, and Burkhard Rost are with the Department of Informatics, Bioinformatics & Computational Biology - i12, Technical University of Munich (TUM), 85748 Garching/Munich, Germany. E-mail: ahmed.elnaggar@tum.de, {mheinzinger, assistant}@rostlab.org, christian@dallago.us, ghalia.rihawi93@gmail.com.*
- *Yu Wang is with the Med AI Technology (Wu Xi) Ltd., Wu Xi, Jiang Su 214000, China. E-mail: wang_yu@hotmail.com.*
- *Llion Jones is with the Google AI, Google, Mountain View, CA 94043 USA. E-mail: llion@google.com.*
- *Tom Gibbs, Tamas Feher, and Christoph Angerer are with the NVIDIA, Santa Clara, CA 95051 USA. E-mail: {tgibbs, tfeher, cangerer}@nvidia.com.*
- *Martin Steinegger is with the School of Biological Sciences, Seoul National University, Seoul 08826, South Korea. E-mail: martin.steinegger@snu.ac.kr.*
- *Debsindhu Bhowmik is with Oak Ridge National Laboratory (ORNL), Oak Ridge, TN 37830 USA. E-mail: bhowmikd@ornl.gov.*

Manuscript received 20 July 2020; revised 3 May 2021; accepted 21 June 2021. Date of publication 7 July 2021; date of current version 9 September 2022. (Corresponding author: Ahmed Elnaggar.)
Recommended for acceptance by C. S. Ong.
Digital Object Identifier no. 10.1109/TPAMI.2021.3095381

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 MSA. Second, feed the EI into the ML through supervised learning of implicit structural or functional constraints. 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. First, when predicting for entire proteomes (all proteins in an organism), compiling the EI becomes computationally costly [25]. Second, EI is not available for all proteins (intrinsically disordered proteins [26] or *dark proteome* [27]). Third, the improvement is best when the EI is most diverse [28], [29]. The latest, and arguably largest leap ever in terms of protein structure prediction, namely AlphaFold2, relies on an advanced combination of EI and ML [30]. Although predicting protein structure at unprecedented levels of accuracy, the method is many orders of magnitude more computationally expensive than the creation of "*minimal*" MSAs.

In analogy to NLP, protein Language Models (pLMs) 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 pLMs extract features directly from single protein sequences, such single-sequence based methods might, for the first time in almost three decades, reach top performance without using EI/MSAs.

In this project, dubbed *ProtTrans*, we pursued two objectives. First, we explored the limits of up-scaling language models trained on proteins as well as protein sequence databases used for training. Second, 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) [31].

2 Methods

Protein Language Models (pLMs) copy the concepts of Language Models from NLP by using as tokens (words in NLP) amino acids from protein sequences, treating entire proteins like sentences in LMs. In *step 1* these pLMs are trained in self-supervised manner, essentially learning to predict masked amino acids (tokens) in already known sequences (Data: 2.1, Method: 2.4). Once trained, we need to establish that the pLMs capture relevant information (Data: 2.2). This first step uses only protein sequences without any annotation as input. In *step 2*, we transfer what the pLMs learned by extracting the embeddings and using them as input for supervised training of per-residue/per-token (secondary structure) and per-protein/pooling (transmembrane proteins and subcellular location) prediction tasks (Data: 2.3, Method: 2.5). The second step uses experimental labels about proteins for the supervised training. Details about Hardware (2.6) and Software (2.7) provide details about implementing pLMs.

TABLE 1
Data Protein LM - UniRef50 and UniRef100 Cluster the UniProt Database at 50 and 100 percent Pairwise Sequence Identity (100 percent Implying That Duplicates are Removed) [32]; BFD Combines UniProt With Metagenomic Data Keeping Only One Copy for Duplicates [24], [33]

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

Units: number of proteins in millions (m), of amino acids in billions (b), and of disk space in GB (uncompressed storage as text).

2.1 Data for Protein Language Models (pLMs)

We measure the impact of data amount on performance through three data sets (Table 1, SOM Fig. 9, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3095381): Uniref50 [32], UniRef100 [32], and BFD (Big Fantastic Database) [24], [33]. Merging UniProt [23] and proteins translated from multiple metagenomic sequencing projects, BFD has become the largest collection of protein sequences: about eight times larger than the largest sets used previously for pLMs [34]. The 8-fold increase in data increased the number of tokens 5-fold (Table 1), as proteins were 1.6-fold longer in UniRef100. 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 [35]) or functional units (domains [36]), 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 [37] top in NLP with about 829m (million) tokens (words), compared to BFD with 393b (billion) tokens (amino acids). Interpreting domains as words, would cut the number of tokens in BFD roughly by a factor of 100 (average domain length [38]) 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.

2.2 Data for Unsupervised Evaluation of Embeddings

We extracted the information captured by the protein LMs through *embeddings*, i.e., vector representations from the last hidden state of the protein LM (Fig. 1) and analyzed it by

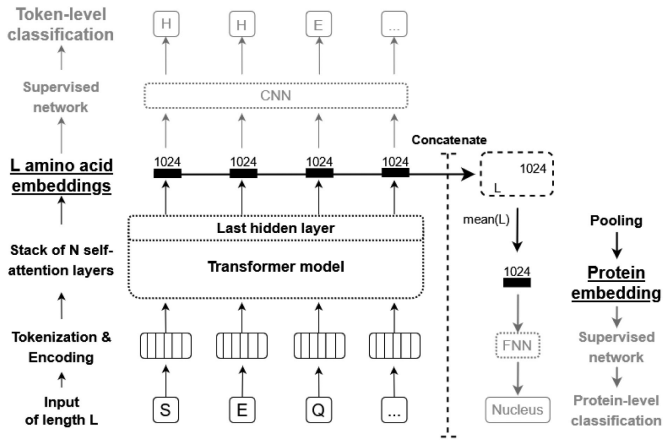

Fig. 1. *Feature extraction overview* - We give a general overview on how ProtTrans models can be used to derive features (embeddings) for arbitrary protein sequences either on the level of single amino acids or whole proteins and how they can be used for classification tasks on both levels. First, an 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.

projecting the high-dimensional representations down to two dimensions (2D) using t-SNE [39]. Toward this end, we took annotations from several sources. First, a non-redundant (PIDE<40%) version of the SCOPe database [40] (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).

2.3 Data for Supervised Training

We also assessed the information captured during self-supervised pre-training of our protein LMs by using embeddings extracted from those models as sole input for 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 4, 3, available online). We also included other public test data sets, namely CB513 [41]), TS115 [42], and CASP12 [43]. 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 [44] with resolutions ≤ 2.5 Å and ≥ 20 residues. MMSeqs2 [45] with highest sensitivity (*-s 7.5*) removed proteins with >20% PIDE to either the training set or to itself. On top, PISCES [46] 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 versus water-soluble and (ii) ten classes of subcellular localization (also referred to as cellular compartments).

2.4 Step 1: Self-Supervised Protein LM Pre-Training

We trained six successful LMs in NLP (T5 [47], Electra [48], BERT [49], Albert [50], Transformer-XL [51] 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 percent). *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 price 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

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

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

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 transferred configurations successfully from NLP to protein sequences [52], [53], [54], 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 percent, 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 [55], 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

1. https://github.com/NVIDIA/DeepLearningExamples/
2. https://github.com/google-research/bert
3. https://github.com/google-research/albert

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 percent 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 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 percent. 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.

4. https://github.com/zihangdai/xlnet
5. https://github.com/google-research/electra
6. https://github.com/google-research/text-to-text-transfer-transformer

2.5 Step 2: Transfer Learning of Supervised Models

In the transfer-learning step, embeddings from our pre-trained protein LMs served as sole input to subsequent supervised training, thereby transferring knowledge acquired during self-supervised pre-training to other tasks. 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 [56]. 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 8, available online), 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 8, available online) 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 [57], Summit [1] or the SuperMUC-NG [58], 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 hardware 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 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 [59].

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

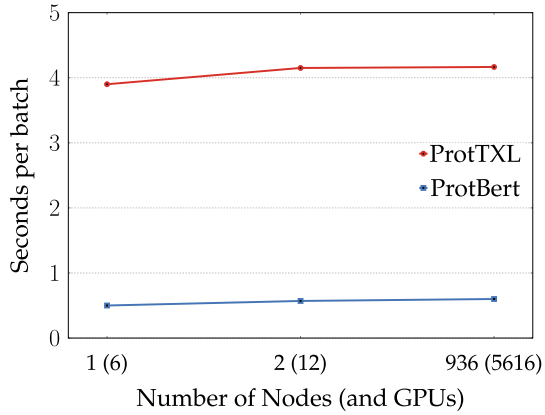

Fig. 2. *Large scale dataset training:* The figure shows the overhead of increasing the number of nodes/gpus for both ProtTXL (blue; low) and ProtBert (red; high). The overhead increases slightly from 1 to 2 nodes but remains constant even when scaling up to 936 nodes with a total of 5616 GPUs. Having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

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 [60] was used for mixed precision training of ProtTXL, because APEX supports pytorch. As ProtTXL training became instable when training with 16 Bit precision, we switched to almost half precision training (more details in SOM - 1.2 Software, available online). 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) [61]. Similar to gradient checkpointing [62], 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. 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 percent for ProtTXL-BFD and to 6.6 percent for ProtBert (SOM Fig. 10a, available online). Additionally, we could increase the batch size by 700 percent for ProtTXL-BFD (SOM Figs. 10b

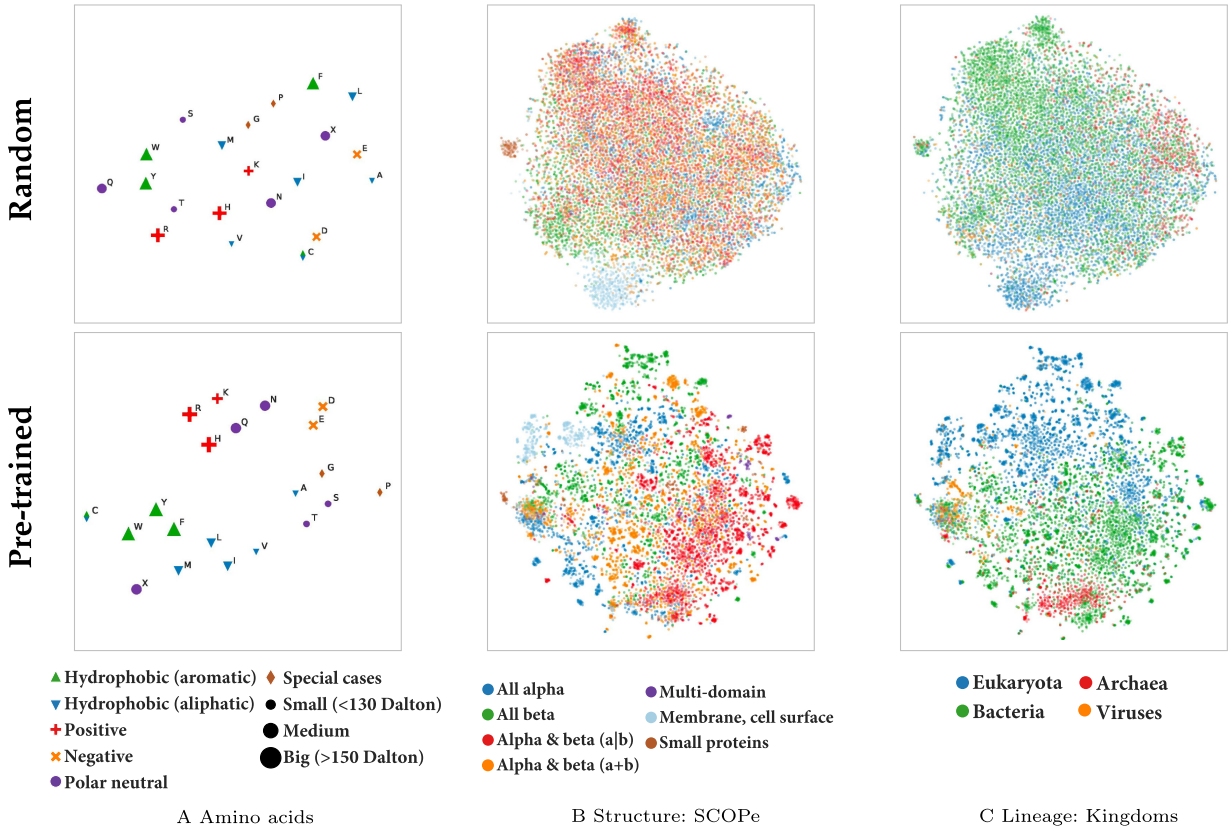

Fig. 3. *Protein LMs learned constraints.* t-SNE projections visualized information extracted by the unsupervised protein LMs (here best-performing ProtT5-XL-U50; upper row: before training (Random), and lower row: after pre-training on BFD & UniRef50. (A) The left-most column highlights single amino 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).

and 10c, available online). The NV-Link between CPU and GPU on Summit-nodes, reduced the training time for ProtTXL by 60 percent while it increased by 72 percent for ProtBert (SOM Fig. 10d, available online).

3 Results

Section 3.1 established that the protein LMs (pLMs) learned to distinguish between different types of proteins by projecting high-dimensional embeddings onto 2D using t-SNE [39]. Section 3.2 showed that pLM embeddings succeeded as exclusive input for supervised training to predict protein features on the per-residue/token-level (secondary structure) and on the per-protein/sentence-level (location and membrane/non-membrane). In fact, predictions based on embeddings from single protein sequences were competitive with, or even surpassed, top methods relying on information from multiple alignments (MSAs). The last Section 3.3 established that the *on par* performance will create substantially lower costs in terms of computing, energy, or CO2 consumption for every future protein prediction.

3.1 Step 1: Protein LMs Learn Without Labels

Embeddings extract constraints about protein function and structure learned by the protein LMs during self-supervised pre-training on raw (unlabeled) protein sequences. t-SNE plots [39] visualized this information by projecting the embeddings onto 2D and by overlaying annotations (labels) of structural, functional or evolutionary features. Using attention maps, we analyzed the DNA-binding zinc-finger motif well conserved in evolution in more detail.

Capturing Biophysical Features of Amino Acids. Applying t-SNE to the uncontextualized token embedding layer visualized information extracted by the pLMs for individual amino acids independent of their context (residues next to it). As previously established for another pLM [53], the t-SNE projections (e.g., ProtT5-XL-U50 SOM Fig. 14a, available online, or ProtBert-BFD SOM Fig. 15a, available online) suggested that all pLMs captured essential biophysical amino acid features, including charge, polarity, size, hydrophobicity, even to the level of aliphatic ([AILMV]) versus 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. 3 A) or some inductive bias of neural networks [63]. The random projection clearly did not carry biophysical information, while the embeddings projection did.

Capturing Protein Structure Classes. We averaged over the length-dimension of the representations derived from the last layer of each pLM (Fig. 1) to derive fixed size representations for each protein and superposed structural class from SCOPe [40]. ProtBert-BFD and especially ProtT5-XL-U50 embeddings visually separated the proteins best (SOM Figs. 14-19, available online). Although sequence length was not explicitly encoded and our pooling squeezed proteins into a fixed vector size, all pLMs separated small from long proteins (brown, e.g., ProtT5-XL-U50 SOM Fig. 14D, available online). All models also distinguished between water-soluble and transmembrane proteins (light blue, e.g., ProtT5-XL-U50 SOM Fig. 14D, available online) and, to some extent, between proteins according to their secondary structure composition(e.g., all-alpha (dark blue) versus all-beta (dark green) ProtT5-XL-U50 Fig. 14D, available online). While having much higher entropy, even the random model clustered small proteins from long proteins (brown, Fig. 3B).

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 above for SCOPe. All pLMs captured some organism-specific aspects (e.g., ProtT5-XL-U50 SOM Fig. 14E, available online). Eukarya and bacteria clustered better than viruses and archaea. Comparing different pLMs revealed the same trend as for protein structure classes: ProtTXL (SOM 19E, available online) and ProtBert (SOM 16E, available online) produced higher entropy clusters, while ProtAlbert (SOM 17E, available online), ProtXLNet (SOM 18E, available online), ProtBERT-BFD (SOM Fig. 15E, available online) and ProtT5-XL-U50 (SOM Fig. 14E, available online) produced 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 [64] and subcellular location (SOM -1.3 pLMs unsupervised, available online). We focused in more detail on the attention mechanism [65] at the core of each Transformer model [10] providing some limited understanding [66], [67]. We visualized [68] the attention weights of ProtAlbert to analyze the structural motif of a zinc-finger binding domain (SOM Fig. 11, available online) crucial for DNA- and RNA-binding and conserved across diverse organisms. Some of the attention heads of ProtAlbert (SOM Fig. 11, available online; line thickness resembles attention weight) focused mostly on the four residues involved in zinc-binding (residues highlighted in the left part of SOM Fig. 11, available online) which is essential for function.

3.2 Step 2: pLMs Competitive as Input to Predict

The acid test to prove that pLM embeddings extracted important constraints is to exclusively use embeddings as input to supervised training. We proxied this through predictions on two different levels, namely on the per-residue or token level (secondary structure) and on the per-protein or sentence level through pooling over entire proteins (location, and classification into membrane/non-membrane proteins). The pLMs 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 pLM (Fig. 1), i.e., pLMs were only used as static feature extractors.

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, we added a novel, non-redundant test set (dubbed *NEW364*). For simplicity, we only presented values for Q3 on CASP12 and NEW364 (TS115 and CB513 contain substantial redundancy; Q8 results brought little novelty; SOM Tables 7, 6, available online). 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 supervised models (Methods): logistic regression (LogReg), FNN, CNN and LSTM. LogReg provided an advanced *baseline* (Q3(LogReg) =74.3-79.3, lower number for set CASP12, upper for set NEW364; SOM Table 5, available online). LSTMs and CNNs performed alike and better than LogReg (Q3(CNN)=76.1-81.1% versus Q3(LSTM)76.1-80.9%). In the following, we focused on the more energy-efficient CNNs.

Comparing pLMs. Trained on UniRef100 (Table 1), ProtBert outperformed other models trained on the same corpus (SOM Tables 7, 6, available online). 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%; SOM Tables 7, 6, available online). 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 those trained only on BFD (Fig. 4, SOM Tables 7, 6, available online). Although these gains were consistently numerically higher, the statistical significance remained within the 68 percent confidence interval (maximal difference: 1.1 percent compared to one standard error of $\pm$0.5%).

Embeddings Reached 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. 4 and SOM Table 6, available online). Both ProtTXL versions fell short compared to an existing ELMo/LSTM-based solution (DeepSeqVec [56]) while all other Transformer-models outperformed DeepSeqVec. Embeddings extracted from another large Transformer (ESMB-1b [67]), improved over all our non-ProtT5 models (Fig. 4 and SOM Table 6, available online). Most solutions using only embeddings as input were outperformed by the top SOA method NetSurfP-2.0 [15] using evolutionary information (Fig. 4 and SOM Tables 7, 6, available online). 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, available online), revealed that 57 percent of the

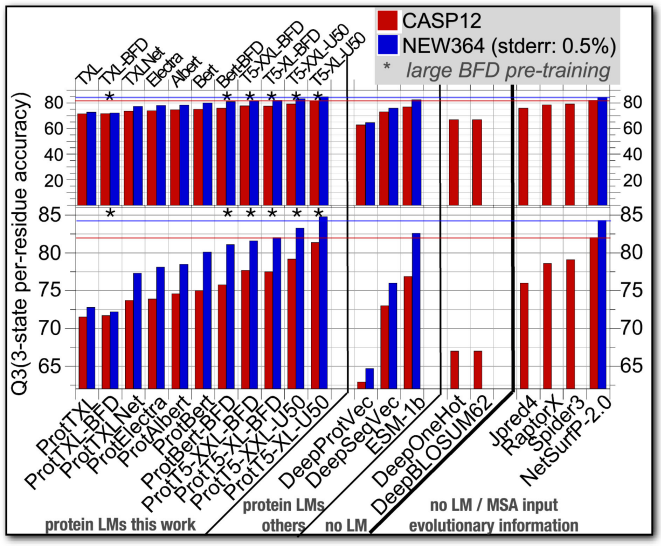

Fig. 4. *Per-residue (token-level) performance for secondary structure prediction: CASP12* (red) and *NEW364* (blue) constitute two test sets. Protein LMs trained here are shown in the left panel of the figure. Additions of *BFD* mark pre-training on the largest database BFD, *U50* mark pre-training with BFD and refining with UniRef50. We included protein LMs described elsewhere (marked as: *protein LMs others*, namely ESM-1b [67], DeepProtVec and DeepSeqVec [56]). All embeddings were input to the same CNN architecture. Two approaches used amino acids instead of embeddings as input (marked as: *no LMs*: DeepOneHot [56] - one-hot encoding - and DeepBLOSUM62 [56] - input BLOSUM62 [69] substitution matrix), as well as, to the current state-of-the-art (SOA) method NetSurfP-2.0 [15], and Jpred4 [70], RaptorX [71], [72], Spider3 [73]. 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.

proteins were predicted with higher Q3 by ProtT5-XL-U50 (CASP12 was too small for such a differential analysis).

pLMs Shone for Small Families. The size of protein families followed the expected power-law/Zipf-distribution (few families have many members, most have fewer [75]). 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 proteins in an MSA clustered at 62 percent PIDE [74], [76]. We analyzed the effect of Neff by comparing NetSurfP-2.0 (using MSAs/EI) to ProtT5-XL-U50 (not using MSAs) through four subsets of NEW364 applying different Neff thresholds, i.e., the subset of proteins without any hit (Neff=1 with 12 proteins), fewer than 10 hits (Neff $\leq$ 10 with 49 proteins) and Neff>10 (314 proteins) (Fig. 5; details on MSA in SOM, available online). ProtT5XL-U50 improved most over NetSurfP-2.0 for the smallest families (Neff=1).

More Samples Better Performance. Despite substantial differences, all pLMs exhibited a similar trend: performance correlated with the number of samples presented to train (*pre-train*). We computed the *number of samples* as the product of the *number of steps* and the *global batch size* (Fig. 6; Spearman's ρ=0.62). In particular, comparing the two largest models (ProtT5-XL and ProtT5-XXL) suggested more pre-training steps to be more beneficial than bigger models.

3.2.2 Per-Protein Location & Membrane Prediction

To investigate per-protein (sentence-level) predictions of protein function, we trained FNNs on subcellular location (also referred to as *cellular compartment*) in ten classes and on the binary classification of membrane versus non-membrane (also referred to as *globular*) proteins. Levels of ten-state (Q10 for location) and two-state (Q2 for membrane/globular) measured performance. Toward this end, we pooled over the entire protein (Fig. 1).

Mean-Pooling Performed Best. Using ProtBERT-BFD embeddings, we compared four different pooling strategies 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 location (Q10) and about three for membrane/other (Q2) compared to the others (mean/*concat*, SOM Table 8, available online). While mean-pooling and *concat* performed similarly for the classification task (membrane/other), mean-pooling outperformed *concat* for location by about ten percentage points (SOM Table 8, available online). In the following, we used only mean-pooling to benchmark the per-protein/sentence-level predictions.

Comparison of pLMs. The per-protein prediction of location largely confirmed the trend observed for per-residue secondary str: All pLMs introduced here (marked by * in SOM Table 9, available online) clearly outperformed the uncontextualized word2vec-based approaches (DeepProtVec; Fig. 7, SOM Table 9, available online). Except for ProtTXL and ProtXLNet, all transformers trained here outperformed the previous ELMo/LSTM-based solution (DeepSeqVec). Increasing the corpus for pre-training the pLMs 10-fold appeared inconsequential (Prot* versus Prot*-BFD in Fig. 7 and SOM Table 9, available online). In contrast, fine-tuning ProtT5 models already trained on BFD using UniRef50 improved (Prot*/Prot*-BFD versus Prot*-U50 in Fig. 7 and SOM Table 9, available online). Although most embedding-based approaches were outperformed by the *SOA* using MSAs (*DeepLoc*), both best ProtT5 models outperformed DeepLoc without MSAs: Q10, Fig. 7 and SOM Table 9, available online.

Similar for Membrane/Other. Results for the classification into membrane/other (Q2; SOM Table 9, available online), largely confirmed those obtained for location (Q10) and secondary structure (Q3/Q8): (1) ProtT5 pLMs fine-tuned on UniRef50 performed best without MSAs, (2) the 10-fold larger pre-training BFD had no noticeable effect, (3) our best pLMs outperformed existing transformer pLMs (ESM-1b) (Fig. 7). In contrast to location and secondary structure, additionally pre-training on UniRef50 appeared not to increase performance (SOM Table 9, available online) 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/MSA-based prediction methods and pLMs by generating MSAs and embeddings for each protein in the human proteome (20,353 proteins with median length of 415). We used the fastest method available, namely MMseqs2 [45],

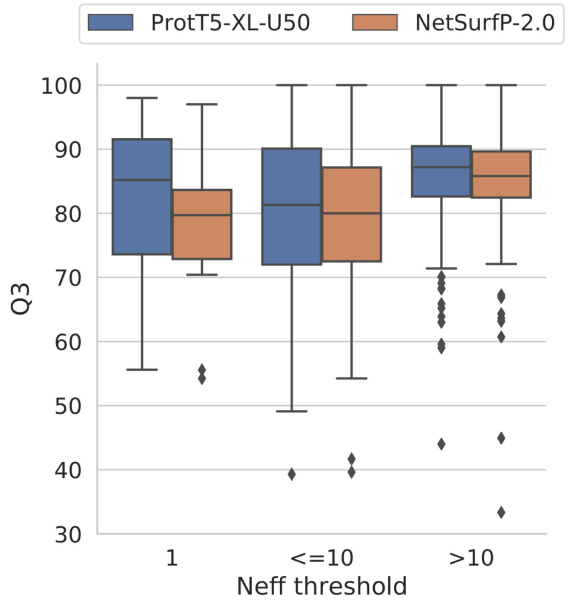

Fig. 5. *Effect of MSA size.* We used our new test set (NEW364) to analyze the effect of the size of an MSA upon secondary structure prediction (Q3) for the two top methods (both reaching Q3=84.3%): NetSurfP-2.0 (using MSA) and ProtT5-XL-U50 (not using MSA). As proxy for MSA size served Neff, the number of effective sequences [74] (clustered at 62 percent 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.

with parameters established by NetSurfP-2.0 to generate MSAs (SOM for more details, available online). MMseqs2 was about 16 to 28-times slower than the fastest pLMs (ProtElectra and ProtBert), and about 4 to 6-times slower than our best model (ProtT5-XL; Fig. 8. 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 10, available online) on the inference speed of different pLMs. When using a single Nvidia

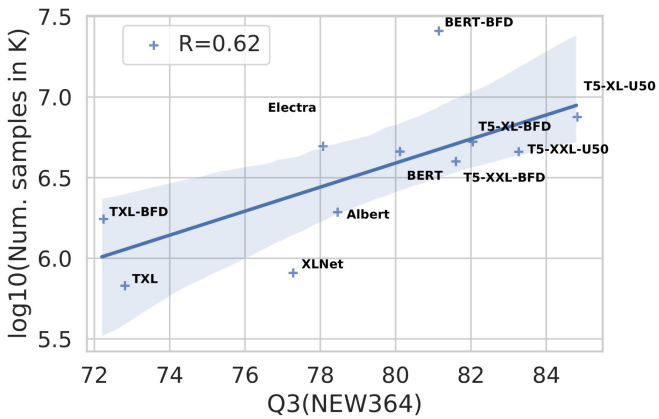

Fig. 6. *Number of pre-training correlated with performance.* We plotted 3-state secondary structure prediction performance (Q3) on NEW364 for all pLMs trained here against the number of samples seen during pre-training (training steps times global batch-size in K). For simplicity, we dropped the *"Prot"* prefix from all models. Despite the vastly different pLMs, the high Spearman's ρ of 0.62 indicated a common trend: more pre-training samples: better prediction.

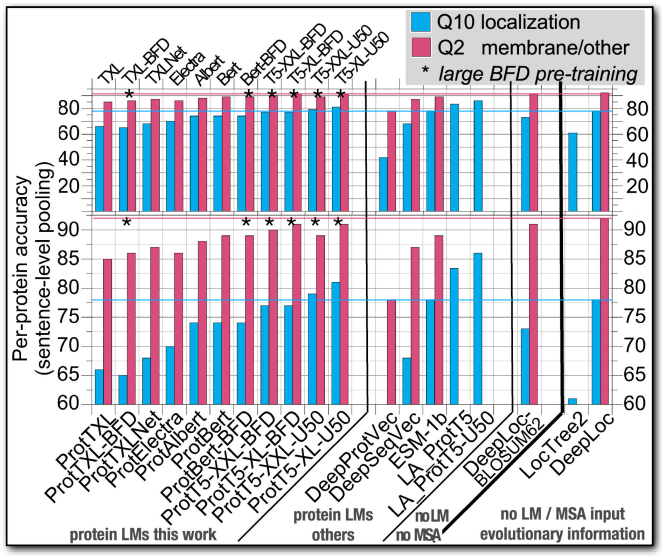

Fig. 7. *Per-protein (sentence-level) performance:* The prediction of location in 10 states (lower bars in cyan: Q10: percentage of proteins with 1 of 10 classes correctly predicted) and the classification of membrane/other (higher bars in magenta: Q2: percentage of proteins correctly classified in either of two classes). Embeddings were derived from pLMs by mean-pooling, i.e., averaging over the length of the entire protein (Fig. 1). Abbreviations as in Table 4 except for one method using neither pLMs 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 [77]. Almost all pLMs 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 location prediction from ProtT5 embeddings through a light-attention mechanism; it clearly outperformed the *SOA* without using MSAs (LA_ProtT5 & LA_ProtT5-U50 [78]). The top row shows the complete range from 0-100, while the lower row zooms into the range of observed differences.

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 pLMs 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 as part of our github repository.

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's DDL [7] and Horovod [6] set the stage for training LMs with billions of free parameters on 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 (pLMs) by fully utilizing 20 percent of Summit for TransformerXL [51], one TPU Pod V3-512 for Bert [49], Electra [48], Albert [50] and XLNet [11],

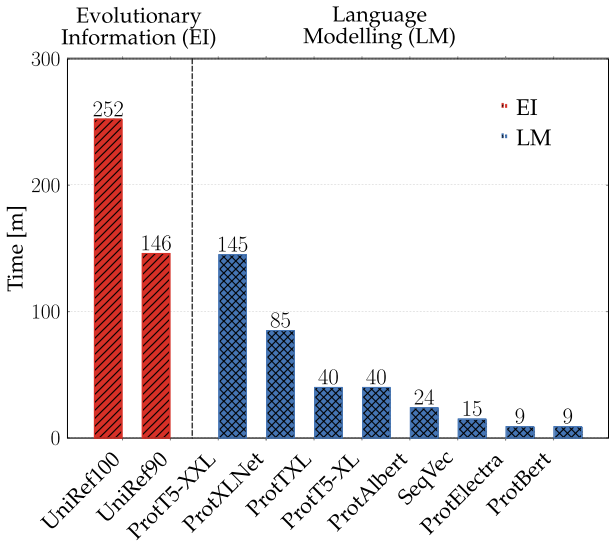

Fig. 8. *Inference speed comparison:* The time required to generate protein representations for the human proteome (20.353 proteins) is compared using either our protein LMs or mmseqs2 (protein sequence search tool [45] used to generate evolutionary information; NetSurfP-2.0 [15] parameters are used). 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).

and a mix of TPU Pod V3-256, V3-512, and V3-1024 for ProtT5-XL and ProtT5-XXL [47]. This implied 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 [55] up to a global batch size of 44K samples (here: protein sequences).

4.2 Training pLMs Longer Most Important

Mostly, we proxied the relevance of information extracted by pre-trained pLMs through the performance of the subsequent supervised tasks trained exclusively on embeddings from pre-trained pLMs. Using pLMs only as static feature extractors, we considered the *pLM to improve (or to be better)* when the supervised task using embeddings from this pLM as input reached higher performance.

BFD - Biggest Data to Pre-Train. We trained our pLMs on the largest protein database ever used for this purpose, namely BFD [33], more than an order of magnitude larger than the standard UniProt [23]. Although bigger did not equate better supervised performance, several pLMs improved through pre-training on more data (UniRef100 versus BFD, Table 1). Nevertheless, the performance increase appeared small given the 10-fold larger data set (e.g., ΔQ3(ProtBert-BFD/UniProt)=1.3%). In contrast, the pre-training protocol by which we first trained on the larger but more noisy (more mistakes in 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%). 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 the wide diversity of pLMs and data sets: the performance of supervised downstream tasks using embeddings from pre-trained pLMs increased with the number of samples presented during pre-training (Fig. 6; 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: training for more steps might require models 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. 4, SOM Tables 6, 7, available online). This finding aligned with results from NLP suggesting that larger models absorb information faster and need less training to achieve similar performance [79]. 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 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 pLMs Learned Global Constraints

Some rudimentary information about how proteins are formed, shaped, and function has been learned by the pLMs during pre-training: all our pLMs (ProtT5, ProtBert, ProtElectra, ProtAlbert, ProtTXL, ProtXLNet) extracted valuable information as revealed by visualizing embeddings without further supervised training on labeled data. The comparisons to randomly initialized pLMs highlighted two other aspects. First, how easy it is to incorrectly imagine patterns when projecting from high-dimensional spaces into 2D: although the random pLMs contained no information, some annotations might have suggested the opposite (Fig. 3 top row untrained). Second, the pLMs extracted important global constraints about protein structure and function (lower row in Fig. 3). These span from the most local (individual token level) biophysical features of amino acid building blocks (e.g., hydrophobicity, charge, and size, Fig. 3 A), over global classifications into structural classes (Fig. 3 B), to the macroscopic domains of life (Fig. 3 C). Global structural (e.g., overall secondary structure content, Fig. 3 B) and biochemical (e.g., transmembrane, SOM Fig. 14B, available online) properties appeared most distinctive. In contrast, local features relying on short motifs were less separated (EC-numbers: Fig. 14F, location: Fig. 14C, available online) 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 Transformer, confirmed the pLMs to have even picked up more subtle signals of short functional motifs. Specifically, a few attention heads of ProtAlbert zoomed into the four residues most important

to coordinate zinc-binding (SOM Fig. 11, available online). Although limited in scope [66], such an analysis provided some explanation about the inner workings of Transformers not 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 SwissProt [80] and of protein structure such as PDB [44] are extremely biased by today's experiments [75], [81], [82].

4.4 pLMs Top Without MSAs - AI Without EI

The t-SNE and UMAP analyses revealed the pLMs to have extracted some *understanding of the language of life*. As prediction is the acid test for understanding, we extracted the pLM embeddings as input to predicting aspects of protein structure. Overall, the results confirmed [56] that evolutionary information (EI, i.e., methods using multiple sequence alignments MSAs) significantly outperformed most pLMs not using such information, except for ProtT5-XL (on all per-residue and per-protein tasks, Figs. 4, 7 and SOM Tables 7, 6, 9, available online). ProtT5-XL eliminated this gap from embeddings-only input: on some tasks/data sets, it outperformed the current state-of-the-art (*SOA*) MSA-based method, on others it remained slightly behind. Newer pLMs using *context* improved over both previous pLM-based approaches [56] (8-9 percentage points in Q3), other transformers [53] (2-4 percentage points in Q3), and over non-contextualized word2vec-type approaches [83], [84] (18-22 percentage points in Q3).

The performance differences between two data sets (CASP12 and NEW364) highlighted another problem. While it is clear that we need to reduce redundancy within the evaluation set and between it and all development sets, it is less clear how to exactly do this. In focusing on CASP12 and NEW364, we approached two different assumptions. CASP12 mostly measures how well predictions will be for proteins with unseen structures. A comprehensive rigorous realization of data sets following this perspective has recently been published [85]. 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. The NEW364 comparisons also highlighted the importance to constantly create up-to-date data sets with enough non-redundant proteins never used for development by any of the methods assessed.

pLMs so Powerful to Render Simple Baseline Effective. While pre-training pLMs is computationally costly, training supervised models using embeddings as input requires much less energy. 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 5, available online). In another example, a parameter-free nearest neighbor lookup using distances from pLM embeddings sufficed to outperform homology-based inference for predicting protein function [86]. This suggested that pLMs are particularly suitable when so few experimental annotations (labels) are available that the complexity of modern deep learning becomes prohibitive. In fact, all supervised solutions presented here that reached the *SOA* were much less complex (fewer free parameters) than the EI-based methods they reached. We focused on the development of pLMs using performance on supervised tasks to rank without optimizing particular supervised solutions. Others have already begun surmounting EI-based methods by custom-designing pLM-based solutions [78] possibly even through end-to-end systems [30], [87]. Such solutions can even compete when MSAs are absolutely essential, e.g., for predicting effects of single amino acid variants (SAVs) [88].

Bi-Directionality Crucial for pLMs. In NLP uni-directional (auto-regressive) and bi-directional (auto-encoding) models perform *on par* [12], [89]. In contrast, the bi-directional context appeared crucial to model aspects of the language of life. While auto-encoding models such as Albert [50] utilize context to both sides during loss calculation, auto-regressive models such as TransformerXL [51] consider only context to one side. Performance increased substantially from uni-directional ProtTXL to bi-directional ProtXLNet (Fig. 4, SOM Tables 6, 7, available online). 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 versus ProtXLNet = 409M) and in pre-training data (SeqVec = 30M versus 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 pLMs Reached a Ceiling?

Short answer: clearly not. For instance, since first submitting this work (July 2020), the mark has been pushed substantially higher. More generally, adopting 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[7] and machine learning [21], [22], [31], [90]. Here we showed for the first time that pLM embeddings 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 pLMs compared to traditional solutions using EI/MSAs 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. Nevertheless, given the pLMs described here and elsewhere [34], [52], [53], [56], [91], [92], [93], we might expect an upper limit for what pLMs can learn

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

through masked language modeling (or auto-regressive pre-training). This work 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 pLMs, 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 prediction offered by BERT or Albert suit protein sequences? A suggestion might be incorporating structure information [94] or evolutionary relations [92], [95]. *(2)* Might the efficiency of training transformer pLMs improve through sparse transformers [96] or attention optimized with locality-sensitive hashing (LSH) [97] as introduced recently by the Reformer model [98] or more recent work of linear Transformers [99]? *(3)* Which data set, pre-processing, redundancy reduction and training batch sampling should be used to improve? *(4)* How much will it improve to tailor the supervised training pipeline to particular tasks? We treated secondary structure or location prediction more as proxies to showcase the success of protein LMs than as an independent end. *(5)* Will the combination of EI and AI [95] 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 conclusively demonstrated that pLMs reach up to or even above the performance of the best methods in the field combining EI and AI without ever exploiting multiple sequence alignments.

5 Conclusion

Here, we introduced many novel protein language models (pLMs) and proved that embeddings extracted from the last pLM layers captured constraints relevant for protein structure and function. Although neither the usage of the largest ever database (BFD) to pre-train pLMs, nor that of very large models generated the most informative embeddings. Instead, pre-training sufficiently long on considerable diversity made a difference, and more recent pLMs performed best. Using embeddings as exclusive input to relatively small-size CNN/FNN models without much optimization yielded methods competitive in predicting secondary structure, sub-cellular location and in classifying proteins into membrane/other. In fact, for the first time, new small-size supervised solutions based on pLM embedding input reached levels of performance challenging the state-of-the-art (SOA) methods based on evolutionary information (EI) taken form multiple sequence alignment (MSA) input. In contrast, the models presented here never used MSAs. This will 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). Additionally, secondary structure predictions for ProtT5-XL-U50 are available through the PredictProtein [25] webserver: "https://predictprotein.org/".

Acknowledgments

The authors would like to thank Tim Karl, TUM, and Jian Kong, TUM, for invaluable help with hard- and software, Inga Weise, TUM, and Aline Schmidt, TUM, for support with many other aspects of this work, Florian Matthes, TUM, for his generous support and encouragement, crucial support and feedback from NVIDIA, in particular to Ulrich Michaelis, Ada Sedova, Geetika Gupta, Axel Koehler, Frederic Pariente, Jonathan Lefman, and Thomas Bradley, and 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). The authors would also like to thank Jack Wells (ORNL) for opening the door to kicking off this project. From IBM, the authors would like to thank Nicolas Castet and Bryant Nelson for their help to fix issues and enhance the performance of IBM PowerAI. From Google, the authors would like to thank 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 was easily publicly available without support from the Hugging Face team, including Patrick von Platen, Julien Chaumond, and Clement Delangue. The authors would also like to thank Konstantin Weißenow for helping with grant writing and providing early results for the structure prediction task. The authors would also like to thank both Adam Roberts and Colin Raffel for help with the T5 model, and the editor and the anonymous reviewers for essential criticism, especially, for suggesting to compare t-SNEs to randomly initialized models. The authors would also like to thank Leibniz Rechenzentrum (LRZ) for providing access

to DGX-1(V100) for the testing phase. This work was supported in part by Software Campus 2.0 (TUM) through the German Ministry for Research and Education (BMBF), in part by the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF), in part by the Deutsche Forschungsgemeinschaft under Grant DFG-GZ: RO1320/4-1, and in part by NVIDIA with the donation of 2 Titan GPUs used for the development phase. The work of Martin Steinegger was supported in part by the National Research Foundation of Korea under Grants 2019R1A6A1A10073437 and NRF-2020M3A9G7103933, in part by the New Faculty Startup Fund and the Creative-Pioneering Researchers Program through Seoul National University. The work of Rostlab was supported in part by Google Cloud and in part by Google Cloud Research Credits Program to fund this project under Covid19 HPC Consortium grant. This work used resources of the Oak Ridge National Laboratory (ORNL) Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Grant DE-AC05-00OR22725, and resources of TPU pods under TensorFlow Research Cloud grant. A. Elnaggar and M. Heinzinger contributed equally to this work.

REFERENCES

[1] J. Wells *et al.*, "Announcing supercomputer summit," Oak Ridge Nat. Lab., Oak Ridge, TNUSA, Tech. Rep., 2016.
[2] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architecture*, Toronto, ON, Canada, Jun. 2017, pp. 1–12.
[3] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.
[4] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[5] D. Kirk, "NVIDIA cuda software and GPU parallel computing architecture," in *Proc. 6th Int. Symp. Memory Manage.*, Montreal, Quebec, Canada, Oct. 2007, pp. 103–104.
[6] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," 2018, *arXiv:1802.05799*.
[7] M. Cho *et al.*, "PowerAI DDL," 2017, *arXiv:1708.02188*.
[8] M. E. Peters *et al.*, "Deep contextualized word representations," 2018, *arXiv:1802.05365*.
[9] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018, *arXiv:1801.06146*.
[10] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[11] Z. Yang *et al.*, "XLNet: Generalized autoregressive pretraining for language understanding," 2020, *arXiv:1906.08237*.
[12] M. Shoeybi *et al.*, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2020, *arXiv:1909.08053*.
[13] C. B. Anfinsen and E. Haber, "Studies on the reduction and reformation of protein disulfide bonds," *J. Biol. Chem.*, vol. 236, no. 5, pp. 1361–1363, 1961.
[14] B. Rost and C. Sander, "Bridging the protein sequence-structure gap by structure predictions," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 25, pp. 113–136, 1996.
[15] M. S. Klausen *et al.*, "NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning," *Proteins Struct. Function Bioinf.*, vol. 87, no. 6, pp. 520–527, 2019.
[16] J. J. A. Armenteros *et al.*, "DeepLoc: Prediction of protein subcellular localization using deep learning," *Bioinformatics*, vol. 33, no. 21, pp. 3387–3395, Nov. 2017.
[17] J. Yang *et al.*, "Improved protein structure prediction using predicted interresidue orientations," *Proc. Nat. Acad. Sci.*, vol. 117, no. 3, pp. 1496–1503, Jan. 2020.
[18] A. Kulandaisamy *et al.*, "Pred-MutHTP: Prediction of disease-causing and neutral mutations in human transmembrane proteins," *Hum. Mutat.*, vol. 41, no. 3, pp. 581–590, 2020.
[19] M. Schelling, T. A. Hopf, and B. Rost, "Evolutionary couplings and sequence variation effect predict protein binding sites," *Proteins Struct. Function Bioinf.*, vol. 86, no. 10, pp. 1064–1074, 2018.
[20] M. Bernhofer *et al.*, "TMSEG: Novel prediction of transmembrane helices," *Proteins Struct. Function Bioinf.*, vol. 84, no. 11, pp. 1706–1716, 2016.
[21] B. Rost and C. Sander, "Improved prediction of protein secondary structure by use of sequence profiles and neural networks," *Proc. Nat. Acad. Sci.*, vol. 90, pp. 7558–7562, 1993.
[22] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *J. Mol. Biol.*, vol. 232, pp. 584–599, 1993.
[23] T. U. Consortium, "UniProt: A worldwide hub of protein knowledge," *Nucleic Acids Res.*, 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," *Nat. Methods*, vol. 16, no. 7, pp. 603–606, 2019.
[25] M. Bernhofer *et al.*, "Predictprotein-predicting protein structure and function for 29 years," *Nucleic Acids Res.*, vol. 49, pp. W535–W540, 2021.
[26] P. Radivojac *et al.*, "Protein flexibility and intrinsic disorder," *Protein Sci.*, vol. 13, no. 1, pp. 71–80, 2004.
[27] N. Perdigão, "Unexpected features of the dark proteome," *Proc. Nat. Acad. Sci.*, vol. 112, no. 52, pp. 15898–15903, 2015.
[28] T. A. Hopf *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," *Curr. Opin. Biotechnol.*, vol. 7, no. 4, pp. 457–461, 1996.
[30] J. John *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] B. Rost and C. Sander, "Combining evolutionary information and neural networks to predict protein secondary structure," *Proteins Struct. Function Genet.*, vol. 19, pp. 55–72, 1994.
[32] B. E. Suzek *et al.*, "UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics*, vol. 31, no. 6, pp. 926–932, Mar. 2015.
[33] M. Steinegger and J. Söding, "Clustering huge protein sequence sets in linear time," *Nat. Commun.*, vol. 9, no. 1, pp. 1–8, Jun. 2018.
[34] A. Madani *et al.*, "ProGen: Language modeling for protein generation," 2020, *arXiv:2004.03497*.
[35] 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)," *Sci. Rep.*, vol. 9, no. 1, pp. 1–16, 2019.
[36] L. Coin, A. Bateman, and R. Durbin, "Enhanced protein domain discovery by using language modeling techniques from speech recognition," *Proc. Nat. Acad. Sci.*, vol. 100, no. 8, pp. 4516–4520, 2003.
[37] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," 2014, *arXiv:1312.3005*.
[38] M. M. Lin and A. H. Zewail, "Hydrophobic forces and the length limit of foldable protein domains," *Proc. Nat. Acad. Sci.*, vol. 109, no. 25, pp. 9851–9856, 2012.
[39] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[40] 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 Res.*, vol. 47, no. D1, pp. D475–D481, Jan. 2019.
[41] Y. Yang, "Sixty-five years of the long march in protein secondary structure prediction: The final stretch?," *Brief. Bioinf.*, vol. 19, no. 3, pp. 482–494, 2018.
[42] J. A. Cuff and G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction," *Proteins Struct. Function Bioinf.*, vol. 34, no. 4, pp. 508–519, 1999.
[43] L. A. Abriata *et al.*, "Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods," *Proteins Struct. Function Bioinf.*, vol. 86, pp. 97–112, 2018.
[44] H. M. Berman *et al.*, "The protein data bank," *Nucleic Acids Res.*, vol. 28, no. 1, pp. 235–242, Jan. 2000.
[45] M. Steinegger and J. Söding, "MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets," *Nat. Biotechnol.*, vol. 35, no. 11, pp. 1026–1028, 2017.

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

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

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

[49] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019, *arXiv:1810.04805*.

[50] Z. Lan *et al.*, "ALBERT: A lite BERT for Self-supervised learning of language representations," 2020, *arXiv:1909.11942*.

[51] Z. Dai *et al.*, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.

[52] R. Rao *et al.*, "Evaluating protein transfer learning with TAPE," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9689–9701.

[53] A. Rives *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," 2019, *bioRxiv:622803*.

[54] A. Nambiar *et al.*, "Transforming the language of life: Transformer neural networks for protein prediction tasks," 2020, *bioRxiv:2020.06.15.153643*.

[55] Y. You *et al.*, "Large batch optimization for deep learning: Training BERT in 76 minutes," 2019, *arXiv:1904.00962*.

[56] M. Heinzinger *et al.*, "Modeling aspects of the language of life through transfer-learning protein sequences," *BMC Bioinf.*, vol. 20, no. 1, Dec. 2019, Art. no. 723.

[57] F. Limited, "Press release announcing Supercomputer Fugaku," RIKEN, Tokyo, Japan, Tech. Rep., Dec. 2019. [Online]. Available: https://www.fujitsu.com/global/about/resources/news/press-releases/2019/1202-01.html

[58] N. Hammer *et al.*, "Extreme scale-out super-MUC phase 2 - lessons learned," 2016, *arXiv:1609.01507*.

[59] Google TPU. Accessed: Jun. 2020. [Online]. Available: https://cloud.google.com/tpu/docs/systemarchitecture

[60] Nvidia Apex. Accessed: Mar. 2020. [Online]. Available: https://github.com/NVIDIA/apex

[61] T. D. Le *et al.*, "TFLMS: Large model support in tensor-flow by graph rewriting," 2019, *arXiv:1807.02037*.

[62] J. Feng and D. Huang, "Optimal gradient checkpoint search for arbitrary computation graphs," 2019, *arXiv:1808.00079*.

[63] K. Jarrett *et al.*, "What is the best multi-stage architecture for object recognition?," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 2146–2153.

[64] A. Bairoch, "The ENZYME database in 2000, " *Nucleic Acids Res.*, vol. 28, no. 1, pp. 304–305, 2000.

[65] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016, *arXiv:1409.0473*.

[66] S. Vashishth *et al.*, "Attention interpretability across NLP tasks," 2019, *arXiv:1909.11218*.

[67] R. M. Rao *et al.*, "Transformer protein language models are unsupervised structure learners," 2020, *bioRxiv:2020.12.15.422761v1*.

[68] J. Vig, "A multiscale visualization of attention in the transformer model," 2019, *arXiv:1906.05714*.

[69] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Nat. Acad. Sci.*, vol. 89, no. 22, pp. 10915–10919, Nov. 1992.

[70] A. Drozdetskiy *et al.*, "Jpred4: A protein secondary structure prediction server," *Nucleic Acids Res.*, vol. 43, no. W1, pp. W389–W394, 2015.

[71] S. Wang *et al.*, "Raptorx-property: A web server for protein structure property prediction," *Nucleic Acids Res.*, vol. 44, no. W1, pp. W430–W435, 2016.

[72] S. Wang *et al.*, "Protein secondary structure prediction using deep convolutional neural fields," *Sci. Rep.*, vol. 6, 2016, Art. no. 18962.

[73] R. Heffernan *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.

[74] D. S. Marks *et al.*, "Protein 3D structure computed from evolutionary sequence variation," *PloS One*, vol. 6, no. 12, Dec. 2011, Art. no. e28766.

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

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

[77] 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.

[78] H. St ärk *et al.*, "Light attention predicts protein location from the language of life," 2021, *bioRxiv:2021.04.25.441334*.

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

[80] A. Bairoch and B. Boeckmann, "The swiss-prot protein sequence data bank," *Nucleic Acids Res.*, vol. 19, no. Suppl, 1991, Art. no. 2247.

[81] P. Radivojac, "Protein flexibility and intrinsic disorder," *Protein Sci.*, vol. 13, no. 1, pp. 71–80, 2004.

[82] A. Schafferhans *et al.*, "Dark proteins important for cellular function," *Proteomics*, vol. 18, pp. 21–22, 2018.

[83] T. Mikolov *et al.*, "Distributed representations of words and phrases and their compositionality," 2013, *arXiv:1310.4546*.

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

[85] M. AlQuraishi, "ProteinNet: A standardized data set for machine learning of protein structure," *BMC Bioinf.*, vol. 20, no. 1, pp. 1–10, 2019.

[86] M. Littmann *et al.*, "Embeddings from deep learning transfer go annotations beyond homology," *Sci. Rep.*, vol. 11, no. 1, pp. 1–14, 2021.

[87] M. AlQuraishi, "End-to-end differentiable learning of protein structure," *Cell Syst.*, vol. 8, no. 4, pp. 292–301, Apr. 2019.

[88] C. Marquet *et al.*, "Embeddings from protein language models predict conservation and variant effects," *Hum. Genetics*, early access, Jun. 3, 2021, doi: 10.21203/rs.3.rs-584804/v1.

[89] S. Rajbhandari *et al.*, "ZeRO: Memory optimization towards training a trillion parameter models," 2019, *arXiv:1910.02054*.

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

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

[92] S. Min *et al.*, "Pre-training of deep bidirectional protein sequence representations with structural information," 2020, *arXiv:1912.05625*.

[93] J. J. A. Armenteros *et al.*, "Language modelling for biological sequences–curated datasets and baselines," 2020, *bioRxiv:2020.03.09.983585*.

[94] T. Bepler and B. Berger, "Learning protein sequence embeddings using information from structure," 2019, *arXiv:1902.08661*.

[95] R. Rao *et al.*, "MSA transformer," 2021, *bioRxiv:2021.02.12.430858*.

[96] R. Child *et al.*, "Generating long sequences with sparse transformers," 2019, *arXiv:1904.10509*.

[97] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. Thirtieth Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.

[98] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," 2019, *arXiv:2001.04451*.

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

[100] M. Elrod-Erickson, T. E. Benson, and C. O. Pabo, "Highresolution 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 currently working toward the PhD degree with the Technical University of Munich. His research interests include self-supervised learning on various modalities, such as text, protein, source code, images, and speech, using high-performance computing.

Michael Heinzinger is currently working toward the PhD degree with Rostlab, TUM, Munich. His 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 on the interface of biology, machine learning, and software engineering, with the goal of improving human health through intelligent machines.

Ghalia Rehawi received the M.Sc. degree in informatics from the Technical University of Munich. She is currently working toward the PhD degree with Helmholtz Zentrum München. Her research interests include the application of machine and deep learning techniques in genome and transcriptome analysis.

Yu Wang received the PhD degree in genomics and bioinformatics from the Technical University of Munich in 2011. He studied AI from Katholieke Universiteit Leuven, Belgium. He later moved to Munich, Germany, and joined MIPS, Helmholtz Zentrum München. He is currently a CTO of Med AI Technology (Wu Xi) Ltd., working on transforming healthcare with AI in China.

Llion Jones is currently a senior software engineer with Google for more than nine years. He was a YouTube engineer before working on machine learning. He was on the original team of researchers who developed the popular transformer model and worked on the initial code base and on the attention visualizations.

Tom Gibbs is currently a developer relations manager with Supercomputing Business Unit, NVIDIA. He has more than 40 years of experience in large scale simulation and modeling, with an emphasis on grand challenge science problems. His research interests include AI for science, the convergence of simulation and experiment, quantum computing, and classical simulation of high-energy physics.

Tamas Feher received the PhD degree from the University of Greifswald. He is currently an AI developer technology engineer with NVIDIA. His research interests include accelerating deep learning and machine learning workloads on GPUs.

Christoph Angerer received the PhD degree from ETH Zurich, Switzerland. He is currently a senior manager with the Autonomous Driving Team, NVIDIA. His team is concerned with designing, implementing, and optimizing AI-based solutions for advanced learning and automation.

Martin Steinegger is currently an assistant professor with Biology Department, 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 currently a computational scientist with the Computational Sciences and Engineering Division and Health Data Sciences Institute, Oak Ridge National Laboratory. His research interests include 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 currently chairs Comp Biol and Bioinformatics, TUM Munich. He rooted the leap through combining evolutionary information and machine learning and the launch of PredictProtein as first Internet prediction server. For more than 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.