Agile Language Transformers for Recombinant Protein Expression Optimization

Jeliazko R. Jeliazkov, Maxim V. Shapovalov, Diego del Alamo, Matt C. Sternke, and Joel D. Karpiak

Protein Design and Informatics, GSK

{jeliazko.r.jeliazkov or joel.x.karpiak}@gsk.com

Abstract

Language Transformers (LaTs) have achieved state-of-the-art performance in a range of challenging protein modeling tasks including structure prediction, design, mutation effect prediction, and others. The lion's share of these improvements derive from exponential increases in the size and depth of these neural networks, which now routinely exceed billions of trainable parameters, rather than fundamental architectural innovations. This explosive growth in model size poses an obstacle to integration into design-build-test cycles, wherein models are iteratively evaluated, retrained, and improved throughout data collection. As a result, large LaTs do not meet the need for lightweight, rapid-to-train models that excel at problems with tight data-model feedback loops. Here, we present a small, 10 million-parameter BERT model with linearly scaling attention that can be trained from scratch on four Nvidia V100 GPUs in under a week and fine-tuned with full back-propagation in hours to days. We demonstrate that this model excels at two challenging active-learning problems, recombinant protein expression prediction and codon optimization, that require interfacing with experiments. Our approach highlights the size-cost tradeoff inherent to LaTs and demonstrates the utility of small, custom-designed models in practical settings.

1 Introduction

The interface of bioinformatics and deep learning has been shaped by public access to massive genetic databases, such as UniProt [1] and MGnify [2], containing millions to billions of naturally occurring amino acid sequences. In recent years, this resource has facilitated the development of Transformer models [3] adapted from the field of natural language processing that excel at a range of challenging scientific tasks, ranging from point mutation effect prediction [4] to structure prediction and design [5]. Recent developments have focused on increasing their size and depth [6], rather than revisions to the architecture responsible for their groundbreaking performance. Modern models routinely exceed one billion trainable parameters and demonstrate both more effective generalization as well as accurate zero-shot performance in adjacent objectives such as function prediction. Improvements in specialized computational resources such as TPUs are certain to further propel massive increases in the size of LaTs. However, such developments put these computational innovations further out of reach of practitioners with limited time, computational resources, or experimental sequence data available for training.

In this manuscript, we demonstrate that small models, trained in under a week using four GPUs and fine-tuned in a matter of days, can provide state-of-the art predictions in a range of specialized tasks. Since large LaTs are designed to be trained on hundreds of millions of wild-type sequences without

Machine Learning for Structural Biology Workshop, NeurIPS 2022.

issue, they are ill-equipped to derive conclusions from smaller, manually curated databases available for training in such settings. Two example problems include recombinant protein expression prediction (in *Escherichia coli*), which determines whether an amino acid sequence will be synthesized and folded, in a cellular setting, in sufficient quantities for purification and downstream assays, and codon optimization, which determines a DNA sequence adherent to constraints, usually to maximize protein expression. LaTs are largely inappropriate for such problems for two reasons. First, the computational complexity and size of the models needlessly raises the cost of fine-tuning. In natural language processing tasks, efforts to address this include distilling models to a minimal set of parameters[7]. Second, reliance on self-attention, which scales quadratically with respect to sequence length, further slows training and inference times. There have been some efforts toward linearizing self-attention in the setting of a protein language model, such as ProteinBERT [8], thus indicating that pursuing these strategies in tandem has the potential to outperform models with up to ten times more parameters.

An ideal model would balance the benefits and tradeoffs of model size when determining which architecture should serve as a foundation for further task-specific fine-tuning. Here, we demonstrate that a custom-designed derivative of ProteinBERT, which at 10 million parameters is orders of magnitude smaller than modern LaTs, can achieve state-of-the-art prediction in two distinct tasks, recombinant protein expression prediction and codon optimization, with only hours to days of training. Instrumental to this rapid training schedule was the refactoring of ProteinBERT to PyTorch [9] and PyTorch Lightning [10], as well as improved accessibility by simplifying its training scheme. Accurate predictions of recombinant protein expression were achieved by topping off the model with two fully-connected layers, whereas codon optimization prediction was enabled by passing the amino acid embeddings obtained from the fine-tuned Transformer model through a separate decoder trained on bacterial genomes. Our results show that full back-propagation through the entire model, made possible by the model's small size, measurably improves accuracy during expression prediction. However, these advantages come at the cost of poorer zero-shot prediction, exemplified by an inability to optimize codons with specific G/C content. Overall, our results demonstrate how the nimbleness of small LaTs make them amenable to rapid model design and testing, thereby providing greater practical value in diverse and challenging tasks.

2 Methods

2.1 Foundational BERT

We re-implemented the network described in [8]. It follows a standard BERT architecture [11] but with two key differences. First, the network is tasked with recovering a corrupted protein sequence and simultaneously predicting function (GO annotation). Second, the network shares information between these two parallel paths via global attention wherein sequence embeddings comprise query and value vectors, but functional embeddings comprise the key vector. Functional embeddings are "global" – there is a single fixed-length vector per protein. This reduces the dimensionality of the query-key dot product by one and results in a decrease in parameter count and speed up during training and inference time.

The original model was released in TensorFlow and a public implementation was soon published in PyTorch [12]. We found the PyTorch implementation more amenable to repeated design-build-test cycles and further improved on it by wrapping it in PyTorch Lightning, introducing sinusoidal positional encodings, and fixing a few small bugs. We elected to train on UniRef50 rather than UniRef90 to decrease the number of functional GO groups from 8943 to 2449 and increase sequence diversity [13]. It has been previously shown that differences between models trained on UniRef100, UniRef90, and UniRef50 are minimal [4, 14]. We further changed the training scheme to not be length-dependent, as reported in the original paper. However, to ensure efficient batching, we elected to train on all proteins of length $<= 1022$ in the dataset (97.4%) with an 85%/10%/5% train/validation/test split. Finally, we trained for $3\times$ the steps reported in the original paper to ensure convergence.

2.2 Recombinant expression prediction

We followed the standard approach of fine-tuning on embeddings extracted from the foundational model. We anticipated expression to be a global property and predicted it as a binary classification problem from the global embedding, rather than the sequence embeddings. We tested several

architectures for this purpose and found two fully-connected (FC) layers to be sufficient while preventing overfitting; we call this the expression prediction head.

We focused on predicting recombinant expression in *E. coli* as there are several rich datasets on this task, although the network architecture could be easily adapted to predict expression in any organism. The data were sourced from the Protein Structure Initiative and the Seattle Structural Genomics Consortium, a total of 31,178 records which all comprised the train set. For the validation and held-out test set we manually curated a further 2,273 and 1,630 "challenging" human sequences from in-house experiments. To prevent overfitting, we trained with 50% dropout in the two FC layers and stopped training early if the accuracy difference between the validation and training datasets diverged by $> 2\%$. For the in-house protein BERT model, we permitted back-propagation through the entire network after a warmup period. Finally, to provide uncertainty estimation we ensembled several trained models.

2.3 Codon optimization

We treated codon optimization as an amino acid to codon translation problem, extracting sequence embeddings from the foundational model and learning codons from these embeddings via a Transformer decoder stack. We minimized categorical cross-entropy loss of the codons and considered additional constrains for downstream engineering such as a target GC content and minimal self-complementarity. Unlike previously reported deep learning codon optimization methods [15], this approach is not autoregressive and does not utilize the known preceding codons at training and inference time.

The initial aim was to optimize expression in *E. coli*. However, the *E. coli* genome only possesses 4,096 expressing genes, so we considered the set of all expressed genes of the *Escherichia* genus, derived from the NCBI RefSeq database, bringing the number of coding genes in our dataset to 331,854. We trained with a random 85%/10%/5% train/validation/test split. Unlike expression prediction, we did not permit back-propagation through the foundational network.

A further aim of codon optimization is to enable downstream experiments, *e.g.* mutational scans, that require precise manipulation of the nucleotide sequence via short, specific oligonucleotides (primers) and polymerase chain reaction (PCR). Typical requirements are that the optimized gene contains unique stretches of oligonucleotides and consistent G/C content (a proxy for the temperature at which a targeting primer can anneal). To this end, we introduced differentiable constraints with custom weights to the loss function and converted the codon log-probability distributions to one-hot nucleotides by the Gumbel-Softmax trick [16].

Self-complementarity loss was implemented as an inner product over the unfolded length dimension and A/T/G/C channels after unfolding the one-hot encoded DNA tensor[1], in Einstein notation: $X_{ile}X_{jle} = T_{ij}$, where X is the unfolded tensor, i and j are the unfolded dimensions of window size (7 or 8), l is the length dimension of and e is the one-hot encoded dimension.

G/C loss was defined as the 1-D convolution over the one-hot encoded nucleotide tensor, with a kernel of user-defined length (15 nucleotides here) with 0 in the A and T channels and 1 in the G and C channels.

As a control, we implemented a genetic algorithm optimizing the same parameters. The approach generated an initial population of 100 genes by sampling the *E. coli* codon table [17]. Over 50 generations, this population was expanded by 100 new members derived from random crossover of the ten lowest-scoring (best) genes and two lucky genes from the rest, with a mutation rate of 5%. Due to memory limitations on the GPU, only the 100 lowest-scoring members were retained per generation.

3 Results

3.1 Recombinant expression prediction

We first replicated the results of [8], observing that the unsupervised foundational model minimizes categorical cross-entropy loss for protein sequences and begins to converge after $\sim$20 epochs of UniRef50 or $\sim$1 billion sequences (not shown). We then added the recombinant expression prediction

[1] https://pytorch.org/docs/stable/generated/torch.Tensor.unfold.html

head, testing whether the global embeddings could be used as a basis for predicting expression and if there was any advantage to propagating through the entire network. The results are summarized in Figure 1. Fine-tuning just the expression head slightly improves accuracy and precision over the baseline (a naïve majority-label predictor). With full back-propagation, the results are further improved by $\sim$6% in both metrics. As a gold standard, we repeated the same experiment with ESM-1b. We found that using ESM-1b as a foundational model provided similar accuracy and precision on the training set but tended to overfit as the embedding vector was 10x larger and the model had 65x parameters. Although this could be alleviated by better structuring the expression prediction layers and hyper-parameter tuning, it would have come at 50x the computational cost of tuning the linear BERT (Table 1).

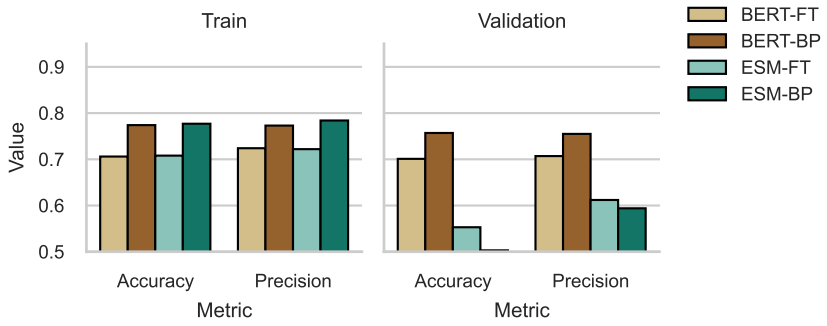

Figure 1: Accuracy and precision of predicting recombinant protein expression on the training/validation datasets for either the linear BERT or ESM-1b as the foundational model with the same two fully connected layers on top. Values are reported for just fine-tuning (FT, only the top layers are trained) and for fine-tuning and back-propagating (BP), where the top layers are fine-tuned for five warm-up epochs and then the full foundational model is back-propagated through for an additional epoch.

Table 1: Comparison of the average computational cost for embedding a single sequence with linear BERT and the much larger ESM-1b model (on the same hardware and back-propagating through an expression prediction head with two fully connected layers).

		Runtime (ms)	
Model	Parameters (M)	Training	Inference
ProteinBERT	10	2	0.8
ESM-1b	650	100	40

3.2 Codon optimization

Next, we tested whether the sequence embeddings could be used for codon optimization via a Transformer decoder architecture. We show overall decoding (amino acid to codon) accuracy in Figure 2. Despite training without any explicit restrictions on amino acid to codon linkage, almost no ($< 1e^{-7}$) amino acids are mistranslated (not shown). Additionally, we compare the decoder accuracy to previously published methods. N-grams are look-up tables that report the most frequent codon based on the single amino acid, single amino acid plus one neighbor either side, and single amino acid plus two neighbors either side. The 5-gram model has close to 3 million parameters and can effectively memorize the *E. coli* coding genome ($\sim$1 million amino acids). Finally, the state-of-the-art deep learning performance is captured by the bi-direction long short-term memory (LSTM) model, which additionally considers preceding codons [15]. Encouragingly, the decoder outperforms the bi-LSTM model without using codon information as input.

Finally, we assessed the capacity for the decoder head to learn engineering constraints such as G/C content and self-complementarity by training with two additional loss terms. At inference time, we sample a single sequence from the learned distribution. We find that the decoder struggles to optimize this new loss as the function derivatives are not smooth. This is made clear by comparison to a

genetic algorithm with the same objective function (Figure 2). Interestingly, the CPU implementation consistently finds lower-scoring variants than the GPU limitation. We speculate that this is due to the memory limits of the GPU, which could only retain a single generation, while the CPU can track all generations throughout evolution and sample more diversity. As expected, the deep learning (DL) method is orders of magnitude faster than the genetic algorithm, which is faster on the GPU than the CPU (Figure 2).

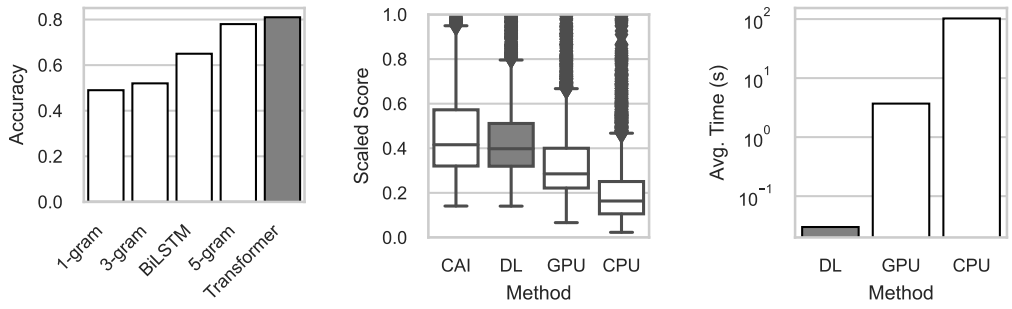

Figure 2: Summary of codon decoder head performance. (Left) Comparison of the amino acid to codon translation accuracy with N-gram methods and the SotA BiLSTM. (Center) Optimized score distributions for a test set of human protein targets. Score is comprised of negative log frequency codon usage plus quadratic penalty for distance from ideal GC content fraction and a self-complementarity penalty. (Right) Time required to produce a single "optimized" sequence with each method.

4 Discussion

Recent work on LaT models in protein science has largely focused on the development and application of models with increasingly large architectures. While expanded LaT model scale can improve performance on specific tasks, the accompanying large compute and memory requirements are non-ideal for data-limited tasks and tasks with rapid design-build-test cycles that require frequent re-training or fine-tuning. In this work, we present a complementary approach by developing an order-of-magnitude smaller LaT model with linear attention to achieve rapid and efficient training and fine-tuning for diverse downstream tasks.

Our model achieves 50x speed up in training and inferencing over the popular, larger ESM-1b LaT model. We leverage this increased training efficiency to fine-tune the model for recombinant protein expression prediction and codon optimization, achieving state-of-the-art performance with significantly reduced computation time. The smaller LaT model is less prone to overfit on a data-limited dataset for protein expression, highlighting its utility for predictive modeling of experimental data which is often limited by cost of collection. The lightweight architecture of the model allows for flexibility in iterative design, build, and test of custom models for specific engineering tasks such as including GC-content and self-complementarity metrics in codon optimization. We see significant future utility in continual re-training efforts for updating with evolving experimental data collection, and coupling with iterative active learning frameworks for further efficient predictive model training.

Acknowledgments and Disclosure of Funding

We acknowledge Cuong Nguyen (AI/ML, GSK) for helpful suggestions at early stages of the project and Brent Dorr (Synthetic Biochemistry, GSK), Paul Smyth (AI/ML, GSK), and the entire Protein Design and Informatics group for helpful discussions.

References

[1] The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa1100. URL

https://doi.org/10.1093/nar/gkaa1100.

[2] Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, Ekaterina Sakharova, Maxim Scheremetjew, Anton Korobeynikov, Alex Shlemov, Olga Kunyavskaya, Alla Lapidus, and Robert D Finn. MGnify: the microbiome analysis resource in 2020. *Nucleic Acids Research*, 48(D1):D570–D578, 11 2019. ISSN 0305-1048. doi: 10.1093/nar/gkz1035. URL https://doi.org/10.1093/nar/gkz1035.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017. doi: 10.48550/ARXIV.1706.03762. URL https://arxiv.org/abs/1706.03762.

[4] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021. doi: 10.1101/2021.07.09.450648. URL https://www.biorxiv.org/content/10.1101/2021.07.09.450648v1.

[5] Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, Jianzhu Ma, and Jian Peng. High-resolution de novo structure prediction from primary sequence. *bioRxiv*, 2022. doi: 10.1101/2022.07.21.500999. URL https://www.biorxiv.org/content/early/2022/07/22/2022.07.21.500999.

[6] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022. doi: 10.1101/2022.07.20.500902. URL https://www.biorxiv.org/content/early/2022/07/21/2022.07.20.500902.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. URL https://arxiv.org/abs/1910.01108.

[8] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, 02 2022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btac020. URL https://doi.org/10.1093/bioinformatics/btac020.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

[10] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL https://github.com/Lightning-AI/lightning.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

[12] Phil Wang. Proteinbert - pytorch. https://github.com/lucidrains/protein-bert-pytorch, 2021.

[13] Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H. Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, 03 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm098. URL https://doi.org/10.1093/bioinformatics/btm098.

[14] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing, 2020. URL https://arxiv.org/abs/2007.06225.

[15] David K. Yang, Samuel L. Goldman, Eli Weinstein, and Debora Marks. Generative models for codon prediction and optimization. 13–14 Dec 2019. URL https://mlcb.github.io/mlcb2019_proceedings/.

[16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016. URL https://arxiv.org/abs/1611.01144.

[17] Yasukazu Nakamura, Takashi Gojobori, and Toshimichi Ikemura. Codon usage tabulated from international DNA sequence databases: status for the year 2000. *Nucleic Acids Research*, 28(1): 292–292, 01 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.292. URL https://doi.org/10.1093/nar/28.1.292.