Bio-xLSTM: Generative modeling, representation and in-context learning of biological and chemical sequences

Niklas Schmidinger[1] **Lisa Schneckenreiter**[1] **Philipp Seidl**[1] **Johannes Schimunek**[1]
Pieter-Jan Hoedt[1] **Johannes Brandstetter**[1,2] **Andreas Mayr**[1]
Sohvi Luukkonen[1] **Sepp Hochreiter**[1,2] **Günter Klambauer**[1,2]

[1] ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning,
Johannes Kepler University, Linz, Austria
[2] NXAI GmbH, Linz, Austria

Abstract

Language models for biological and chemical sequences enable crucial applications such as drug discovery, protein engineering, and precision medicine. Currently, these language models are predominantly based on Transformer architectures. While Transformers have yielded impressive results, their quadratic runtime dependency on sequence length complicates their use for long genomic sequences and in-context learning on proteins and chemical sequences. Recently, the recurrent xLSTM architecture has been shown to perform favorably compared to Transformers and modern state-space models (SSMs) in the natural language domain. Similar to SSMs, xLSTMs have linear runtime dependency and allow for constant-memory decoding at inference time, which makes them prime candidates for modeling long-range dependencies in biological and chemical sequences. In this work, we tailor xLSTM towards these domains and we propose a suite of language models called Bio-xLSTM. Extensive experiments in three large domains, genomics, proteins, and chemistry, were performed to assess xLSTM's ability to model biological and chemical sequences. The results show that Bio-xLSTM is a highly proficient generative model for DNA, protein, and chemical sequences, learns rich representations, and can perform in-context learning for proteins and small molecules.

1 Introduction

Accurate computational models for biological sequences are essential for translating data into actionable insights in modern biology. Biological sequences like DNA, RNA, and proteins are central to molecular biology, genomics, and drug discovery. Major projects like the Human Genome Project (Lander et al., 2001) and the 1000 Genomes Project (1000 Genomes Project Consortium, 2010) have driven large- scale data collection efforts. Modeling these sequences is key to advancing life sciences (Benegas et al., 2023; Karollus et al., 2024), interacting with biological systems (Hopf et al., 2017; Riesselman et al., 2018; Yang et al., 2019) or predicting phenotypes from genetic variants (Ashley, 2016; Brandes et al., 2023; Acosta et al., 2022). Similar efforts exist for protein sequences (The UniProt Consortium, 2023) and small molecules (Kim et al., 2023; Zdrazil et al., 2023), used for tasks like protein engineering (Arnold, 2018; Yang et al., 2019), predicting 3D structures (Jumper et al., 2021), and drug discovery (Zhavoronkov et al., 2019). Large language models (LLMs) (Brown et al., 2020; Bubeck et al., 2023) have emerged as prime candidates for modeling biological sequences

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

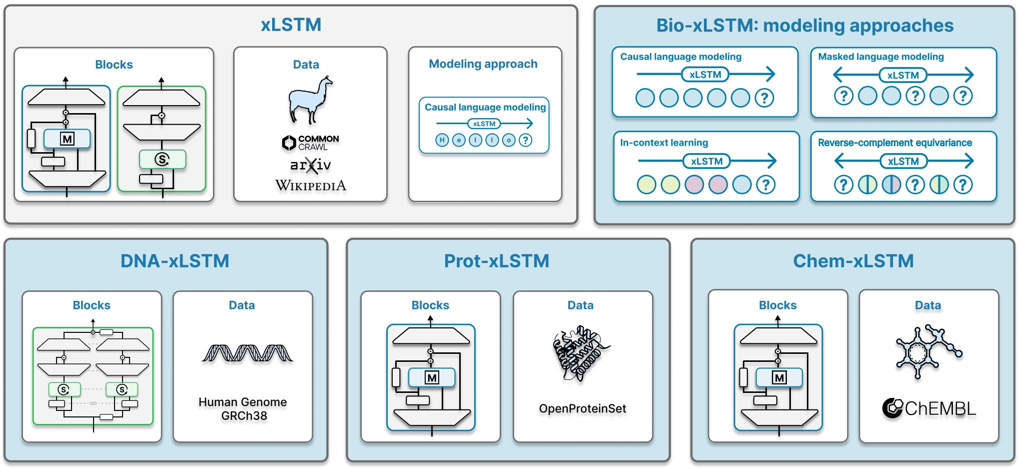

Figure 1: Overview of Bio-xLSTM. **Top left:** xLSTM for natural language processing tasks. **Top right:** Additional modeling approaches for biological sequences: masked language modeling, equivariance to reverse complementary sequence, and in-context learning. **Bottom left:** DNA-xLSTM models are trained on genomic DNA sequences and then used for fine-tuning on downstream tasks. **Bottom center:** Prot-xLSTM models are trained in a causal modeling setting with a fill-in-the-middle objective and use homologous proteins for in-context learning. **Bottom right:** Chem-xLSTM models are trained to generate small molecules. For an in-context learning setting, Chem-xLSTM uses molecules with known properties.

and serving as foundation models for molecular biology and chemistry (Ji et al., 2021; Schiff et al., 2024; Nguyen et al., 2023; Rives et al., 2021; Lin et al., 2023).

Large language models for biological sequences must handle long sequences and incorporate context. The rise of LLMs (Radford et al., 2018; Brown et al., 2020; Bubeck et al., 2023) has revolutionized numerous fields, including life sciences. Most LLMs are based on the Transformer architecture (Vaswani et al., 2017), which excels at predicting the next or missing token using self-attention. However, this mechanism scales quadratically with sequence length, making long-sequence processing computationally expensive. Biological sequences, with their important long-range interactions due to 3D folding, require long context windows for accurate modeling, which is essential for gene regulation in DNA (Bouwman & de Laat, 2015) and protein function (Anfinsen, 1973). Long contexts also benefit models to exploit homologous proteins (Truong Jr & Bepler, 2023; Sgarbossa et al., 2024) and molecular context for small molecules (Papadatos et al., 2010; Schimunek et al., 2023). The human genome spans around three billion base-pairs (bps), far exceeding the context limits of Transformer-based models. As a result, most biological sequence models use short contexts (Rives et al., 2021; Ji et al., 2021; Dalla-Torre et al., 2023). The emergence of state-space models (SSMs), like S4 (Gu et al., 2022), Hyena (Poli et al., 2023), and Mamba (Gu & Dao, 2023), enables handling longer contexts in biological domains (Nguyen et al., 2023; Schiff et al., 2024; Sgarbossa et al., 2024). However, the recently proposed xLSTM (Beck et al., 2024), a recurrent neural network, has outperformed these architectures in natural language processing (Beck et al., 2024). For further related work, see Appx. A.

The recently proposed xLSTM is a powerful architecture for sequence modeling and a promising candidate for biological and chemical sequences. The xLSTM architecture (Beck et al., 2024) introduces enhanced memory structures and exponential gates that boost its performance, particularly in natural language modeling. Despite these enhancements, xLSTM retains the efficiency of a recurrent neural network and can handle varying sequence lengths effectively (Beck et al., 2024). It introduces two new layers: a) sLSTM, with exponential gates that improve state tracking (Merrill et al., 2024), and b) mLSTM, which allows switching between parallel training and recurrent inference modes, enabling scalability to larger contexts (Katharopoulos et al., 2020; Choromanski et al., 2021). These features make xLSTM ideal for modeling: i) DNA sequences, which are inherently long and contain long-range interactions, ii) protein sequences, where modeling strongly benefits from contextual information of evolutionary-related proteins (Rives et al., 2021), and iii) small molecules represented as chemical sequences, such as Simplified Molecular Input Line Entry System (SMILES) (Weininger, 1988), for which in-context learning (ICL) abilities are required to generate new molecules with desired properties or from a particular molecular domain (Segler et al., 2018; Schimunek et al., 2023). However, how to best tailor xLSTM for biological and chemical sequences remains unclear, or how it compares to other domain-specific LLMs.

Contributions. We introduce: a) DNA-xLSTM, a model tailored for DNA sequences with reverse-complement equivariant blocks, and evaluate its performance on long-context generative modeling, representation learning, and downstream tasks. b) Prot-xLSTM, a homology-aware protein language model with in-context learning, which we benchmark on generative modeling and conditioned protein design tasks. c) Chem-xLSTM, a sequence model for SMILES representations of small molecules for which we demonstrate ICL capabilities. An overview of Bio-xLSTM is shown in Fig. 1.

2 xLSTM: Background and Notation

sLSTM and mLSTM layers. xLSTM (Beck et al., 2024) make use of two types of layers: sLSTM (see Appendix Section B.1) and mLSTM (see Appendix Section B.2) which are the main components within residual block structures (see Appendix Section B.3) of its multi-layer architectures. We consider a series of input vectors $\boldsymbol{x}_t \in \mathbb{R}^D$ given at a certain time step $t \in \{1, \ldots, T\}$. $\boldsymbol{X} = \boldsymbol{X}_{1:T} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{D \times T}$ denotes the matrix of stacked input vectors from all time steps. Both sLSTM and mLSTM are recurrent neural networks, which either map a state $(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{n}_{t-1})$ to a successor state $(\boldsymbol{h}_t, \boldsymbol{c}_t, \boldsymbol{n}_t)$ given an input $\boldsymbol{x}_{t-1}$ (sLSTM) or a state $(\boldsymbol{h}_{t-1}, \boldsymbol{C}_{t-1}, \boldsymbol{n}_{t-1})$ to a successor state $(\boldsymbol{h}_t, \boldsymbol{C}_t, \boldsymbol{n}_t)$ given an input $\boldsymbol{x}_{t-1}$ (mLSTM). Here, $\boldsymbol{h}_t \in \mathbb{R}^d$ denotes a hidden state, $\boldsymbol{c}_t \in \mathbb{R}^d$ and $\boldsymbol{C}_t \in \mathbb{R}^{d \times d}$ denote cell states responsible for long-term memory and, $\boldsymbol{n}_t \in \mathbb{R}^d$ denotes a normalizer state. sLSTM and mLSTM utilize several adjustable weight matrices and bias vectors and employ input-, output-, and forget-gates, activated by exponential $(\exp)$ or the sigmoid functions (σ). For cell inputs in sLSTM, the hyperbolic tangent function $(\tanh$, abbreviated as $\varphi)$ is used as an activation function.

The xLSTM architecture (detailed in Appendix Section B.3), including all layers, normalization, blocks, and other components, defines a mapping from an input sequence of length t to an output sequence. This mapping is denoted as $\mathrm{xLSTM} : \mathbb{R}^{D \times t} \mapsto \mathbb{R}^{D \times t}$, where xLSTM transforms the stacked inputs up to time step t, i.e., $\boldsymbol{X}_{1:t} := (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \in \mathbb{R}^{D \times t}$, to the corresponding stacked outputs of sequence length t, i.e., $\boldsymbol{Y}_{1:t} := (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t) \in \mathbb{R}^{D \times t}$ [1]. The i-th sequence element is denoted with the subscript i, e.g. the i-th element from $\boldsymbol{X}_{1:t}$ would be $(\boldsymbol{X}_{1:t})_i$. Similarly to the mapping xLSTM, we also define mappings for the sequence-wise input-/output behaviour of layers themselves for an sLSTM layer $(\mathrm{sLSTM} : \mathbb{R}^{D \times t} \mapsto \mathbb{R}^{D \times t})$ or an mLSTM layer $(\mathrm{mLSTM} : \mathbb{R}^{D \times t} \mapsto \mathbb{R}^{D \times t})$. If the specific parameters used for the mapping are unclear, we will denote this by including a second argument in the function, separated by a semicolon.

2.1 Modes of Operation: Parallel, Chunkwise, and Recurrent

The recurrent forms of sLSTM and mLSTM, detailed in Appendix Sections B.1 and B.2, provide efficient, constant-memory decoding during inference. This eliminates the need for expensive key-value caching, which represents a major challenge for Transformer models in long-range settings. Like Transformers, mLSTM allows for parallelization across the sequence length which significantly speeds up training. Additionally, similar to linear attention variants (Katharopoulos et al., 2020; Yang et al., 2024), mLSTM supports chunkwise parallel processing, blending recurrent and parallel modes. This approach is especially advantageous for long-sequence training and prompt encoding. For further details, refer to Appendix B.4.

3 Bio-xLSTM: Longe-Range Modeling of Biological and Chemical Sequences

Bio-xLSTM introduces three xLSTM-based variants tailored specifically to DNA (Section 3.3), proteins (Section 3.4) and small molecules (Section 3.5). For these application domains, we extend xLSTM from causal language modeling (CLM) to new modeling approaches such as fill-in the middle

[1] Here $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ represent the inputs to and outputs from a particular model from an instance of an xLSTM architecture, rather than the inputs and outputs of a specific sLSTM or mLSTM layer

(FIM) (Section 3.1), in-context learning (ICL) (Section 3.1), and masked language modeling (MLM) (Section 3.2).

3.1 Causal Language Modeling and Next-Token Prediction

Causal language modeling (CLM) uses the

$$\text{CLM loss: } \mathcal{L}^{\text{CLM}} = \mathbb{E}_{\boldsymbol{X} \sim p_{\boldsymbol{X}}} \, \mathbb{E}_{t \sim [[1, T-1]]} \, \text{CE}\left(\boldsymbol{x}_{t+1}, \text{xLSTM}(\boldsymbol{X}_{1:t})_t\right), \tag{1}$$

where CE is the cross-entropy loss (with logits), $p_{\boldsymbol{X}}$ is the data distribution, and $[[1, T-1]]$ is the discrete uniform distribution from 1 to $T-1$. The objective measures how well a particular sequence token $\boldsymbol{x}_{t+1}$ can be predicted based on the previous tokens $\boldsymbol{X}_{1:t}$ by the model $\text{xLSTM} : \mathbb{R}^{D \times t} \mapsto \mathbb{R}^{D \times t}$. Therefore, this type of modeling is sometimes also called *next token prediction (NTP)*, *uni-directional modeling* or *autoregressive (AR) modeling* and the loss is also called *NTP loss*.

Fill-in the middle (FIM) (Bavarian et al., 2022) is a training paradigm that integrates aspects of both CLM and MLM. In this approach, parts of the sequence are replaced with mask tokens, which are then appended to the end of the sequence. This allows the model to utilize the entire context to predict the masked tokens while maintaining an AR training framework. This strategy, allows the model to perform both a) generative modeling and b) inpainting with CLM.

In-context learning (ICL) is a paradigm that describes that the predictions of the model $\boldsymbol{Y} = \text{xLSTM}(\boldsymbol{X})$ improve when a suitable context $\boldsymbol{Z} \in \mathbb{R}^{D \times S}$ is provided: $\boldsymbol{Y}' = \text{xLSTM}([\boldsymbol{Z}, \boldsymbol{X}])_{(S+1):(S+T)}$, where $[\boldsymbol{Z}, \boldsymbol{X}]$ indicates concatenation, the subscript $(S+1):(S+T)$ denotes that the last output tokens (those corresponding to the $\boldsymbol{X}$ tokens) are selected, and $\boldsymbol{Y}'$ is the output of the model with context $\boldsymbol{Z}$ as input. For natural language processing tasks, $\boldsymbol{Z}$ often contains the solution to a similar problem, or some exemplary solutions, that inform the input and the model. For biological and chemical sequences, $\boldsymbol{Z}$ could be similar genetic regions, homologous proteins, or molecules with desired properties.

3.2 Masked Language Modeling (MLM)

Bio-xLSTM extends xLSTM to masked modeling of biological sequences, for which the typical de-masking or de-noising objective (Vincent et al., 2010; Devlin et al., 2019) is used, concretely the

$$\text{MLM loss: } \mathcal{L}^{\text{MLM}} = \mathbb{E}_{\boldsymbol{X} \sim p_{\boldsymbol{X}}} \, \mathbb{E}_{t \sim [[1, T]]} \, \mathbb{E}_{\boldsymbol{M} \sim p_{\boldsymbol{M}}} \, \text{CE}\left(\boldsymbol{x}_t, \text{xLSTM}(\boldsymbol{X} \odot \boldsymbol{M})_t\right), \tag{2}$$

where $\boldsymbol{M} \in \{0, 1\}^{D \times T}$ is a random matrix with binary entries which are usually drawn from a Bernoulli distribution $p_{\boldsymbol{M}}$, and $\odot$ is element-wise multiplication. The objective measures how well the original sequence $\boldsymbol{X}$ can be reconstructed from a noisy version $\boldsymbol{X} \odot \boldsymbol{M}$ by the model $\text{xLSTM} : \mathbb{R}^{D \times T} \mapsto \mathbb{R}^{D \times T}$. This modeling paradigm has also been called *bidirectional modeling*. It has been highly successful in learning representations of proteins at evolutionary scale (Rives et al., 2021), which has powered many subsequent applications such as protein engineering and machine-learning guided directed evolution (Yang et al., 2019). For details on how xLSTM is extended to the MLM setting, we refer to Appendix Section B.5.

Reverse complement (RC) equivariance. We develop an xLSTM block that is equivariant to the RC of an input sequence, a property particularly relevant to DNA-based applications. In double-helix DNA structures, both strands are semantically equivalent, with one strand being the RC of the other. The RC strand is oriented in the opposite direction of the *forward* strand, with base pairs converted from `A` to `T` and `C` to `G`. Shrikumar et al. (2017) show that a data-driven approach to learn the equivalence between RC sequences can fail. Therefore, Schiff et al. (2024) propose to enforce RC-equivariance by design, making use of two different inductive biases, post-hoc conjoining (PH) (Zhou et al., 2022) and parameter sharing (PS), in the model architecture. In PH models, the backbone is trained to handle both DNA sequences and their RCs by applying RC augmentations during pre-training. For downstream tasks, PH models are applied to both the original sequence and its RC, and their outputs are summed to reach overall RC invariance. In contrast, PS models, as introduced in Schiff et al. (2024), integrate RC-equivariant xLSTM blocks with equivariant word embeddings and language model heads. For additional details, see Appendix Section C.4.

3.3 DNA-xLSTM

For the DNA domain, we propose the DNA-xLSTM architecture to enhance sequence modeling capabilities, particularly for varying context lengths. We introduce three versions of the DNA-xLSTM architecture: two sLSTM-based models trained with a context window of 1,024 tokens (DNA-xLSTM-500k and DNA-xLSTM-2M), and a mLSTM-based model trained with a context window of 32,768 tokens (DNA-xLSTM-4M). The short-context model, DNA-xLSTM-500k, has an embedding dimension of 128, 5 sLSTM blocks, an up-projection ratio of 1.25:1 to match the baseline model parameter count, and a total parameter count of 500k, while DNA-xLSTM-2M has an embedding dimension of 256, 6 sLSTM blocks, a 1:1 up-projection ratio, and 2M parameters, The long-context model, DNA-xLSTM-4M, has an embedding dimension of 256, 9 mLSTM blocks, a 2:1 up-projection ratio, and is augmented with Rotary Position Encodings (RoPE) (Su et al., 2024a) to handle long-range dependencies effectively, with a total of 4M parameters. All three model architectures are trained with both CLM and MLM. Furthermore, we introduce RC-equivariant versions, xLSTM-PH and xLSTM-PS, which use the original sequence and its reverse complement. We benchmarked these models against state-of-the-art DNA models, such as Transformers, DNA-Mamba (Caduceus) (Schiff et al., 2024), and HyenaDNA (Nguyen et al., 2023), showing competitive or better performance on pre-training and downstream classification tasks (see Section 4.1).

3.4 Prot-xLSTM

For the protein domain, we propose Prot-xLSTM to address the complexities of protein sequence data, particularly in capturing long-range dependencies to enable homology-conditioned modeling. We introduce two versions of Prot-xLSTM: Prot-xLSTM-26M and Prot-xLSTM-102M, with 26M and 102M parameters, respectively. Both architectures are trained with variable context sizes ranging from 2,048 to 131,072 tokens. Both models consist of 16 mLSTM blocks, with embedding dimensions of 512 for Prot-xLSTM-26M and 1,024 for Prot-xLSTM-102M and maintaining a consistent 2:1 projection ratio across both models. To effectively manage the wide range of protein sequence lengths and context sizes, RoPEs (Su et al., 2024a) are implemented for Prot-xLSTM. These models are trained with CLM using a FIM strategy on non-aligned homologous sequences from the OpenProteinSet dataset (Ahdritz et al., 2023), enabling them to perform ICL at inference time in two modes: (a) generative and (b) inpainting. Both approaches can be used for protein design, with the latter also suited for residue-based predictions, such as mutant fitness estimation. Prot-xLSTM shows better performance than similarly conceived Mamba-and Transformer-based models and shows promising results for homology-conditioned sequence generation (see Section 4.2).

3.5 Chem-xLSTM

For the chemical sequence modeling domain, Chem-xLSTM is developed to enhance the generative modeling capabilities for SMILES strings (Weininger, 1988), a sequence representation of small molecules. The Chem-xLSTM-15M model for unconditional molecule generation consists of 15M parameters and 9 mLSTM blocks, each with an embedding dimension of 512 and a 1.3:1 projection ratio. The model is trained with CLM with a context length of 100 tokens for consistency with previous work (Özçelik et al., 2024). Additionally, we introduce a conditional Chem-xLSTM model, which shares the same architecture but is trained on concatenated SMILES sequences with a context length of 4,096 tokens to enable ICL tasks. This model can generate molecules within a specific domain without fine-tuning, a highly sought-after capability in drug discovery, and also demonstrates few-shot activity prediction abilities. The models have been benchmarked against other generative models for SMILES and at their ICL capabilities (see Section 4.3).

4 Experiments and Results

4.1 DNA Sequences

For the DNA-xLSTM experiments, we followed the experimental protocol outlined in Schiff et al. (2024) and Nguyen et al. (2023) for both pre-training and downstream adaptation.

Pre-training. The training data for both the CLM and MLM tasks was sourced from the human reference genome (Church et al., 2011), with context lengths set to 1,024 and 32k tokens. Our

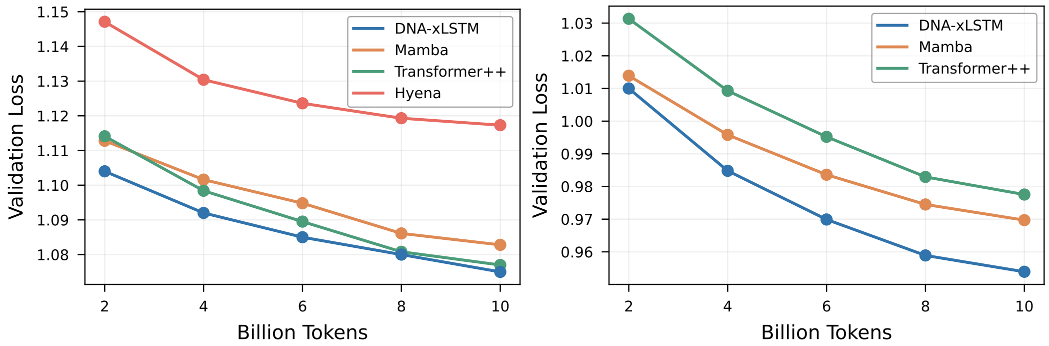

Figure 2: Pre-training of 2M-parameter DNA models on the human reference genome (GRCh38). Models are trained at single-nucleotide resolution with a context length of 1024 bases. **Left: causal language modeling.** Learning curves display **NTP loss** (↓) on a test set, plotted against the number of tokens processed. **Right: masked language modeling.** Learning curves showing **MLM loss** (↓) on a test set across the number of tokens seen for various models. In both tasks, the xLSTM-based models consistently achieve the lowest loss values across all update steps.

baseline models included HyenaDNA (Nguyen et al., 2023) and Caduceus, which is based on the Mamba architecture (Schiff et al., 2024). Additionally, we trained Transformer++ baselines, building on the Llama architecture (Touvron et al., 2023). Similar to Caduceus, we experimented with both PH- and PS-equivariant xLSTM configurations, benchmarking them against the corresponding Mamba baselines. All models that did not use PS-equivariance were trained with RC augmentation. Hyperparameters were selected using a separate validation set. Figure 2 presents the test losses for 2M parameter CLM and MLM models trained with RC augmentation, i.e. non-PS models, and a context size of 1,024 tokens. In the CLM setting, DNA-xLSTM-2M achieved the best performance, surpassing Transformer++, Mamba, and HyenaDNA. The performance gap became even more pronounced on the MLM task, where DNA-xLSTM-2M outperformed both Transformer-based models and Mamba. Additionally, we extended DNA-xLSTM-2M to the PS equivariant setting and trained smaller RC-equivariant DNA-xLSTM-500k models. The resulting PH and PS models were subsequently used for downstream fine-tuning. In Appendix Section C, we present additional pre-training results including comparisons for large-context and PS equivariant models. We found that xLSTM-DNA matches or outperforms strong baselines in all pre-training settings.

Downstream fine-tuning. To evaluate the learned representations, we fine-tuned the pre-trained DNA-xLSTM-2M and DNA-xLSTM-500K (both PH and PS) on two genomic classification benchmarks: the Genomic benchmark (Grešová et al., 2023) and the Nucleotide Transformers Tasks (Dalla-Torre et al., 2023), which span 18 datasets from five studies. DNA-xLSTM-2M-PH and DNA-xLSTM-2M-PS models pre-trained with context size 1,024 were compared against HyenaDNA and Mamba-PS and Mamba-PH. DNA-xLSTM performed best (see Table 1), outperforming baseline models in the under 2M parameter range on 12 out of 18 tasks, and was comparable to the much larger Nucleotide Transformer (500M parameters), winning 8 tasks. The comparisons with larger Transformer models and xLSTM-DNA-500k performance on the Genomics benchmark are presented in Appendix Section C.

4.2 Protein Sequences

We followed the experimental protocols from Sgarbossa et al. (2024) for protein sequences.

Homology-aware training. Training data was sourced from the filtered OpenProteinSet (Ahdritz et al., 2023), consisting of 270k UniClust30 clusters (508M sequences, 110B residues). Using the ProtMamba pipeline, we constructed homology-aware, alignment-free inputs by concatenating unaligned homologous sequences and mask patches for training with the FIM strategy. We trained two xLSTM-based models: Prot-xLSTM-26M and Prot-xLSTM-102M. For comparison, we also trained a smaller ProtMamba (ProtMamba-28M) and Transformer-based (Prot-Transformer++-26M) (Touvron et al., 2023) model and used the *ProtMamba Long Foundation* (ProtMamba-107M) provided by Sgarbossa et al. (2024). Training followed a context length scheduling strategy, with models gradually

Table 1: Downstream adaption of DNA models. The performance of 2M parameter models fine-tuned on Nucleotide Transformer classification tasks on the test set is shown. PS or PH indicate models trained to be RC equivariant. Performance is averaged over 10 random seeds and error bars indicate the difference between maximum and minimum values across the 10 runs. The best values are highlighted in green. DNA-xLSTM outperforms both Mamba and Hyena on 12 out of 18 tasks. Scores for Mamba- and Hyena-based models were obtained from Schiff et al. (2024).

Task	Metric	HyenaDNA	Mamba-PS[a]	Mamba-PH[a]	xLSTM-PS	xLSTM-PH
Histone Markers						
H3	MCC ↑	$0.779^{\pm 0.037}$	$0.799^{\pm 0.029}$	$0.815^{\pm 0.048}$	$0.796^{\pm 0.014}$	$0.824^{\pm 0.010}$
H3K14AC	MCC ↑	$0.612^{\pm 0.065}$	$0.541^{\pm 0.212}$	$0.631^{\pm 0.026}$	$0.570^{\pm 0.008}$	$0.598^{\pm 0.017}$
H3K36ME3	MCC ↑	$0.613^{\pm 0.041}$	$0.609^{\pm 0.109}$	$0.601^{\pm 0.129}$	$0.588^{\pm 0.019}$	$0.625^{\pm 0.010}$
H3K4ME1	MCC ↑	$0.512^{\pm 0.024}$	$0.488^{\pm 0.102}$	$0.523^{\pm 0.039}$	$0.490^{\pm 0.012}$	$0.526^{\pm 0.001}$
H3K4ME2	MCC ↑	$0.455^{\pm 0.095}$	$0.388^{\pm 0.101}$	$0.487^{\pm 0.170}$	$0.489^{\pm 0.024}$	$0.504^{\pm 0.012}$
H3K4ME3	MCC ↑	$0.549^{\pm 0.056}$	$0.440^{\pm 0.202}$	$0.544^{\pm 0.045}$	$0.520^{\pm 0.019}$	$0.537^{\pm 0.012}$
H3K79ME3	MCC ↑	$0.672^{\pm 0.048}$	$0.676^{\pm 0.026}$	$0.697^{\pm 0.077}$	$0.662^{\pm 0.011}$	$0.697^{\pm 0.007}$
H3K9AC	MCC ↑	$0.581^{\pm 0.061}$	$0.604^{\pm 0.048}$	$0.622^{\pm 0.030}$	$0.622^{\pm 0.013}$	$0.627^{\pm 0.008}$
H4	MCC ↑	$0.763^{\pm 0.044}$	$0.789^{\pm 0.020}$	$0.811^{\pm 0.022}$	$0.793^{\pm 0.011}$	$0.813^{\pm 0.008}$
H4AC	MCC ↑	$0.564^{\pm 0.038}$	$0.525^{\pm 0.240}$	$0.621^{\pm 0.054}$	$0.558^{\pm 0.018}$	$0.583^{\pm 0.014}$
Regulatory Annotation						
Enhancer	MCC ↑	$0.517^{\pm 0.117}$	$0.491^{\pm 0.066}$	$0.546^{\pm 0.073}$	$0.375^{\pm 0.030}$	$0.545^{\pm 0.024}$
Enhancer Types	MCC ↑	$0.386^{\pm 0.185}$	$0.416^{\pm 0.095}$	$0.439^{\pm 0.054}$	$0.444^{\pm 0.046}$	$0.466^{\pm 0.011}$
Promoter: All	F1 ↑	$0.960^{\pm 0.005}$	$0.967^{\pm 0.004}$	$0.970^{\pm 0.004}$	$0.962^{\pm 0.002}$	$0.967^{\pm 0.001}$
NonTATA	F1 ↑	$0.959^{\pm 0.011}$	$0.968^{\pm 0.006}$	$0.968^{\pm 0.010}$	$0.963^{\pm 0.002}$	$0.970^{\pm 0.001}$
TATA	F1 ↑	$0.944^{\pm 0.040}$	$0.957^{\pm 0.015}$	$0.953^{\pm 0.016}$	$0.948^{\pm 0.006}$	$0.952^{\pm 0.005}$
Splice Site Annotation						
All	Accuracy ↑	$0.956^{\pm 0.011}$	$0.927^{\pm 0.021}$	$0.940^{\pm 0.027}$	$0.965^{\pm 0.006}$	$0.974^{\pm 0.004}$
Acceptor	F1 ↑	$0.958^{\pm 0.010}$	$0.936^{\pm 0.077}$	$0.937^{\pm 0.033}$	$0.970^{\pm 0.005}$	$0.953^{\pm 0.008}$
Donor	F1 ↑	$0.949^{\pm 0.024}$	$0.874^{\pm 0.289}$	$0.948^{\pm 0.025}$	$0.962^{\pm 0.004}$	$0.951^{\pm 0.005}$

[a] this method is also called Caduceus (Schiff et al., 2024).

increasing context from 2^{11} to 2^{17} tokens. We evaluated models using negative log-likelihood and perplexity, calculated for different parts of the concatenated-FIM sequences. As shown in Fig. 3 and Tab. 2, Prot-xLSTM outperformed the other architectures. Its advantage becomes even more pronounced with longer contexts, which Prot-Transformer++ cannot handle, and where Prot-xLSTM significantly outperforms ProtMamba. Furthermore, Prot-xLSTM-102M outperforms ProtMamba-107M, despite being trained on less than a third of the total training tokens used for ProtMamba-107M. Further details are provided in Appendix Section D.1.

Table 2: Performance comparison of protein language models at homology-conditioned generation. Test set **perplexity** (↓) of different models with a context size of 2^{17} is shown across different token subsets. The average and 95% confidence interval values are computed across the test set clusters. Prot-xLSTM outperforms ProtMamba, especially when using a long context.

	Prot-xLSTM-26M	ProtMamba-28M	Prot-xLSTM-102M	ProtMamba-107M
All tokens	$8.73^{\pm 0.31}$	$10.15^{\pm 0.32}$	$6.83^{\pm 0.25}$	$7.47^{\pm 0.26}$
First seq tokens	$15.40^{\pm 0.26}$	$15.28^{\pm 0.26}$	$13.36^{\pm 0.35}$	$13.04^{\pm 0.36}$
Last seq tokens	$9.19^{\pm 0.30}$	$11.08^{\pm 0.27}$	$7.32^{\pm 0.29}$	$8.37^{\pm 0.29}$
FIM tokens	$6.77^{\pm 0.25}$	$7.96^{\pm 0.27}$	$5.52^{\pm 0.20}$	$6.47^{\pm 0.23}$

Homology-conditioned protein generation. We generate 2,500 protein sequences each for 19 clusters using different parameters and score them using multiple metrics. Hamming distance, HMMER score, and structural scores correlate well with sequence perplexity, with an average absolute Pearson correlation of 0.57 across clusters for the large Prot-xLSTM model (Table A10). Table 3 shows Kolmogorov-Smirnov test statistics, which quantifies how well the score distributions of the generated proteins match those of the real proteins. For each cluster, we compared scores between 100 random real proteins and the 100 generated proteins with the lowest perplexity. For further details see Appx. D.2.

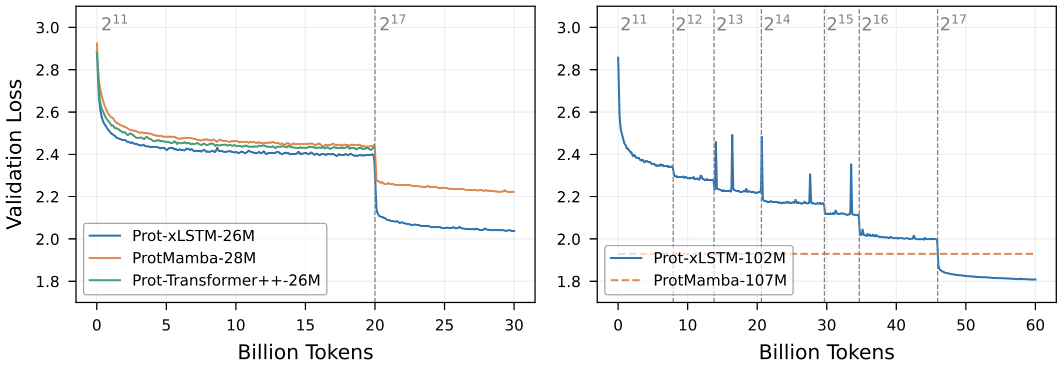

Figure 3: Generative pre-training of protein language models. The learning curves show the validation loss of homology-aware protein language models during training. **Left:** Smaller (∼25M parameters) models trained for 20B tokens with a context size of 2^{11} and fine-tuned for 10B with a context of 2^{17} tokens. Transformer++ can only be run for a small context size. **Right:** Prot-xLSTM-102M model trained with increasing context sizes from 2^{11} to 2^{17}. The orange dashed line corresponds to the validation loss of ProtMamba-107M trained up to a context size of 2^{17} for a total of 195B tokens. Vertical gray dashed lines mark the points where context size was increased. Prot-xLSTM consistently outperforms other models and sets a new state-of-the-art at homology-aware generation.

Table 3: Homology-conditioned protein generation. Average **Kolmogorov-Smirnov statistic** (↓) between scores of natural and generated sequences with 95% confidence intervals across 19 homology clusters. For three of five metrics, score distributions of Prot-xLSTM-generated sequences are closest to natural sequences.

	Prot-xLSTM-26M	ProtMamba-28M	Prot-xLSTM-102M	ProtMamba-107M
Sequence Length	$0.41^{\pm 0.09}$	$0.52^{\pm 0.09}$	$0.40^{\pm 0.08}$	$0.36^{\pm 0.08}$
Min. Hamming	$0.43^{\pm 0.08}$	$0.60^{\pm 0.11}$	$0.47^{\pm 0.09}$	$0.42^{\pm 0.07}$
HMMER	$0.57^{\pm 0.10}$	$0.54^{\pm 0.11}$	$0.44^{\pm 0.09}$	$0.49^{\pm 0.10}$
pLDDT	$0.40^{\pm 0.09}$	$0.68^{\pm 0.12}$	$0.27^{\pm 0.05}$	$0.30^{\pm 0.07}$
pTM	$0.38^{\pm 0.08}$	$0.72^{\pm 0.10}$	$0.26^{\pm 0.05}$	$0.28^{\pm 0.05}$

4.3 Chemical Sequences

Unconditional molecule generation aims to produce valid small organic molecules without imposing specific constraints, such as being from a particular molecular domain. Following the setup from Özçelik et al. (2024), we trained models to generate SMILES strings using a CLM approach on a dataset derived from ChEMBL with a context length of 100 tokens. We compared our Chem-xLSTM model with several architectures, including LSTM, GPT, S4, and Mamba, with all models containing approximately 15 million parameters. The evaluation focused on two primary metrics: perplexity and Fréchet ChemNet Distance (FCD) (Preuer et al., 2018). Chem-xLSTM achieved the lowest FCD of 0.13 and a competitive perplexity score of 1.68, indicating its strong ability to generate realistic chemical structures (see Table 4). All models produced valid, unique, and novel molecules, showcasing their effectiveness in this task. Further details and results are provided in Appx. E.1.

Table 4: Unconditional generation of molecules with 15M parameter models. 102,400 SMILES sequences have been generated and evaluated. Error bars represent standard deviations across training re-runs. Green cells highlight the best values per column. Chem-xLSTM yields the best FCD and SMILES-GPT the best perplexity.

	SMILES-LSTM[a]	SMILES-GPT[b]	SMILES-S4[c]	Chem-Mamba[d]	Chem-xLSTM
FCD ↓	$0.46^{\pm <0.01}$	$0.15^{\pm <0.01}$	$0.28^{\pm <0.01}$	$0.21^{\pm <0.01}$	$0.13^{\pm <0.01}$
Perplexity ↓	$1.88^{\pm 3.8}$	$1.65^{\pm 0.6}$	$1.73^{\pm 2.4}$	$1.74^{\pm 0.5}$	$1.68^{\pm 1.0}$

[a] Segler et al. (2018) [b] Adilov (2021) [c] Özçelik et al. (2024) [d] Adapted to SMILES in this work.

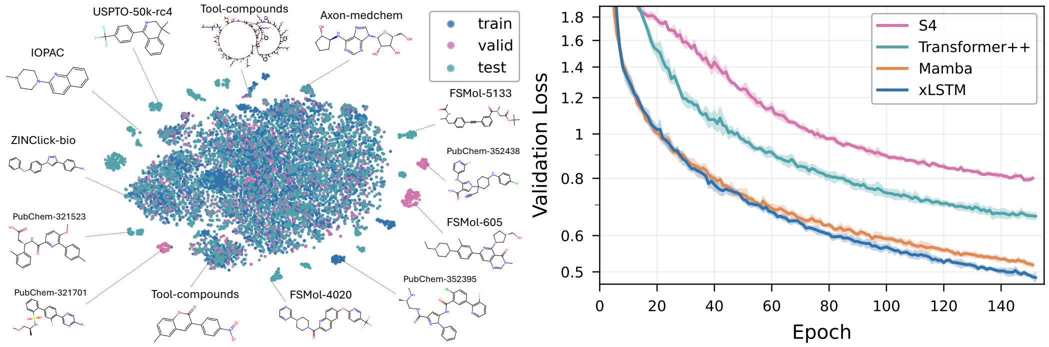

Figure 4: Conditional generation of molecules via ICL and 15M parameter models. **Left:** Visualization of different molecular domains contained in the *molecular domains* dataset. A t-SNE down-projection of molecules from different domains is shown. Clusters on the exterior represent highly specific molecular domains. The validation and test set contain highly specific, unseen molecular domains. **Right:** Generative training of chemistry language models on the *molecular domains* dataset. Learning curves showing mean **CLM loss** (↓) on a validation set across the training epochs. Shaded areas represent the standard-deviation over runs. The Chem-xLSTM achieved the lowest loss at conditional generation of molecules using ICL.

For **conditional molecule generation**, the objective is to generate molecules belonging to a specific molecular domain or possessing desired properties. Here, we focus on generating molecules from a particular domain using the in-context learning abilities of LLMs. To achieve this, we assembled a dataset, referred to as the *molecular domains* dataset, that comprises a diverse range of molecular domains: natural products, click-chemistry, proteolysis-targeting chimera (PROTACs), DNA-encoded chemical libraries, approved and failed drugs, and bioactive compounds from various bioassays. Molecules from the same domain, are concatenated as a long sequence, and augmented through permutation during training. We split the dataset into training, validation, and test domains, following an 8:1:1 ratio (Figure 4, left). The validation and test sets contained molecules from unseen domains, enabling us to evaluate the models' conditional generation capabilities through in-context learning. We trained Chem-xLSTM, Mamba, Transformer++, and S4-based models with the CLM approach on the *molecular domains* with an increased context length of 4,096 tokens. The context length for S4 models was restricted to 2,048 due to memory constraints. We evaluated the models based on NTP loss across unseen domains. The trained model Chem-xLSTM-15M-icl shows promising results in this conditional setting, outperforming the other benchmarked model-classes (Figure 4, right). This demonstrates Chem-xLSTM's capability to generate molecules from an unseen chemical domain when provided with only a few exemplary molecules without fine-tuning. Further details and results are provided in Appx. E.2.

5 Limitations and Conclusions

Limitations. Manual hyperparameter selection may not yield optimal configurations of the models, and the reliance on character-level tokenization for DNA could restrict performance with larger context sizes. Additionally, the generalizability of our models across different organisms and chemical domains is uncertain due to biases in the training datasets. Metrics like perplexity, commonly used as performance proxies, may not fully capture the true capacity of the models (Appendix F).

Conclusion. Despite these limitations, Bio-xLSTM demonstrated effectiveness in DNA sequence modeling, performing best in masked and causal language tasks across context sizes. In protein modeling, Bio-xLSTM excelled at long-range modeling, becoming a promising approach for generating homologous proteins. For small molecules, Bio-xLSTM achieved the best FCD in unconditional generation and showed ICL capabilities. We have clarified how to tailor xLSTM for biological and chemical sequences and demonstrated how it competes with other domain-specific models (see Appendix G). Our findings underpin that Bio-xLSTM is a prime candidate for foundation models in molecular biology.

References

1000 Genomes Project Consortium. A map of human genome variation from population scale sequencing. *Nature*, 467(7319):1061, 2010.

Julián N Acosta, Guido J Falcone, Pranav Rajpurkar, and Eric J Topol. Multimodal biomedical ai. *Nature Medicine*, 28(9):1773–1784, 2022.

Sanjar Adilov. Generative pre-training from molecules. *ChemRxiv preprint chemrxiv-2021-5fwjd*, 2021.

Gustaf Ahdritz, Nazim Bouatta, Sachin Kadyan, Lukas Jarosch, Dan Berenberg, Ian Fisk, Andrew Watkins, Stephen Ra, Richard Bonneau, and Mohammed AlQuraishi. Openproteinset: Training data for structural biology at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 4597–4609. Curran Associates, Inc., 2023.

Walid Ahmad, Elana Simon, Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. ChemBERTa-2: Towards chemical foundation models. *arXiv preprint arXiv:2209.01712*, 2022.

Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-LSTM: xLSTM as generic vision backbone. *arXiv preprint arXiv:2406.04303*, 2024.

Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, December 2019. ISSN 1548-7091.

Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096): 223–230, 1973. doi: 10.1126/science.181.4096.223.

Frances H Arnold. Directed evolution: bringing new chemistry to life. *Angewandte Chemie*, 57(16): 4143, 2018.

Euan A Ashley. Towards precision medicine. *Nature Reviews Genetics*, 17(9):507–522, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. MolGPT: molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9): 2064–2076, 2021.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. *Neural Infomation Processing Systems (NeurIPS)*, 2024.

Gonzalo Benegas, Sanjit Singh Batra, and Yun S Song. DNA language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120 (44):e2311219120, 2023.

Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. *International Conference on Learning Representations (ICLR)*, 7, 2019.

Britta AM Bouwman and Wouter de Laat. Getting the genome in shape: the formation of loops, domains and compartments. *Genome Biology*, 16(1):154, August 2015. ISSN 1474-760X. doi: 10.1186/s13059-015-0730-1.

Andres M Bran and Philippe Schwaller. Transformers and large language models for chemistry and drug discovery. *arXiv preprint arXiv:2310.06083*, 2023.

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, 2022.

Nadav Brandes, Grant Goldman, Charlotte H Wang, Chun Jimmie Ye, and Vasilis Ntranos. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature Genetics*, 55 (9):1512–1522, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. ChemBERTa: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, volume 9, 2021.

Deanna M Church, Valerie A Schneider, Tina Graves, Katherine Auger, Fiona Cunningham, Nathan Bouk, Hsiu-Chuan Chen, Richa Agarwala, William M McLaren, Graham RS Ritchie, et al. Modernizing reference genome assemblies. *PLoS biology*, 9(7):e1001091, 2011.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *BioRxiv*, pp. 2023–01, 2023.

T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, volume 12, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, 2021.

Noelia Ferruz, Stefan Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13:4348, 2022.

Daniel Flam-Shepherd, Kevin Zhu, and Alán Aspuru-Guzik. Language models can learn complex molecular distributions. *Nature Communications*, 13(1):3293, 2022.

Qitao Geng, Runtao Yang, and Lina Zhang. A deep learning framework for enhancer prediction using word embedding and sequence generation. *Biophysical Chemistry*, 286:106822, 2022.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks ICANN '99*, pp. 850–855. IET, 1999.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, volume 10, 2022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Ho-Jin Gwak and Mina Rho. ViBE: a hierarchical BERT model to identify eukaryotic viruses using metagenome sequencing data. *Briefings in Bioinformatics*, 23(4):bbac204, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings CVPR*, pp. 770–778, 2016.

A Hoarfrost, A Aptekmann, G Farfañuk, and Y Bromberg. Deep learning of a bacterial and archaeal universal language of life enables transfer learning and illuminates microbial dark matter. *Nature Communications*, 13(1):2606, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Shion Honda, Shoi Shi, and Hiroki R Ueda. SMILES transformer: Pre-trained molecular fingerprint for low data drug discovery. *arXiv preprint arXiv:1911.04738*, 2019.

Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta PI Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, 35(2):128–135, 2017.

Bozhen Hu, Jun Xia, Jiangbin Zheng, Cheng Tan, Yufei Huang, Yongjie Xu, and Stan Z Li. Protein language models and structure prediction: Connection and progression. *arXiv preprint arXiv:2211.16742*, 2022.

Stanisław Jastrzębski, Damian Leśniak, and Wojciech Marian Czarnecki. Learning to SMILE(S). *arXiv preprint arXiv:1602.06289*, 2016.

Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Alexander Karollus, Johannes Hingerl, Dennis Gankin, Martin Grosshauser, Kristian Klemon, and Julien Gagneur. Species-aware DNA language models capture regulatory elements and their evolution. *Genome Biology*, 25(1):83, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pp. 5156–5165. PMLR, 13–18 Jul 2020.

Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2023 update. *Nucleic Acids Research*, 51 (D1):D1373–D1380, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 3, 2015.

Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William Fitzhugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

Doriana Levré, Chiara Arcisto, Valentina Mercalli, and Alberto Massarotti. Zinclick v. 18: expanding chemical space of 1, 2, 3-triazoles. *Journal of Chemical Information and Modeling*, 59(5): 1697–1702, 2018.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. Publisher: American Association for the Advancement of Science.

Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012.

Ali Madani, Ben Krause, Eric R. Greene, and et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41:1099–1106, 2023.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chemical science*, 9(24):5441–5451, 2018.

Eyal Mazuz, Guy Shtar, Bracha Shapira, and Lior Rokach. Molecule generation using transformers and policy gradient reinforcement learning. *Scientific Reports*, 13(1):8799, 2023.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 35492–35506. PMLR, 2024.

Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. ColabFold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, June 2022. ISSN 1548-7105. Publisher: Nature Publishing Group.

Venkata Chandrasekhar Nainala, Kohulan Rajan, Sri Ram Sagar Kanakam, Nisha Sharma, Viktor Weißenborn, Jonas Schaub, and Christoph Steinbeck. Coconut 2.0: A comprehensive overhaul and curation of the collection of open natural products database. *ChemRxiv preprint chemrxiv-2024-fxq2s*, 2024.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 43177–43201. Curran Associates, Inc., 2023.

Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. ProteinGym: Large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 64331–64379. Curran Associates, Inc., 2023.

Pascal Notin, Nathan Rollins, Yarin Gal, Chris Sander, and Debora Marks. Machine learning for functional protein design. *Nature Biotechnology*, 42(2):216–228, February 2024. ISSN 1546-1696. Publisher: Nature Publishing Group.

Mhaned Oubounyt, Zakaria Louadi, Hilal Tayara, and Kil To Chong. Deepromoter: robust promoter predictor using deep learning. *Frontiers in Genetics*, 10:286, 2019.

Rıza Özçelik, Sarah de Ruiter, Emanuele Criscuolo, and Francesca Grisoni. Chemical language modeling with structured state space sequence models. *Nature Communications*, 15(1):6176, 2024.

George Papadatos, Muhammad Alkarouri, Valerie J Gillet, Peter Willett, Visakan Kadirkamanathan, Christopher N Luscombe, Gianpaolo Bravi, Nicola J Richmond, Stephen D Pickett, Jameed Hussain, et al. Lead optimization using matched molecular pairs: inclusion of contextual information for enhanced prediction of herg inhibition, solubility, and lipophilicity. *Journal of Chemical Information and Modeling*, 50(10):1872–1886, 2010.

Tho Hoan Phaml, Dang Hung Tran, Tu Bao Ho, Kenji Satou, and Gabriel Valiente. Qualitatively predicting acetylation and methylation areas in DNA sequences. *Genome Informatics*, 16(2):3–11, 2005.

M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré. Hyena hierarchy: Towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 28043–28078. PMLR, 2023.

Ofir Press, Noah A. Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 1, pp. 5493–5505. Association for Computational Linguistics, 2021.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, 2018.

Ian K Quigley, Andrew Blevins, Brayden J Halverson, and Nate Wilkinson. Belka: The big encoded library for chemical assessment. In *NeurIPS 2024 Competition Track*, 2024.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with TAPE. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:9689–9701, 2019.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. MSA transformer. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pp. 8844–8856. PMLR, 2021.

Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, 15(10):816–822, 2018.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021. ISSN 0027-8424.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022.

Nicolas Scalzitti, Arnaud Kress, Romain Orhand, Thomas Weber, Luc Moulinier, Anne Jeannin-Girardon, Pierre Collet, Olivier Poch, and Julie D Thompson. Spliceator: multi-species splice site prediction using convolutional neural networks. *BMC Bioinformatics*, 22:1–26, 2021.

Yair Schiff, Chia Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range DNA sequence modeling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 43632–43648. PMLR, 2024.

Johannes Schimunek, Philipp Seidl, Lukas Friedrich, Daniel Kuhn, Friedrich Rippmann, Sepp Hochreiter, and Günter Klambauer. Context-enriched molecule representations improve few-shot drug discovery. In *International Conference on Learning Representations (ICLR)*, volume 11, 2023.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5(9):1572–1583, 2019.

Philippe Schwaller, Daniel Probst, Alain C Vaucher, Vishnu H Nair, David Kreutter, Teodoro Laino, and Jean-Louis Reymond. Mapping the space of chemical reactions using attention-based neural networks. *Nature Machine Intelligence*, 3(2):144–152, 2021.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1): 120–131, 2018.

Philipp Seidl, Andreu Vall, Sepp Hochreiter, and Günter Klambauer. Enhancing activity prediction models in drug discovery with the ability to understand human language. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 30458–30490. PMLR, 2023.

Damiano Sgarbossa, Cyril Malbranke, and Anne-Florence Bitbol. ProtMamba: a homology-aware but alignment-free protein state space model. *bioRxiv*, pp. 2024–05, 2024.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Reverse-complement parameter sharing improves deep learning models for genomics. *BioRxiv*, pp. 103663, 2017.

Ctibor Skuta, Martin Popr, Tomas Muller, Jindrich Jindrich, Michal Kahle, David Sedlak, Daniel Svozil, and Petr Bartunek. Probes & drugs portal: an interactive, open data resource for chemical biology. *Nature methods*, 14(8):759–760, 2017.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pp. 2377–2385. Curran Associates, Inc., 2015.

Megan Stanley, John F Bronskill, Krzysztof Maziarz, Hubert Misztela, Jessica Lanini, Marwin Segler, Nadine Schneider, and Marc Brockschmidt. FS-Mol: A few-shot learning dataset of molecules. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024a.

Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. SaProt: Protein language modeling with structure-aware vocabulary. In *International Conference on Learning Representations (ICLR)*, volume 12, 2024b.

The UniProt Consortium. Uniprot: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 2023.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Timothy Truong Jr and Tristan Bepler. PoET: A generative model of protein families as sequences-of-sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 77379–77415. Curran Associates, Inc., 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010.

Ruohan Wang, Zishuai Wang, Jianping Wang, and Shuaicheng Li. Splicefinder: ab initio prediction of splice sites using convolutional neural network. *BMC Bioinformatics*, 20:1–13, 2019a.

Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. SMILES-BERT: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pp. 429–436, 2019b.

Ye Wang, Honggang Zhao, Simone Sciabola, and Wenlu Wang. cMolGPT: A conditional generative pre-trained transformer for target-specific de novo molecular generation. *Molecules*, 28(11):4430, 2023.

David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28(1):31–36, February 1988. ISSN 1549-9596.

Yuxin Wu and Kaiming He. Group normalization. *International Journal of Computer Vision*, 128(3): 742–755, 2020.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, 16(8):687–694, 2019.

Meng Yang, Lichao Huang, Haiping Huang, Hui Tang, Nan Zhang, Huanming Yang, Jihong Wu, and Feng Mu. Integrating convolution and self-attention improves language model of human genome for interpreting non-coding regions at base-resolution. *Nucleic Acids Research*, 50(14):e81–e81, 2022.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 56501–56523. PMLR, 2024.

Barbara Zdrazil, Eloy Felix, Fiona Hunter, Emma J Manners, James Blackshaw, Sybilla Corbett, Marleen de Veij, Harris Ioannidis, David Mendez Lopez, Juan F Mosquera, Maria Paula Magarinos, Nicolas Bosc, Ricardo Arcila, Tevfik Kizilören, Anna Gaulton, A Patrícia Bento, Melissa F Adasme, Peter Monecke, Gregory A Landrum, and Andrew R Leach. The ChEMBL database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic Acids Research*, 52(D1):D1180–D1192, 2023.

Qiang Zhang, Keyang Ding, Tianwen Lyv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, et al. Scientific large language models: A survey on biological & chemical domains. *arXiv preprint arXiv:2401.14656*, 2024.

Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, Arip Asadulaev, et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nature Biotechnology*, 37(9):1038–1040, 2019.

Hannah Zhou, Avanti Shrikumar, and Anshul Kundaje. Towards a better understanding of reverse-complement equivariance for deep learning models in genomics. In *Machine Learning in Computational Biology*, pp. 1–33. PMLR, 2022.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana V Davuluri, and Han Liu. DNABERT-2: Efficient foundation model and benchmark for multi-species genomes. In *International Conference on Learning Representations (ICLR)*, volume 12, 2024.

Contents

A Related Work **18**

B xLSTM Architecture Details **20**
B.1 sLSTM 20
B.2 mLSTM 20
B.3 xLSTM and Bio-xLSTM Blocks 20
B.4 Modes of Operation: Parallel, Chunkwise or Recurrent 21
B.5 Efficient Bidirectional Modeling for Weight-Tied Layers of Bio-xLSTM 22

C DNA-xLSTM: Details and Additional Results **23**
C.1 Pre-Training 23
C.2 Downstream Tasks 23
C.3 Architecture and Hyperparameters 25
C.4 Reverse-Complement Invariance 25
C.5 Implementation Details 26

D Prot-xLSTM: Details and Additional Results **28**
D.1 Homology-Aware Training 28
D.2 ICL: Homology-Conditioned Protein Generation 29
D.3 Protein Variant Fitness Prediction 30

E Chem-xLSTM: Details and Additional Results **33**
E.1 Unconditional Molecule Generation 33
E.2 Conditional Molecule Generation with In-Context Learning 33
E.3 Architecture and Hyperparameter Selection 34
E.4 Implementation Details 35
E.5 Additional Results 35

F Limitations **36**

G Conclusions **36**

A Related Work

In all three areas, genomics, proteomics, and chemistry, we observe a similar trend that until around 2018 the language models were based on LSTMs (Hochreiter & Schmidhuber, 1997), then a large amount of models were based on Transformers (Vaswani et al., 2017), with different training paradigms and styles, and from 2023 onwards the first state-space models appeared.

Language models for genomic sequence data. DNABERT (Ji et al., 2021) and its successor DNABERT-2 (Zhou et al., 2024) are Transformer-based models that leverage bidirectional encoder representations and masked language modeling to capture nucleotide context, achieving high performance in tasks like promoter and splice site prediction. LOGO (Yang et al., 2022), another Transformer-based model, applies self-supervised learning to the human genome for sequence labeling and variant prioritization, while VIBE (Gwak & Rho, 2022) employs a hierarchical BERT model to enhance the detection of eukaryotic viruses in metagenomic data. Models like LookingGlass (Hoarfrost et al., 2022), based on recurrent neural network (RNN), and GPN (Benegas et al., 2023), which uses convolutional neural networks (CNNs), are examples of non-Transformer-based approaches, with LookingGlass focusing on microbial genomes and GPN on plant genomes. More recent developments include the nucleotide transformer (NT) (Dalla-Torre et al., 2023), a Transformer model trained on the human genome and data from the 1000 Genomes Project, and SpeciesLM (Karollus et al., 2024), which trains Transformer-based models on 1500 fungal genomes. The latest advances, represented by Caduceus (Schiff et al., 2024) based on Mamba (Gu & Dao, 2023) and HyenaDNA (Nguyen et al., 2023), introduce SSMs that allow generative modeling and representation learning for long DNA sequences.

Language models for protein sequence data. Until around 2019, the field was dominated by RNNs and LSTM-based models trained with CLM. Notable examples include UniRep (Alley et al., 2019), which employed multiplicative-LSTM to capture rich protein representations, and SSA (Bepler & Berger, 2019), which used bidirectional RNNs for structural similarity prediction. Since then the field has shifted towards Transformer-based models and MLM, driven by their success in natural language processing. Early adopters of this shift included the TAPE benchmark for protein downstream tasks (Rao et al., 2019), which evaluated both an LSTM and a Transformer architecture trained with CLM and MLM, respectively. Elnaggar et al. (2021) further expanded the use of Transformers with large-scale MLM, setting new benchmarks in protein sequence analysis with Prot-T5. ESM (Rives et al., 2021) applied MLM to a Transformer on a massive scale, capturing evolutionary patterns across diverse protein sequences. Other significant Transformer-based models include MSA-Transformer (Rao et al., 2021), which applied MLM to multiple-sequence alignments (MSA), and ProGen (Madani et al., 2023), which used CLM and Transformers for controlled protein sequence generation. Additionally, models like ProtGPT2 (Ferruz et al., 2022) and ProteinBERT (Brandes et al., 2022) utilized the power of Transformer architectures in generating novel protein sequences and functional predictions. Furthermore, (Su et al., 2024b) introduced a "structure-aware vocabulary" which they use as input for Transformer-based models. The recently proposed PoET (Truong Jr & Bepler, 2023) is an autoregressive Transformer model trained on non-aligned homologous sequences, providing a novel approach for conditional protein design and protein fitness prediction. Building on the concept of non-aligned homologous sequences, ProtMamba (Sgarbossa et al., 2024) leverages emerging SSMs to manage long-context conditioning on proteins, effectively utilizing autoregressive and FIM strategies. For a more comprehensive review of these advancements, including their applications in functional protein design, see Notin et al. (2024) and Hu et al. (2022).

Language models for chemical sequence data. The first language model for chemical sequences was an LSTM-based, autoregressive method developed by Segler et al. (2018), which demonstrated that the SMILES syntax (Weininger, 1988) and generation of realistic organic molecules can be learned. Honda et al. (2019) introduced a Transformer model for this task, showing that this leads to informative representations of molecules. The Molecular Transformer (Schwaller et al., 2019) consists of a Transformer-based encoder and decoder, trained on chemical reaction data to translate between reactants and products. More recently, SSMs have been used for generative modeling of SMILES strings (Özçelik et al., 2024). Subsequent models such as MolGPT (Bagal et al., 2021) and cMolGPT (Wang et al., 2023) utilized the GPT architecture to generate SMILES strings, with MolGPT conditioning on chemical properties and scaffolds, and cMolGPT focusing on biomolecular targets. Transformer-based approaches have also been employed to optimize the properties of small molecules in a reinforcement-learning setting (Mazuz et al., 2023). Encoder-style language models for chemistry, such as SmilesLSTM (Mayr et al., 2018), ChemNet (Preuer et al., 2018), and CNN-based models (Jastrzębski et al., 2016), initially used activity and property prediction as pre-training or training objectives. Later, these encoder-style language models were trained with the masking language modeling objective, as seen in ChemBERTA (Chithrananda et al., 2020), Chemberta-2 (Ahmad et al., 2022), SMILES-BERT (Wang et al., 2019b), MolFormer (Ross et al., 2022) and rxnfp-BERT (Schwaller et al., 2021). Some models have also adopted contrastive objectives (Seidl et al., 2023). Large language models for molecules have also been shown to learn complex molecular distributions (Flam-Shepherd et al., 2022). For a more thorough and comprehensive overview, we refer to the surveys by Bran & Schwaller (2023) and Zhang et al. (2024)

B xLSTM Architecture Details

B.1 sLSTM

The forward pass of the sLSTM forward in the vectorized version is:

$$\boldsymbol{c}_t = \mathbf{f}_t \odot \boldsymbol{c}_{t-1} + \mathbf{i}_t \odot \boldsymbol{z}_t \qquad \text{cell state} \tag{3}$$

$$\boldsymbol{n}_t = \mathbf{f}_t \odot \boldsymbol{n}_{t-1} + \mathbf{i}_t \qquad \text{normalizer state} \tag{4}$$

$$\boldsymbol{h}_t = \mathbf{o}_t \odot \tilde{\boldsymbol{h}}_t \,, \qquad \tilde{\boldsymbol{h}}_t = \boldsymbol{c}_t \odot \boldsymbol{n}_t^{-1} \qquad \text{hidden state} \tag{5}$$

$$\boldsymbol{z}_t = \varphi\left(\tilde{\boldsymbol{z}}_t\right) \,, \qquad \tilde{\boldsymbol{z}}_t = \boldsymbol{W}_{\boldsymbol{z}}\, \boldsymbol{x}_t + \boldsymbol{R}_{\boldsymbol{z}}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{z}} \qquad \text{cell input} \tag{6}$$

$$\mathbf{i}_t = \exp\left(\tilde{\mathbf{i}}_t\right) \,, \qquad \tilde{\mathbf{i}}_t = \boldsymbol{W}_{\mathbf{i}}\, \boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{i}}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{i}} \qquad \text{input gate} \tag{7}$$

$$\mathbf{f}_t = \exp\left(\tilde{\mathbf{f}}_t\right) \text{ OR } \sigma\left(\tilde{\mathbf{f}}_t\right) \,, \qquad \tilde{\mathbf{f}}_t = \boldsymbol{W}_{\mathbf{f}}\, \boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{f}}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{f}} \qquad \text{forget gate} \tag{8}$$

$$\mathbf{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right) \,, \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_{\mathbf{o}}\, \boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{o}}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{o}} \qquad \text{output gate,} \tag{9}$$

where $\mathbf{i}_t, \mathbf{o}_t, \mathbf{f}_t \in \mathbb{R}^d$ are the input, output and forget gate, respectively, $\boldsymbol{W}_{\boldsymbol{z}}, \boldsymbol{W}_{\mathbf{i}}, \boldsymbol{W}_{\mathbf{f}}, \boldsymbol{W}_{\mathbf{o}} \in \mathbb{R}^{d \times D}$, $\boldsymbol{R}_{\boldsymbol{z}}, \boldsymbol{R}_{\mathbf{i}}, \boldsymbol{R}_{\mathbf{f}}, \boldsymbol{R}_{\mathbf{o}} \in \mathbb{R}^{d \times d}$, and $\boldsymbol{b}_{\boldsymbol{z}}, \boldsymbol{b}_{\mathbf{i}}, \boldsymbol{b}_{\mathbf{f}}, \boldsymbol{b}_{\mathbf{o}} \in \mathbb{R}^d$ are trainable weight matrices and biases.

B.2 mLSTM

The forward pass of the mLSTM is defined as follows:

$$\boldsymbol{C}_t = \mathrm{f}_t\, \boldsymbol{C}_{t-1} + \mathrm{i}_t\, \boldsymbol{v}_t\, \boldsymbol{k}_t^\top \qquad \text{cell state} \tag{10}$$

$$\boldsymbol{n}_t = \mathrm{f}_t\, \boldsymbol{n}_{t-1} + \mathrm{i}_t\, \boldsymbol{k}_t \qquad \text{normalizer state} \tag{11}$$

$$\boldsymbol{h}_t = \mathbf{o}_t \odot \tilde{\boldsymbol{h}}_t \,, \qquad \tilde{\boldsymbol{h}}_t = \boldsymbol{C}_t \boldsymbol{q}_t \,/\, \max\left\{\left|\boldsymbol{n}_t^\top \boldsymbol{q}_t\right|, 1\right\} \qquad \text{hidden state} \tag{12}$$

$$\boldsymbol{q}_t = \boldsymbol{W}_q\, \boldsymbol{x}_t + \boldsymbol{b}_q \qquad \text{query input} \tag{13}$$

$$\boldsymbol{k}_t = \frac{1}{\sqrt{d}} \boldsymbol{W}_k\, \boldsymbol{x}_t + \boldsymbol{b}_k \qquad \text{key input} \tag{14}$$

$$\boldsymbol{v}_t = \boldsymbol{W}_v\, \boldsymbol{x}_t + \boldsymbol{b}_v \qquad \text{value input} \tag{15}$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_{\mathrm{i}}^\top\, \boldsymbol{x}_t + b_{\mathrm{i}} \qquad \text{input gate} \tag{16}$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right) \,, \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_{\mathrm{f}}^\top\, \boldsymbol{x}_t + b_{\mathrm{f}} \qquad \text{forget gate} \tag{17}$$

$$\mathbf{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right) \,, \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_{\mathbf{o}}\, \boldsymbol{x}_t + \boldsymbol{b}_{\mathbf{o}} \qquad \text{output gate} \tag{18}$$

where $\mathrm{i}_t, \mathrm{o}_t, \mathrm{f}_t \in \mathbb{R}$ are the input, output and forget gate, respectively, $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t \in \mathbb{R}^d$ are query, key and value inputs with trainable weight matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times D}$, $\boldsymbol{w}_{\mathrm{i}}, \boldsymbol{w}_{\mathrm{f}} \in \mathbb{R}^D$ are input and forget gate weights and the respective $b_{\mathrm{i}}, b_{\mathrm{f}} \in \mathbb{R}$ biases. All other quantities are identical to sLSTM.

B.3 xLSTM and Bio-xLSTM Blocks

Beck et al. (2024) suggested xLSTM blocks, which are residual (Srivastava et al., 2015; He et al., 2016) block modules, into which the sLSTM and mLSTM layers can be integrated. The two basic blocks can in principle be characterized by either applying post-sLSTM/mLSTM up- and down-projections (similar to Vaswani et al. (2017)) or by applying pre-sLSTM/mLSTM up-projections and post-sLSTM/mLSTM down-projections (similar to Dao (2024)). An sLSTM block integrates the sLSTM layer into the up- and down-projection block, while the mLSTM block integrates the mLSTM layer into the pre-up-projection and post-down-projection block. The two basic xLSTM blocks also make use of neural network modules like layer normalization (Ba et al., 2016), short causal convolutions, and, group normalization (Wu & He, 2020). For the exact architecture of the blocks, we refer to Beck et al. (2024, Sec.2.4). An xLSTM architecture is constructed by residually stacking the suggested xLSTM blocks. For that, the most commonly used pre-LayerNorm residual backbone is used.

For Bio-xLSTM we keep the basic xLSTM building blocks and the basic xLSTM architecture template, but adjust them to the respective domains. Figure A1 depicts sLSTM and mLSTM blocks, as well as, a bidirectional mLSTM configuration with weight-tied layers.

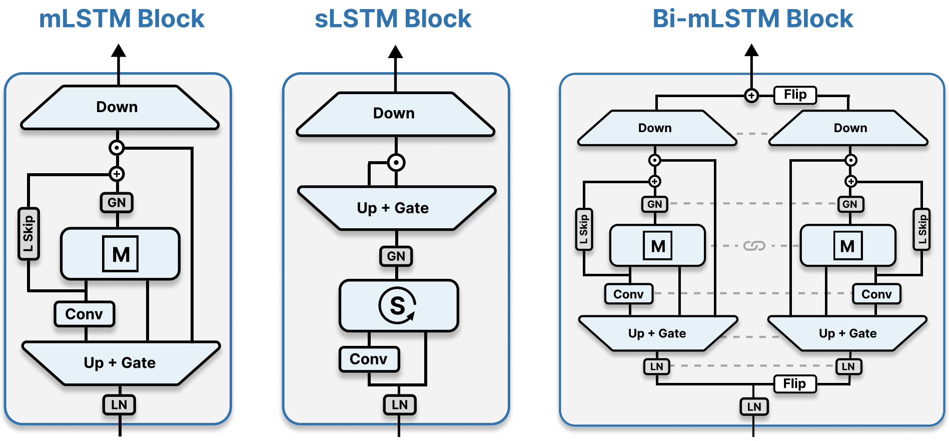

Figure A1: xLSTM and Bio-xLSTM blocks. **Left: mLSTM block.** *LN* (Layer Normalization) and *GN* (Group Normalization) refer to normalization modules, while *L Skip* represents learnable skip connections and *Conv* denotes causal 1D convolutions. The mLSTM block utilizes a gated pre-up-projection structure, akin to modern state-space models, with gates activated by the Swish function. **Middle: sLSTM block.** The sLSTM block features a GELU-gated post-up-projection structure, similar to Transformer architectures. **Right: Bidirectional mLSTM block.** For bidirectional processing, the xLSTM applies each block to the input sequence twice before combining the outputs: once left-to-right and once right-to-left.

B.4 Modes of Operation: Parallel, Chunkwise or Recurrent

Similar to linear attention variants (Katharopoulos et al., 2020; Yang et al., 2024), the mLSTM has three possible forms: parallel, recurrent or chunkwise. The presentation in section B.2 (and Beck et al., 2024) focuses on the recurrent form:

$$\begin{aligned}
\boldsymbol{C}_t &= \sigma(\tilde{f}_t)\,\boldsymbol{C}_{t-1} + \exp(\tilde{i}_t) \odot \boldsymbol{v}_t \boldsymbol{k}_t^\top \\
\boldsymbol{n}_t &= \sigma(\tilde{f}_t)\,\boldsymbol{n}_{t-1} + \exp(\tilde{i}_t)\,\boldsymbol{k}_t \\
\boldsymbol{h}_t &= \sigma(\tilde{\boldsymbol{o}}_t) \odot \frac{\boldsymbol{C}_t \boldsymbol{q}_t}{\max\big(|\boldsymbol{n}_t \boldsymbol{q}_t|, 1\big)}.
\end{aligned}$$

This form is especially useful for inference when samples arrive one time-step at a time.

The omission of the recurrent connections in mLSTM allows for a parallel implementation (Beck et al., 2024, Appendix):

$$\begin{aligned}
\tilde{\boldsymbol{F}} &= \begin{bmatrix}
0 & 0 & 0 & \dots & 0 \\
\ln\sigma(\tilde{f}_2) & 0 & 0 & \dots & 0 \\
\ln\sigma(\tilde{f}_2) + \ln\sigma(\tilde{f}_3) & \ln\sigma(\tilde{f}_3) & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\sum_{t=2}^{T} \ln\sigma(\tilde{f}_t) & \sum_{t=3}^{T} \ln\sigma(\tilde{f}_t) & \sum_{t=4}^{T} \ln\sigma(\tilde{f}_t) & \dots & 0
\end{bmatrix} \\
\boldsymbol{D} &= \exp\big(\tilde{\boldsymbol{F}} + \mathbf{1} \otimes \tilde{\boldsymbol{i}}\big) \odot \boldsymbol{M} \\
\boldsymbol{H} &= \sigma(\tilde{\boldsymbol{O}}) \odot \frac{\boldsymbol{D} \odot \boldsymbol{Q}\boldsymbol{K}^\top}{\max\big(|(\boldsymbol{D} \odot \boldsymbol{Q}\boldsymbol{K}^\top) \cdot \mathbf{1}|, \mathbf{1}\big)} \boldsymbol{V},
\end{aligned}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \tilde{\boldsymbol{O}} \in \mathbb{R}^{T \times d}$, $\tilde{\boldsymbol{i}} \in \mathbb{R}^T$ and $\boldsymbol{M} \in \{0, 1\}^{T \times T}$ is a causal (i.e. lower-triangular) masking matrix. The $\otimes$ refers to an outer product, while $\odot$ is a Hadamard (i.e. element-wise) product. The fraction, $\max$ and other non-linear functions are also applied element-wise. This parallel form enables an efficient training regime, similar to Transformers.

The chunkwise implementation is a hybrid of the recurrent and parallel forms:

$$
\begin{aligned}
\tilde{\boldsymbol{F}} &= \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \ln\sigma(\tilde{f}_{t-C+2}) & 0 & 0 & \dots & 0 \\ \ln\sigma(\tilde{f}_{t-C+2}) + \ln\sigma(\tilde{f}_{t-C+3}) & \ln\sigma(\tilde{f}_{t-C+3}) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{\tau=2}^{C} \ln\sigma(\tilde{f}_{t-C+\tau}) & \sum_{\tau=3}^{C} \ln\sigma(\tilde{f}_{t-C+\tau}) & \sum_{\tau=4}^{C} \ln\sigma(\tilde{f}_{t-C+\tau}) & \dots & 0 \end{bmatrix} \\
\boldsymbol{D} &= \exp\big(\tilde{\boldsymbol{F}} + \boldsymbol{1} \otimes \tilde{\boldsymbol{i}}\big) \odot \boldsymbol{M} \\
\boldsymbol{f} &= \left(\sigma(\tilde{f}_{t-C+1}), \sigma(\tilde{f}_{t-C+1})\,\sigma(\tilde{f}_{t-C+2}), \dots, \prod_{\tau=1}^{C} \sigma(\tilde{f}_{t-C+\tau})\right) \\
\boldsymbol{H} &= \sigma(\tilde{\boldsymbol{O}}) \odot \frac{(\boldsymbol{D} \odot \boldsymbol{Q}\boldsymbol{K}^{\top})\,\boldsymbol{V} + \mathrm{diag}(\boldsymbol{f})\,\boldsymbol{Q}\boldsymbol{C}_{t-C}^{\top}}{\max\big(|(\boldsymbol{D} \odot \boldsymbol{Q}\boldsymbol{K}^{\top}) \cdot \boldsymbol{1} + \mathrm{diag}(\boldsymbol{f})\,\boldsymbol{Q}\boldsymbol{n}_{t-C}|, \boldsymbol{1}\big)} \\
\boldsymbol{C}_t &= \left(\prod_{\tau=1}^{C} \sigma(\tilde{f}_{t-C+\tau})\right) \boldsymbol{C}_{t-C} + \boldsymbol{V}^{\top}\,\mathrm{diag}(\boldsymbol{d}_C)\,\boldsymbol{K} \\
\boldsymbol{n}_t &= \left(\prod_{\tau=1}^{C} \sigma(\tilde{f}_{t-C+\tau})\right) \boldsymbol{n}_{t-C} + \boldsymbol{K}^{\top}\boldsymbol{d}_C,
\end{aligned}
$$

with $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \tilde{\boldsymbol{O}} \in \mathbb{R}^{C \times d}$ and $\tilde{\boldsymbol{i}} \in \mathbb{R}^{C}$ the pre-activations from $t - C + 1$ to t. Furthermore, $\boldsymbol{M} \in \{0, 1\}^{C \times C}$, is a local causal (i.e. lower-triangular) masking matrix, $\boldsymbol{d}_C$ denotes the last row of $\boldsymbol{D}$, diag transforms a vector into a diagonal matrix, and C is the chunk size. For $C = 1$, we recover the recurrent form, whereas for $C = T$, we obtain the parallel form.

B.5 Efficient Bidirectional Modeling for Weight-Tied Layers of Bio-xLSTM

Bidirectional modeling is often required to learn representations of biological and chemical sequences, for example with the MLM paradigm. The default approach for bidirectional modeling would be to use an mLSTM layer on the usual sequence $\boldsymbol{X}_{1:T} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T)$ and then applying a weight-tied layer on the reversed sequence $\boldsymbol{X}_{T:1} = (\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, \dots, \boldsymbol{x}_1)$ and subsequently summing those outputs:

$$
\begin{aligned}
\boldsymbol{H}^{+} &= \mathrm{mLSTM}(\boldsymbol{X}_{1:T}; \boldsymbol{w}) && (19) \\
\boldsymbol{H}^{-} &= \mathrm{mLSTM}(\boldsymbol{X}_{T:1}; \boldsymbol{w}) && (20) \\
\boldsymbol{H} &= \boldsymbol{H}^{+} + \boldsymbol{H}^{-}_{T:1}, && (21)
\end{aligned}
$$

where $\boldsymbol{H}^{-}_{T:1}$ indicates that the sequence is reversed again, and $\boldsymbol{w}$ are the parameters of the LSTM-layer $\mathrm{mLSTM}(\boldsymbol{X}_{1:T}; \boldsymbol{w})$ which are assumed to be the same for both directions, i.e. weight-tied. This approach is schematically depicted in Figure A1 (Right). However, this approach is inefficient with respect to memory and operations because it has to calculate and store all internal quantities, such as the cell states, twice for the backward pass. A variant of this approach is to perform the forward direction in one block (Eq. 19) and the reverse direction in a consecutive block (Eq. 20) of the architecture (Alkin et al., 2024).

We propose an efficient bidirectional modeling approach. Because of the parallelism of mLSTM and its gates depending only on the current time step, the weighted cumulative sum required for the cell state (Eq. 10), can be done backwards to obtain the representations for the reversed sequence

$$
\begin{aligned}
\boldsymbol{C}_t^{+} &= \mathrm{f}_t \boldsymbol{C}_{t-1}^{+} + \mathrm{i}_t \boldsymbol{v}_t\, \boldsymbol{k}_t^{\top} & \qquad \boldsymbol{C}_t^{-} &= \mathrm{f}_t \boldsymbol{C}_{t+1}^{-} + \mathrm{i}_t \boldsymbol{v}_t\, \boldsymbol{k}_t^{\top} && (22) \\
\boldsymbol{n}_t^{+} &= \mathrm{f}_t \boldsymbol{n}_{t-1}^{+} + \mathrm{i}_t \boldsymbol{k}_t & \qquad \boldsymbol{n}_t^{-} &= \mathrm{f}_t \boldsymbol{n}_{t+1}^{-} + \mathrm{i}_t \boldsymbol{k}_t. && (23)
\end{aligned}
$$

The resulting $\boldsymbol{h}_t = \boldsymbol{h}_t^{+} + \boldsymbol{h}_t^{-}$ is a bidirectional representation of the input sequence, whereby this variant is more efficient with respect to memory usage because of shared quantities. Note that the two variants, the default approach, and the efficient approach, are mathematically equivalent.

C DNA-xLSTM: Details and Additional Results

In this section, we provide further details regarding the architecture, training setup, and evaluation metrics for the DNA-xLSTM models.

C.1 Pre-Training

Experimental setup. We followed the experimental protocol established in Schiff et al. (2024) and Nguyen et al. (2023). The human reference genome (Church et al., 2011) was used as the training dataset for two main tasks: **a)** causal language modeling (CLM) and **b)** masked language modeling (MLM). We employed context lengths of 1,024 and 32,000 tokens for these tasks.

To ensure a fair comparison with previous methods, such as Schiff et al. (2024), we used character- or base pair-level tokenization, training models with parameter sizes ranging from 500k to 4M. This experimental setup enabled us to evaluate model performance for both **a)** generative modeling of DNA sequences and **b)** learning rich DNA sequence representations—core tasks in this domain.

Methods and hyperparameters. In our pre-training experiments, we compared several architectures: a Transformer variant based on the Llama architecture, referred to as Transformer++ (Touvron et al., 2023), DNA-xLSTM, HyenaDNA (Nguyen et al., 2023), and DNA-Mamba (also known as Caduceus) (Schiff et al., 2024). Each architecture was trained under both CLM and MLM settings. Additionally, we assessed two types of reverse-complement (RC) equivariant models when applicable: DNA-Mamba-PH and DNA-Mamba-PS, as well as DNA-xLSTM-PH and DNA-xLSTM-PS. For non-equivariant models, reverse-complement augmentation was applied, following the approach described in Schiff et al. (2024). Further details on RC-equivariant modeling can be found in Section C.4. The hyperparameters for DNA-xLSTM and Transformer++ were optimized using a validation set, with the final configurations reported in Appendix Tables A4 and A3.

Metrics. We report cross-entropy loss on a held-out test set for both CLM and MLM pre-training experiments.

Results. Our experiments show that the sLSTM-based DNA-xLSTM-2M model, trained with a context size of 1,024 and reverse-complement augmentation, outperforms DNA-Mamba (Schiff et al., 2024), HyenaDNA (Nguyen et al., 2023), and Transformer++ across both CLM and MLM tasks. Notably, the performance gap between DNA-xLSTM and the baseline models increases in the MLM setting. See Figure 2.

We further enhanced DNA-xLSTM-500k and DNA-xLSTM-2M models by incorporating reverse-complement equivariance via parameter sharing. For smaller models, we achieved MLM losses comparable to DNA-Mamba-PS, with a significant improvement over DNA-Mamba-PS as model size scaled to 2M parameters (Figure A3). Additionally, we pre-trained a long-range DNA-xLSTM model based on mLSTM, with a context size of 32k, using both CLM and MLM objectives. This model achieved the lowest cross-entropy loss in both tasks, outperforming Transformers and HyenaDNA, while performing comparably to Mamba (Figure A2).

C.2 Downstream Tasks

Experimental setup. Two sets of downstream tasks were used for evaluating the learned representations: the Genomic benchmark (Grešová et al., 2023) and the Nucleotide Transformers Tasks (Dalla-Torre et al., 2023), which is a collection of 18 datasets derived from five peer-reviewed studies (Phaml et al., 2005; Oubounyt et al., 2019; Wang et al., 2019a; Scalzitti et al., 2021; Geng et al., 2022). These classification tasks were selected to determine how rich the learned representations of the architectures are. To extract representations from the pre-trained xLSTM-DNA models, we perform average pooling on the activations from the final xLSTM block. For each downstream dataset, these representations served as inputs to a task-specific classification head that were jointly fine-tuned with the pre-trained model parameters.

Methods and hyperparameters. For Nucleaotide Transformer tasks, we compared HyenaDNA, DNA-Mamba, and xLSTM-based models pre-trained with 2M parameters. For Genomic benchmark tasks, we compare the smaller xLSTM-500k against Mamba. In both settings, models were pre-trained with a context size of 1,024.

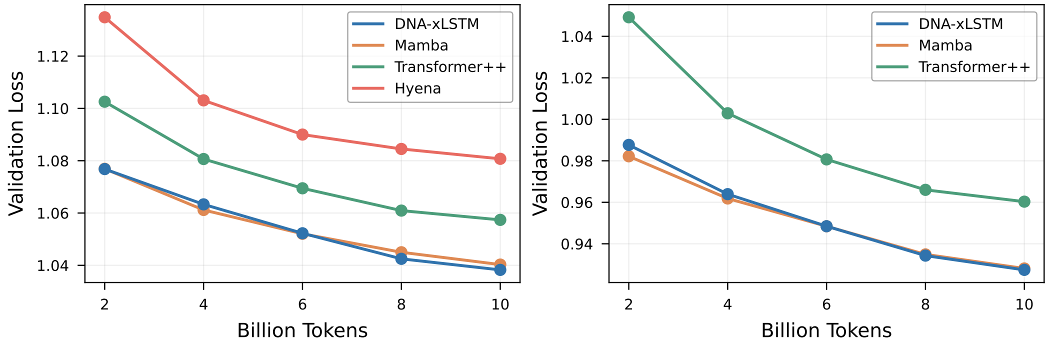

Figure A2: Pre-training of 4M-parameter DNA models on the human reference genome (GRCh38). The models are trained on the human reference genome at single-nucleotide resolution with a context length of 32k bases. **Left: causal language modeling.** Learning curves display **CLM loss** (↓) on a held-out test set, plotted against the number of tokens processed. **Right: masked language modeling.** Learning curves for bidirectional models trained with the **MLM** objective (↓). The DNA-xLSTM-4M model outperforms both Transformer++ and Hyena-DNA models of similar size, and matches the performance of Caduceus-4M.

Table A1: Downstream adaption of DNA models (extended version). The test set performance of DNA models with 2M parameters and models with over 100M parameters, fine-tuned on Nucleotide Transformer classification tasks, is shown. Models marked with PS or PH are trained to be RC equivariant. The used metric is provided in the *Metric* column and best values are highlighted in green Results are averaged over 10 random seeds, with error bars representing the difference between the maximum and minimum values across the runs. The best scores are highlighted in green. xLSTM-DNA-PH with 2M parameters outperforms similarly sized Hyena- and Mamba-based models, while achieving comparable results to the much larger Nucleotide Transformer. Scores for all models except xLSTM were obtained from Schiff et al. (2024).

Task	Metric	> 100M Param. Models			2M Param. Models				
		Enformer (252M)	DNABERT-2 (117M)	NT-v2 (500M)	HyenaDNA	Mamba-PS	Mamba-PH	xLSTM-PS	xLSTM-PH
Histone Markers									
H3	MCC ↑	$0.719^{\pm0.048}$	$0.785^{\pm0.033}$	$0.784^{\pm0.047}$	$0.779^{\pm0.037}$	$0.799^{\pm0.029}$	$0.815^{\pm0.048}$	$0.796^{\pm0.014}$	$0.824^{\pm0.010}$
H3K14AC	MCC ↑	$0.288^{\pm0.077}$	$0.516^{\pm0.028}$	$0.551^{\pm0.021}$	$0.612^{\pm0.065}$	$0.541^{\pm0.212}$	$0.631^{\pm0.026}$	$0.570^{\pm0.008}$	$0.598^{\pm0.017}$
H3K36ME3	MCC ↑	$0.344^{\pm0.055}$	$0.591^{\pm0.020}$	$0.625^{\pm0.030}$	$0.613^{\pm0.041}$	$0.609^{\pm0.109}$	$0.601^{\pm0.129}$	$0.588^{\pm0.019}$	$0.625^{\pm0.010}$
H3K4ME1	MCC ↑	$0.291^{\pm0.061}$	$0.511^{\pm0.028}$	$0.550^{\pm0.021}$	$0.512^{\pm0.024}$	$0.488^{\pm0.102}$	$0.523^{\pm0.039}$	$0.490^{\pm0.012}$	$0.526^{\pm0.001}$
H3K4ME2	MCC ↑	$0.211^{\pm0.069}$	$0.336^{\pm0.040}$	$0.319^{\pm0.045}$	$0.455^{\pm0.095}$	$0.388^{\pm0.101}$	$0.487^{\pm0.170}$	$0.489^{\pm0.024}$	$0.504^{\pm0.012}$
H3K4ME3	MCC ↑	$0.158^{\pm0.072}$	$0.352^{\pm0.077}$	$0.410^{\pm0.033}$	$0.549^{\pm0.056}$	$0.440^{\pm0.202}$	$0.544^{\pm0.045}$	$0.520^{\pm0.019}$	$0.537^{\pm0.012}$
H3K79ME3	MCC ↑	$0.496^{\pm0.042}$	$0.613^{\pm0.030}$	$0.626^{\pm0.046}$	$0.672^{\pm0.048}$	$0.676^{\pm0.026}$	$0.697^{\pm0.077}$	$0.662^{\pm0.011}$	$0.697^{\pm0.007}$
H3K9AC	MCC ↑	$0.420^{\pm0.063}$	$0.542^{\pm0.029}$	$0.562^{\pm0.040}$	$0.581^{\pm0.061}$	$0.604^{\pm0.048}$	$0.622^{\pm0.030}$	$0.622^{\pm0.013}$	$0.627^{\pm0.008}$
H4	MCC ↑	$0.732^{\pm0.076}$	$0.796^{\pm0.027}$	$0.799^{\pm0.025}$	$0.763^{\pm0.044}$	$0.789^{\pm0.020}$	$0.811^{\pm0.022}$	$0.793^{\pm0.011}$	$0.813^{\pm0.008}$
H4AC	MCC ↑	$0.273^{\pm0.063}$	$0.463^{\pm0.041}$	$0.495^{\pm0.032}$	$0.564^{\pm0.038}$	$0.525^{\pm0.240}$	$0.621^{\pm0.054}$	$0.558^{\pm0.018}$	$0.583^{\pm0.014}$
Regulatory Annotation									
Enhancer	MCC ↑	$0.451^{\pm0.108}$	$0.516^{\pm0.098}$	$0.548^{\pm0.144}$	$0.517^{\pm0.117}$	$0.491^{\pm0.066}$	$0.546^{\pm0.073}$	$0.375^{\pm0.030}$	$0.545^{\pm0.024}$
Enhancer Types	MCC ↑	$0.309^{\pm0.134}$	$0.423^{\pm0.051}$	$0.424^{\pm0.132}$	$0.386^{\pm0.185}$	$0.416^{\pm0.095}$	$0.439^{\pm0.054}$	$0.444^{\pm0.046}$	$0.466^{\pm0.011}$
Promoter: All	F1 ↑	$0.954^{\pm0.006}$	$0.971^{\pm0.006}$	$0.976^{\pm0.006}$	$0.960^{\pm0.005}$	$0.967^{\pm0.004}$	$0.970^{\pm0.004}$	$0.962^{\pm0.002}$	$0.967^{\pm0.001}$
NonTATA	F1 ↑	$0.955^{\pm0.010}$	$0.972^{\pm0.005}$	$0.976^{\pm0.006}$	$0.959^{\pm0.011}$	$0.968^{\pm0.006}$	$0.968^{\pm0.010}$	$0.963^{\pm0.002}$	$0.970^{\pm0.001}$
TATA	F1 ↑	$0.960^{\pm0.023}$	$0.955^{\pm0.021}$	$0.966^{\pm0.013}$	$0.944^{\pm0.040}$	$0.957^{\pm0.015}$	$0.953^{\pm0.016}$	$0.948^{\pm0.006}$	$0.952^{\pm0.005}$
Splice Site Annotation									
All	Accuracy ↑	$0.848^{\pm0.019}$	$0.939^{\pm0.009}$	$0.983^{\pm0.008}$	$0.956^{\pm0.011}$	$0.927^{\pm0.021}$	$0.940^{\pm0.027}$	$0.965^{\pm0.006}$	$0.974^{\pm0.004}$
Acceptor	F1 ↑	$0.914^{\pm0.028}$	$0.975^{\pm0.006}$	$0.981^{\pm0.011}$	$0.958^{\pm0.010}$	$0.936^{\pm0.077}$	$0.937^{\pm0.033}$	$0.970^{\pm0.005}$	$0.953^{\pm0.008}$
Donor	F1 ↑	$0.906^{\pm0.027}$	$0.963^{\pm0.006}$	$0.985^{\pm0.022}$	$0.949^{\pm0.024}$	$0.948^{\pm0.025}$	$0.874^{\pm0.289}$	$0.962^{\pm0.004}$	$0.951^{\pm0.005}$

Metrics. For the Nucleotide Transformer downstream tasks different metrics are used depending on the type of task: MCC was used for histone markers and enhancer annotation, F1-score was used for promoter annotation and splice site acceptor/donor, and accuracy was used for the splice site. The downstream performance on the Genomic benchmark was evaluated using the Top-1 accuracy.

Results. On the extensive set of downstream tasks, DNA-xLSTM is the best model with fewer than 2M parameters outperforming other small models on 12 of 18 tasks. In a comparison with much larger models, DNA-xLSTM and is on par with the 500M parameter model Nucleotide Transformer (NT-v2) winning 8 of 18 tasks (see Table A1). On the Genomic benchmark, DNA-xLSTM is overall on par with Mamba-DNA and shows especially strong results with posthoc conjoining, winning 5 of 8 tasks compared to Mamba-DNA-PH. Results are reported in Table A2.

Table A2: Downstream adaption of DNA language models on the Genomics Benchmarks. The Top-1 **accuracy** (↑) for RC-equivariant PS and PH xLSTM and Mamba-based Caduceus models, both with 500k parameters, are shown. Error bars represent the range of scores across five random seeds. xLSTM achieves comparable overall performance to Mamba and demonstrates superior accuracy when both models employ post-hoc conjoining. Scores for all models except xLSTM were obtained from Schiff et al. (2024).

	Mamba-PH-500k	xLSTM-PH-500k	Mamba-PS-500k	xLSTM-PS-500k
Mouse Enhancers	$0.754^{\pm 0.074}$	$0.780^{\pm 0.018}$	$0.793^{\pm 0.058}$	$0.778^{\pm 0.007}$
Coding. vs. Intergenomic	$0.915^{\pm 0.003}$	$0.931^{\pm 0.001}$	$0.910^{\pm 0.003}$	$0.934^{\pm 0.002}$
Human vs. Worm	$0.973^{\pm 0.001}$	$0.965^{\pm 0.001}$	$0.968^{\pm 0.002}$	$0.956^{\pm 0.001}$
Human Enhancers Cohn	$0.747^{\pm 0.004}$	$0.742^{\pm 0.005}$	$0.745^{\pm 0.007}$	$0.734^{\pm 0.005}$
Human Enhancers Ensemble	$0.893^{\pm 0.008}$	$0.920^{\pm 0.001}$	$0.900^{\pm 0.006}$	$0.902^{\pm 0.004}$
Human Regulatory	$0.872^{\pm 0.011}$	$0.872^{\pm 0.002}$	$0.873^{\pm 0.007}$	$0.869^{\pm 0.005}$
Human OCR Ensembl	$0.828^{\pm 0.006}$	$0.826^{\pm 0.002}$	$0.818^{\pm 0.006}$	$0.800^{\pm 0.002}$
Human NonTATA Promoters	$0.946^{\pm 0.007}$	$0.951^{\pm 0.004}$	$0.945^{\pm 0.010}$	$0.949^{\pm 0.001}$

Table A3: Pre-training hyperparameters for DNA-Transformer++ models with 2M and 4M parameters. Comma-separated values represent hyperparameter sweeps, with the chosen values indicated in bold.

Hyperparameters	DNA-Transformer++-2M	DNA-Transformer++-4M
Embedding Dimension	256	256
Number of Blocks	4	6
Number of Heads	8	8
Up-Projection Ratio	1.25:1	2:1
Norm Bias and Linear Bias	false	false
Context Length	1,024	32,768
Position Embeddings	RoPE	RoPE
Learning Rate	6e-3, 8e-3, **1e-2**	6e-3, 8e-3, **1e-2**

C.3 Architecture and Hyperparameters

The hyperparameters and composition of the DNA-xLSTM and DNA-Transformer++ models for pre-training with context size 1k and 32k are reported in Tables A4 and A3. The hyperparameters were selected on a separate validation set using manual hyperparameter selection due to limited computational resources.

C.4 Reverse-Complement Invariance

We develop an xLSTM version that is invariant to the RC of an input sequence which is relevant for DNA applications following Schiff et al. (2024). In double-helix DNA structures, both strands are semantically equivalent, as one strand is the RC of the other. Given a strand, □, its RC, $\overline{\square}$, is oriented in the opposite direction with a base conversion from `A` to `T` and `C` to `G` (Schiff et al., 2024). Shrikumar et al. (2017) show that a data-driven approach to learning the equivalence between reverse-complement sequences can fail, which is why Schiff et al. (2024) propose to enforce RC-equivariance by design, making use of two different inductive biases in the model architecture: PH (Zhou et al., 2022) and PS. For PH models, sequence-to-sequence models — in our case realized by the xLSTM — learn to handle both DNA sequences and their RC during pre-training by applying RC augmentation to the inputs. RC augmentation refers to the process of randomly replacing input sequences by their RCs. For downstream tasks PH models are applied once to the original sequence and once to the RC and eventually outputs are summed:

$$\boldsymbol{Y} = \mathrm{xLSTM}(\boldsymbol{X}) + \mathrm{xLSTM}(\overline{\boldsymbol{X}}). \tag{24}$$

For PS models — we assume models are realized by xLSTM architectures and therefore a block refers to a single mLSTM or sLSTM block — both the DNA sequence and its RC are provided simultaneously to each block in the architecture (for both pre-training and downstream task fine-tuning). Precisely, a joint representation, originating from combining a sequence representation and its RC representation, is split into $\boldsymbol{X} \in \mathbb{R}^{D \times t}$ and $\overline{\boldsymbol{X}} \in \mathbb{R}^{D \times t}$ and fed into the mLSTM or sLSTM

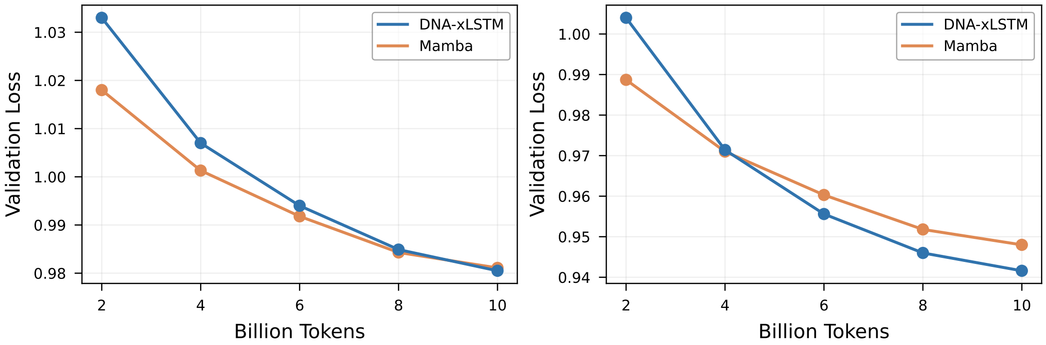

Figure A3: Pre-training of RC-equivariant xLSTM-DNA-PS and Caduceus-PS models with 500k and 2M parameters trained on the human reference genome. Models were trained on 1k context windows using the MLM objective. **Left: MLM losses** (↓) for models with 500k parameters. **Right: MLM losses** (↓) for models in the 2M parameter range. DNA-xLSTM-PS outperforms Caduceus-PS in both settings, with the performance gap widening at larger scales.

block:

$$\left[\boldsymbol{H}, \overline{\boldsymbol{H}}\right] = \left[\text{block}(\boldsymbol{X}), \text{RC}(\text{block}(\text{RC}(\overline{\boldsymbol{X}})))\right] . \tag{25}$$

Notably, for each block the reverse-complement input is built by the RC-function which flips both dimensions of $\overline{\boldsymbol{X}}$ and $[\cdot,\cdot]$ indicates concatenation along the first dimension. Eventually, logits for the input sequence and its reverse complement are combined. For more details, we refer to Schiff et al. (2024).

C.5 Implementation Details

For both CLM and MLM pre-training we perform 10,000 update steps holding the number of tokens per step constant at 2^{20}. CLM models are trained using autoregressive next-token prediction. For MLM pre-training, we follow the methodology presented by Devlin et al. (2019), where 15% of the input tokens are masked and the model is tasked to predict the corrupted tokens. Concretely, 80% of the masked tokens are replaced by a special [MASK] token, 10% are replaced by random tokens sampled from the vocabulary and 10% remain unchanged. For MLM settings, we use weight-tied bidirectionality as a default (see Section B.5). For long-context bidirectional modeling, we use unidirectional xLSTM cells and alternate the modeling direction at each block.

To fine-tune pre-trained models on downstream tasks, we follow the framework from Schiff et al. (2024). Pre-trained models are augmented with a task-specific classification head, which is trained on average-pooled activations from a model's final block. During fine-tuning, all model parameters are unfrozen. For the Genomic benchmark, we perform five randomly seeded train-validation splits, fine-tune models for 10 epochs, and use early-stopping on validation performance. Final test results are reported as the mean performance ± max/min over the 5 seeds on a held-out test set. For the Nucleotide Transformer tasks, we use 20 epochs and 10 seeds. For both the Genomic benchmarks and the Nucleotide Transformer tasks, we performed a hyperparameter search for both PH and PS models over batch sizes $\{64, 128, 256, 512\}$, and learning rates {4e-4, 6e-4, 8e-4, 1e-3, 2e-3}. The best results for each Nucleotide Transformer task can be found in Table A5.

Table A4: Pre-training hyperparameters of DNA-xLSTM Models from 500k to 4M parameters. Comma-separated values represent hyperparameter sweeps, with the chosen values indicated in bold.

Hyperparameters	DNA-xLSTM-500k	DNA-xLSTM-2M	DNA-xLSTM-4M
Embedding Dimension	128	256	256
Number of Blocks	5	6	9
Conv 1D Kernel Size	4	4	4
Number of Heads	4	4	4
Up-Projection Ratio	1.25:1	1:1	2:1
Bidirectionality	alternating, **blockwise**	alternating, **blockwise**	**alternating**, native, blockwise
Norm Bias and Linear Bias	true, **false**	true, **false**	true
QKV Projection Blocksize	-	-	4
m/sLSTM ratio	**[0:1]**, [1:0]	**[0:1]**, [1:0]	[0:1], **[1:0]**
Context Length	1,024	1,024	32,768
Position Embeddings	None	None	RoPE
Optimizer	AdamW $\beta = (0.9, 0.95)$	AdamW $\beta = (0.9, 0.95)$	AdamW $\beta = (0.9, 0.95)$
Learning Rate	6e-3, **8e-3**, 1e-2	6e-3, **8e-3**, 1e-2	6e-3, **8e-3**, 1e-2
Learning Rate Schedule	Cosine Decay	Cosine Decay	Cosine Decay
Learning Rate Warmup Steps	1,000	1,000	1,000
Weight Decay	0.1	0.1	0.1
Dropout	0	0	0
Batch Size	1,024	1,024	32
Update Steps	10,000	10,000	10,000

Table A5: Hyperparameter selection for DNA-xLSTM-PS and DNA-xLSTM-Ph on Nucleotide Transformer tasks. Fine-tuning hyperparameters were chosen based on best scores averaged over ten train-validation splits.

	DNA-xLSTM-Ph		DNA-xLSTM-PS	
	Learning Rate	Batch Size	Learning Rate	Batch Size
Histone Markers				
H3	8e-4	128	4e-4	64
H3K14AC	6e-4	128	4e-4	64
H3K36ME3	6e-4	64	4e-4	64
H3K4ME1	8e-4	128	1e-3	128
H3K4ME2	6e-4	64	2e-3	512
H3K4ME3	8e-4	128	1e-3	512
H3K79ME3	1e-3	128	4e-4	64
H3K9AC	4e-4	64	1e-3	128
H4	8e-4	64	6e-4	64
H4AC	4e-4	64	1e-3	128
Regulatory Annotation				
Enhancers	2e-3	512	2e-3	512
Enhancers Types	2e-3	512	2e-3	512
Promoter All	4e-4	64	1e-3	128
Promoter No TATA	1e-3	128	1e-3	128
Promoter TATA	3e-3	128	1e-3	128
Splice Site Annotation				
Splice Sites All	8e-4	64	2e-3	128
Splice Sites Acceptor	2e-3	128	2e-3	128
Splice Sites Donors	3e-3	128	2e-3	128

D Prot-xLSTM: Details and Additional Results

In this section, we provide further details regarding the architecture, training setup, and evaluation metrics for the Prot-xLSTM models. Additionally, we present supplementary results that complement the main findings discussed in Section 4.2.

To evaluate the performance of our Prot-xLSTM models, we adopted the experimental protocols outlined in Sgarbossa et al. (2024). We conducted three key experiments to assess the models' capabilities: **a) protein language modeling** (Section D.1), **b) homology-conditioned protein design** (Section D.2), and **c) protein variant fitness prediction** (Section D.3).

D.1 Homology-Aware Training

For protein sequences, we followed the experimental protocols from Sgarbossa et al. (2024).

Data. The protein language model training data was derived from the filtered OpenProteinSet (Ahdritz et al., 2023), comprising 270k UniClust30 MSA clusters that included a total of 508M sequences and 110B residues. We used the ProtMamba pipeline to construct the training data, which is illustrated in Fig. 1 of Sgarbossa et al. (2024), and involved two key steps: (i) creating homology-aware but alignment-free training inputs by concatenating unaligned homologous sequences, and (ii) masking patches of tokens in each sequence and concatenating the unmasked patches at the end of each sequence to train the model autoregressively with the FIM strategy. We also use the train, validation (192 clusters), and test (500 clusters) split provided by ProtMamba.

Methods and hyperparameters. We trained two versions of the model: Prot-xLSTM-26M and Prot-xLSTM-102M. The larger model was designed to match the architecture and size of the original ProtMamba model (ProtMamba-107M), and we optimized the xLSTM architecture on the smaller model. For comparison, we also trained a smaller ProtMamba model (ProtMamba-28M with a reduced embedding dimension of 512) and implemented an LLama-based model (Prot-Transformer++-26M) (Touvron et al., 2023). The composition of the Prot-xLSTM and Prot-Transformer++ models are reported in Tables A6 and A7, respectively.

Table A6: Hyperparameter space considered for the Prot-xLSTM at different sizes. The selected values are marked in bold.

Hyperparameter	Prot-xLSTM-26M	Prot-xLSTM-102M
Embedding dimension	512	1024
Context length[a]	2^{11},2^{17}	$2^{11\text{-}17}$
Number of blocks	16	16
m/sLSTM ratio	[0:1], **[1:0]**, [1:7][b]	[1:0]
Conv 1D kernel size	4	4
QKV projection blocksize	4	4
Number of heads	4	4
Up projection dimension	1024	2048
Norm bias and linear bias	False	False
Position embeddings	AbPE, AbPE$_{2D}$, **RoPE**, RoPE$_{2D}$	RoPE

[a] Context length was increased during training.
[b] sLSTM blocks at position 1 and 15.

Training details. We trained our models using the ProtMamba pipeline with CLM with the FIM strategy. The pipeline efficiently handles long, concatenated sequences by extending the context length up to $T = 2^{17}$, supported by a context-length scheduling strategy. For the Prot-xLSTM-102M model, we adhered to the ProtMamba protocol, gradually increasing the context length from 2^{11} to 2^{17}, doubling T at each stage when the loss plateaued. In contrast, for the smaller models (Prot-xLSTM-26M and ProtMamba-28M), as recommended in previous work (Devlin et al., 2019; Press et al., 2021), we initially trained with $T = 2^{11}$ for 20B tokens, then switched to $T = 2^{17}$ for an additional 10B tokens. Due to the quadratic scaling of Transformer architectures, Prot-Transformer++-26M was only trained with $T = 2^{11}$, as it could not handle the computational demands of $T = 2^{17}$.

Table A7: Hyperparameters of Prot-Transformer++ model

Hyperparameter	Prot-Transformer++-26M
Embedding Dimension	512
Context Length	2^{11}
Number of Blocks	6
Up Projection Dimension	2176
Norm Bias and Linear Bias	False
Position Embeddings	RoPE

Table A8: Hyperparameters for training protein sequence models.

Effective batch size[a,b]	64-1
Optimizer	AdamW $\beta = (0.9, 0.95)$
Learning rate[b,c]	6e-4
Learning rate scheduler	constant
Learning rate warmup steps	500
Weight decay	0.1
Dropout	0

[a] Decreased with context size to maintain a fixed total number of tokens per batch. For the larger model, the rule was relaxed for $T = 2^{16}$ and 2^{17} to enable multi-GPU training, with the batch size set to the number of GPUs.
[b] Prot-Transformer++ was trained on 6 GPUs with an effective batch size of 96 and a learning rate of 9e-4.
[c] Due to unstable training of the larger model at $T = 2^{17}$ and 2^{18} the learning rate was reduced to 1e-4.

Given the substantial computational resources required, we did not fine-tune the training parameters. Instead, we used the default settings established by ProtMamba, which are reported in Table A8.

Metrics. During training, we evaluated the next-token prediction capabilities of the models using negative log-likelihood and token perplexity. The perplexity was calculated for different segments of the concatenated-FIM sequence: the first sequence (first_seq), the second sequence (second_seq), and the last protein sequence (last_seq). We also evaluated performance specifically on the FIM tokens (fim) and the entire concatenated sequence. Once the models were trained we evaluated their performance on the independent test set with $T = 2^{17}$.

D.2 ICL: Homology-Conditioned Protein Generation

Experimental setup. To evaluate the capacity of Prot-xLSTM to autoregressively generate novel protein sequences given a context of known homologs, we follow the protocol outlined in Section 3.4 of Sgarbossa et al. (2024). For a subset of 19 homology clusters from the test set, we generate sequences with contexts consisting of 10, 100, 500, 1000 and N (total number of sequences in the cluster) sequences. For each context length, we generate 100 sequences each with the following parameter combinations of generation temperature (τ), top-k, which restricts the output selection to the k most probable tokens, and top-p, which limits the output to tokens reaching a cumulative probability p: $(\tau, \text{top-}k, \text{top-}p) \in \{(0.8, 10, 0.9), (0.9, 10, 0.95), (1, 10, 0.95), (1, 10, 1), (1, 15, 1)\}$ (Ferruz et al., 2022). This results in a total of 2,500 sequences per cluster.

Methods compared and hyperparameter selection. We compare both Prot-xLSTM models to ProtMamba models with a similar number of parameters. Note that the large Prot-xLSTM model was evaluated after training for ~45B tokens with context length up to 2^{16}.

Metrics. We evaluate the novelty of generated sequences by calculating the Hamming distance to the closest natural sequence in the cluster using pairwise Smith-Waterman alignment. Additionally, we measure sequence similarity to homologs with the HMMER score from a Hidden Markov Model (HMM) trained on the cluster's MSA. The generated sequences are also folded using ESMFold (Lin et al., 2023) and assessed by pTM and average pLDDT confidence scores. To compare these scores with natural sequences, we compute Kolmogorov-Smirnov test statistics between the scores of 100 natural sequences and the 100 generated sequences with the lowest perplexity.

Results. Figure A4 displays the distribution of scores for 100 randomly sampled natural sequences from each cluster as well as the 100 sequences with the lowest perplexity generated by Prot-xLSTM and ProtMamba models for 10 randomly selected clusters. Table A9 shows the average across all 19 evaluated test clusters. Sequences generated by Prot-xLSTM-102M were on average longer, more similar to other proteins in the cluster (measured by Hamming distance), and got a higher HMMER score and higher folding confidence scores compared to ProtMamba-generated sequences. Notably, these observations mostly also hold when compared to natural sequences.

Table A9: Score comparison of natural and generated proteins. Average scores (sequence length, Hamming distance to the closest natural neighbor, HMMER score, pLDDT and pTM) across 19 test clusters for sequences generated with Prot-xLSTM and ProtMamba models. Error bars indicate 95% confidence intervals across clusters.

	Natural Seqences	Prot-xLSTM	ProtMamba	Prot-xLSTM[a]	ProtMamba
		-26M	-28M	-102M	-107M
Sequence length	$211^{\pm 28}$	$290^{\pm 36}$	$326^{\pm 43}$	$286^{\pm 38}$	$276^{\pm 40}$
Min. Hamming ↓	$0.51^{\pm 0.04}$	$0.55^{\pm 0.05}$	$0.64^{\pm 0.04}$	$0.44^{\pm 0.07}$	$0.56^{\pm 0.03}$
HMMER ↑	$96^{\pm 25}$	$182^{\pm 56}$	$122^{\pm 50}$	$165^{\pm 45}$	$163^{\pm 45}$
pLDDT ↑	$0.81^{\pm 0.03}$	$0.79^{\pm 0.04}$	$0.67^{\pm 0.07}$	$0.80^{\pm 0.03}$	$0.80^{\pm 0.03}$
pTM ↑	$0.77^{\pm 0.06}$	$0.74^{\pm 0.06}$	$0.54^{\pm 0.10}$	$0.75^{\pm 0.06}$	$0.74^{\pm 0.06}$

[a] Trained for ∼45B tokens with context length up to 2^{16}.

Table A10 demonstrates that Hamming distance, HMMER score, pTM and pLDDT correlate well with sequence perplexity for both, Prot-xLSTM and ProtMamba, with an average Pearson correlation coefficient of 0.57 for both large models.

Table A10: Score distribution comparison of natural and generated proteins. Average Pearson correlation between model perplexity and sequence scores (sequence length, Hamming distance to the closest natural neighbor, HMMER score, pLDDT and pTM) for sequences generated with Prot-xLSTM and ProtMamba models. Error bars indicate 95% confidence intervals across 19 test clusters.

	Prot-xLSTM-26M	ProtMamba-28M	Prot-xLSTM-102M [a]	ProtMamba-107M
Pearson $r_{\text{ppl/score}}$ (↑)				
Min. Hamming	$0.53^{\pm 0.10}$	$0.41^{\pm 0.10}$	$0.59^{\pm 0.08}$	$0.57^{\pm 0.11}$
HMMER Score	$0.59^{\pm 0.06}$	$0.54^{\pm 0.07}$	$0.54^{\pm 0.07}$	$0.57^{\pm 0.09}$
pLDDT	$0.66^{\pm 0.05}$	$0.53^{\pm 0.07}$	$0.60^{\pm 0.08}$	$0.62^{\pm 0.08}$
pTM	$0.59^{\pm 0.06}$	$0.44^{\pm 0.08}$	$0.55^{\pm 0.07}$	$0.57^{\pm 0.07}$

[a] Trained for ∼45B tokens with context length up to 2^{16}.

D.3 Protein Variant Fitness Prediction

Experimental setup. We evaluate Prot-xLSTM's ability to predict mutational effects by leveraging its inpainting capabilities from the FIM training objective. This assessment follows the protocol described in Section 3.2 of Sgarbossa et al. (2024) for the ProteinGym DMS substitution benchmark (Notin et al., 2023), which consists of 217 datasets of single and multiple substitutions in protein sequences, allowing comparison with state-of-the-art methods for protein variant fitness prediction. Briefly, for each wild-type sequence, three sets of 200 homologs were obtained by subsampling MSAs following the ColabFold protocol (Mirdita et al., 2022) to be used as context. The context sequences are ordered from the least similar to the most similar one. The wild-type sequence is then concatenated with the context, the mutated residue is masked, and this residue is predicted using the FIM method. Fitness is evaluated as the difference in likelihood between the concatenated sequence with the wild-type and the mutated amino acid and averaged over the triplicate. For multiple mutations, fitness is approximated as the sum of the likelihoods of single mutations.

Methods compared and hyperparameter selection. We compare both Prot-xLSTM models to the **ProtMamba** models, as well as to **PoET** (Truong Jr & Bepler, 2023), a transformer that introduced the concept of non-aligned homologous sequences for protein language modeling, and **SaProt** (Su et al., 2024b), a transformer with a structure-aware vocabulary that currently leads the ProteinGym leaderboard.

Metrics. ProteinGym's main metric is the average Spearman correlation between the fitness predictions and the experimental DMS results.

Results. Table A11 summarizes the results on the ProteinGym benchmark.

Table A11: ProteinGym zero-shot DMS substitution benchmark. The average **Spearman correlation** (↑) between predicted fitness scores and experimental measures over 217 DMS assays is shown. While even small Prot-xLSTM models already yield high scores, the large SaProt model, which uses additional structure tokens, performs best.

Prot-xLSTM -26M	ProtMamba -28M	SaProt -35M	Prot-xLSTM -102M[a]	ProtMamba -107M	PoET -201M	SaProt -650M
0.411	0.360	0.406[b]	0.415	0.416	0.484[c]	0.457[a,d]

[a] Trained for 60B tokens.
[b] Values from `proteingym.org`.
[c] Value from Truong Jr & Bepler (2023), not verified by ProteinGym leaderboard.
[d] Leader of ProteinGym leaderboard on 27/09/2024.

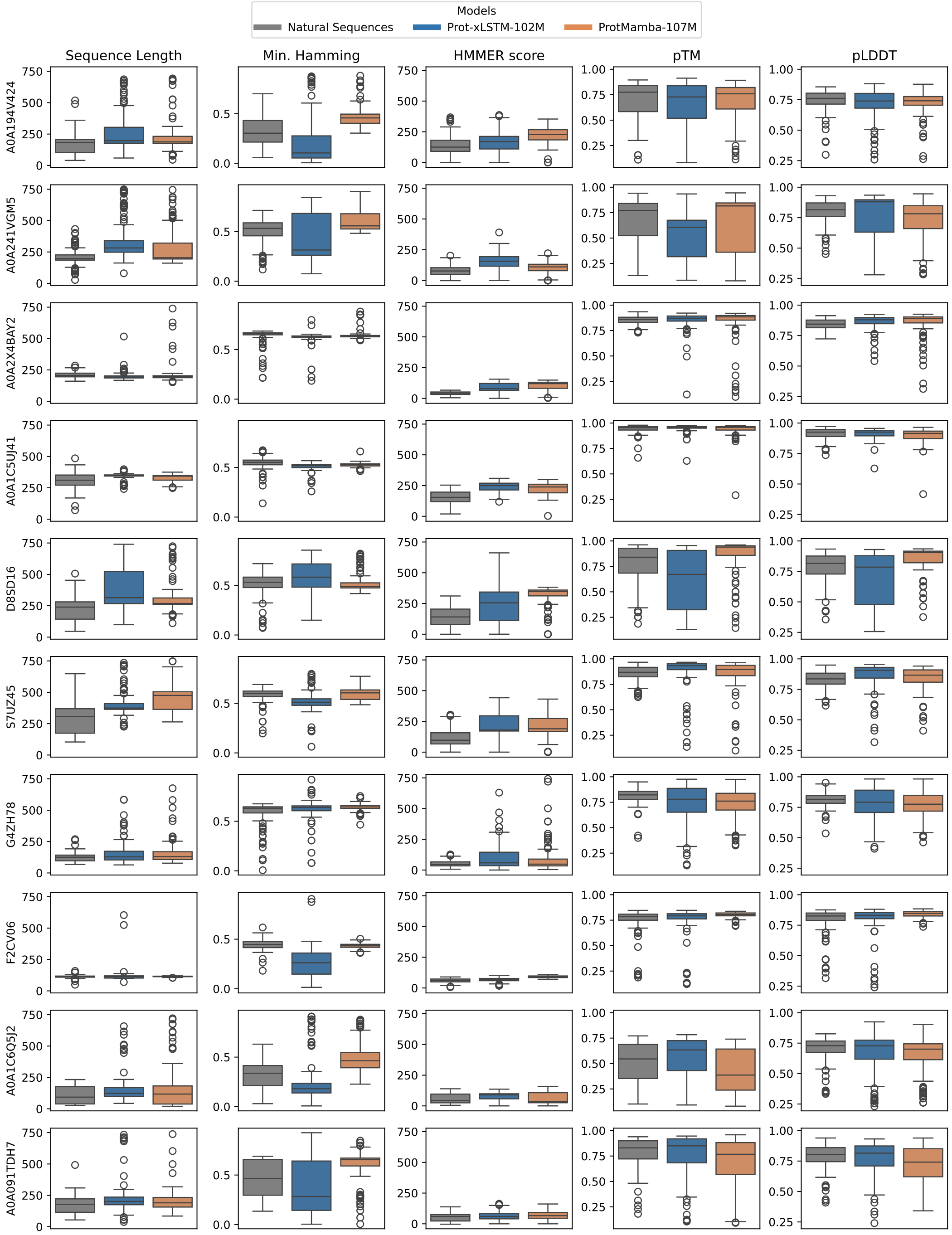

Figure A4: Scores of natural and generated proteins. Boxplots of score distributions (sequence length, Hamming distance to the closest natural neighbor, HMMER score, pLDDT, and pTM) for 10 randomly selected clusters evaluated for 100 randomly chosen natural sequences and 100 generated sequences with lowest perplexity values for large Prot-xLSTM and ProtMamba models.

E Chem-xLSTM: Details and Additional Results

For chemical sequences, we perform two sets of experiments: **a) unconditional molecule generation** where we follow the experimental protocol of Özçelik et al. (2024). Additionally, we propose a new and more challenging task: **b) conditional generation with ICL**, in which we generate new compounds conditional based on provided in-context compounds.

E.1 Unconditional Molecule Generation

Unconditional molecule generation is the task of generating valid small molecules without imposing constraints on their characteristics or properties. Generative models aim to learn a general distribution by processing many examples of desirable results. To this end, models are trained on large training sets of arbitrary small molecules without particular conditions or constraints (Segler et al., 2018; Gómez-Bombarelli et al., 2018; Özçelik et al., 2024). Following this approach, we compared the ability of xLSTM and several other models to generate valid and diverse molecules.

Experimental setup. For comparability, we aligned our experiments with the setting and dataset of Özçelik et al. (2024). This means that all models are trained to generate molecules as SMILES strings (Weininger, 1988) using a CLM paradigm. The dataset used in (Özçelik et al., 2024) is derived from ChEMBL with a random split in 1.9M training, 100k validation, and 23k test molecules, which have been encoded as SMILES. Before training, all SMILES strings were tokenized using a regular expression, containing all elements. This results in atoms being represented as one token as well as additional SMILES symbols.

Methods and hyperparameters. We compared xLSTM with several other model classes. The first baseline is the default LSTM (Hochreiter & Schmidhuber, 1997) in PyTorch, which includes a forget gate (Gers et al., 1999). This can be considered the direct predecessor of the xLSTM architecture. We also included a variant GPT-2 (Radford et al., 2019) model based on the Transformer architecture (Vaswani et al., 2017) with causal masking. Finally, we included two SSMs in our comparison. On one side, we considered an S4 model with the implementation from Gu et al. (2022), following (Özçelik et al., 2024). On the other side, we incorporated a Mamba model, using the official repository provided with (Gu & Dao, 2023). For our Chem-xLSTM, we used an xLSTM using only mLSTM blocks (Beck et al., 2024). The 15M-parameter model consists of 9 layers with a hidden dimension of 512 and 8 heads. We trained the model for up to 100 epochs with a batch size of 1,024, a context length of 100, a dropout rate of 0.25, and a learning rate of 0.005. All models were trained using the Adam optimizer (Kingma & Ba, 2015) using $\beta = (0.9, 0.999)$, $\epsilon = 1e^{-8}$, and a learning-rate schedule with warm-up and cosine decay. We selected the best model based on the minimum validation loss observed at the end of each epoch. The hyperparameters were manually tuned to match the model parameter count for a fair comparison.

Metrics. We evaluated each model with respect to the perplexity on the next token, and the FCD (Preuer et al., 2018). The FCD has been introduced as an alternative to the FID, which is used to evaluate image generation, for molecule generation. Additionally, we evaluate auxiliary metrics that measure the syntactic correctness, novelty, diversity, or synthetic accessibility.

Results. Our proposed Chem-xLSTM model achieved the best results, with the lowest FCD (0.13) and a perplexity (1.68) that is competitive with that of GPT-based models. This indicates that Chem-XLSTM is able to generate realistic chemical structures that match the target distribution well.

All models in our comparison were able to produce valid, unique, and novel molecules. Even though these models have not been optimized for these properties. This is evidenced by the auxiliary metrics surpassing practical thresholds (see Table A13).

E.2 Conditional Molecule Generation with In-Context Learning

Conditional molecule generation with in-context learning (ICL) leverages contextual information to guide the design of novel molecules tailored for specific domains. By incorporating a sequence of molecules as the input, models can conditionally generate new compounds of the same distribution, without the need for fine-tuning.

Experimental setup. Similar to the unconditional setup, the input consists of SMILES strings. In the conditional setup, we additionally model sets of molecules from the same molecular domain as a

Table A12: Hyperparameter space considered for the Chem-xLSTM at different sizes. The selected values are marked in bold.

Hyperparameter	Chem-xLSTM-15Mn	Chem-xLSTM-15M-icl
Number of layers	**9**	**9**
Number of heads	**8**	**8**
Embedding dimension	**512**	**512**
Hidden dimension	**512**	**512**
Batch size	16, **32**, 64, 128	16, **32**
Proj. factor	**1.3**	**1.3**
Learning rate	1e-4, **2e-4**, 3e-4, 5e-4	16, 1e-4, **2e-4**, 3e-4, 5e-4
Optimizer	**Adam**, AdamW	**Adam**

sequence. Molecules from one molecular domain are serialized and concatenated, separated with the "." token. During training, the order of the molecules is permuted to improve generalization and robustness. We construct a novel dataset derived from a variety of molecular domains:

- We consider `natural-products` as domain and utilize the Coconut (Nainala et al., 2024) as source dataset.
- `Kinase inhibitors`, `withdrawn`, `malaria`, `tool compounds`, `pathogen`, `NIH mechanistic`, `lopac`, `natural product-based probes and drugs`, `zinc tool`, `axon medchem`, `adooq bioactive`, `novartis chemogenetic`, `drug matrix`, `PROTACs`, `covalentIn db`, `DrugBank compounds`, `reframe`, `cayman bioactive` all from the Probes & Drugs portal (Skuta et al., 2017),
- `product molecules` from the reaction dataset USPTO-50k (Lowe, 2012) split into 10 reaction classes.
- The domains `bio`, `diversity`, `green`, `yellow`, `orange`, and `red`, from ZINClick (Levré et al., 2018).
- Active molecules from the domains `BACE`, `BBBP`, `Clintox`, `HIV`, `SIDER`, `Tox21`, `Tox21-10k`, and `Toxcast` from MoleculeNet (Wu et al., 2018).
- Active molecules from 95 bioassays from FS-MOL (Stanley et al., 2021) considered each as separate domain.
- Active molecules from 109 bioassays from PubChem (Kim et al., 2023) considered each as separate domain.
- A subset of active molecules from the BELKA challenge (Quigley et al., 2024) is modeled as a domain.

For the domains that are defined by the active molecules from a particular bioassay, we selected assays with at least 300 active molecules and only use the active compounds. For the dataset each of the total 249 domains is limited to 100,000 compounds, where compounds are selected at random. The final dataset is split at 8:1:1 into train-, validation- and test-domains, sorted by their character length in descending order.

Methods and hyperparameters. We benchmark and orient our choices for the model classes as well as hyperparameters based on the unconditional molecule generation results, We consider a context length of 4,096 and adjust batch sizes as well as accumulation steps to accommodate GPU memory constraints. For the S4 model, we were only able to fit a context length of 2,048.

Metrics. To evaluate conditional molecule generation we evaluate NTP loss. This metric quantifies how well the model predicts the next token in a sequence, thus assessing whether a model is able to generate molecules from an unseen, and potentially very special, molecular domain given only a few molecules from that domain.

E.3 Architecture and Hyperparameter Selection

Considered and selected hyperparameters for Chem-xLSTM are given in A12.

Table A13: Diversity and correctness metrics for the 15M parameter models for small molecules (SMILES). The table reports the percentage of valid, unique, and novel molecules, the synthetic accessibility (SA), and the diversity metric by the percentage of unique Murcko scaffolds divided by the total number of generated molecules.

Model	valid %	unique %	novel %	SA ↓	diverse %
SMILES-LSTM (Segler et al., 2018)	$90.11^{\pm 10.7}$	$56.72^{\pm 3.4}$	$56.66^{\pm 3.6}$	$2.85^{\pm 0.0}$	$44.71^{\pm 1.1}$
SMILES-GPT (Adilov, 2021)	$99.05^{\pm 0.5}$	$62.09^{\pm 12.1}$	$61.81^{\pm 12.0}$	$2.90^{\pm 0.0}$	$48.82^{\pm 9.7}$
SMILES-S4 (Özçelik et al., 2024)	$97.48^{\pm 0.0}$	$61.47^{\pm 0.0}$	$61.34^{\pm 0.0}$	$2.86^{\pm 0.0}$	$48.49^{\pm 0.0}$
Chem-Mamba[a]	$91.41^{\pm 8.9}$	$57.75^{\pm 3.2}$	$57.63^{\pm 3.8}$	$2.84^{\pm 0.0}$	$45.65^{\pm 7.2}$
Chem-xLSTM (ours)	$97.08^{\pm 0.7}$	$61.09^{\pm 8.9}$	$60.84^{\pm 9.6}$	$2.83^{\pm 0.0}$	$45.97^{\pm 5.5}$

[a] adapted to SMILES in this work

E.4 Implementation Details

Unlike Özçelik et al. (2024), we do not backpropagate the loss for [PAD] tokens, nor do we interpret them for decoding. We observed that not ignoring [EOS] and [PAD] token leads to more diversity but is not the standard way of decoding in e.g. NLP. Padding tokens are not typically generated during decoding. They are primarily a pre-processing step to handle batches of data efficiently. In our implementation, we end decoding the SMILES string with the [EOS] token. Further, we do not use SMILES augmentation, which could further improve the performance of all architectures.

E.5 Additional Results

Practical thresholds are defined based on several key metrics. First, a high percentage of generated SMILES strings must correspond to chemically valid molecules, with a threshold typically set above 90% to ensure reliability. Additionally, a practical threshold for uniqueness might require that over 80% of the generated molecules are unique, ensuring diversity in the explored chemical space. For novelty, at least 50-70% of the generated molecules should be novel compared to known chemical databases, indicating the model's ability to explore new regions of chemical space. Finally, all models exhibit favorable synthetic accessibility (SA) scores, typically ranging between 2.5 and 5, ensuring that the generated molecules are feasible for synthesis. Further metrics and details are provided in the appendix.

F Limitations

While Bio-xLSTM shows strong performance across DNA, protein, and chemical sequence modeling, it has several limitations. The manual hyperparameter selection, which was due to limited computational resources, may prevent optimal model configurations. We will explore the hyperparameter spaces in the future, which might yield even better models. For DNA, the reliance on character-level tokenization might also restrict the performance and scaling to larger context sizes. The models DNA-xLSTM, Prot-xLSTM, and Chem-xLSTM are currently constrained by the training dataset and their generalizability across organisms and chemical domains needs further exploration. Across all three domains, the datasets used for training contain biases – whether it's population biases in the genomic data, sequence distribution biases in protein datasets, or chemical exploration biases in molecular datasets. These biases could influence the model's predictions and limit its generalizability in real-world applications. In line with many works, we consider the perplexity metric, for example, next token perplexity, or the related cross-entropy losses as a proxy for performance on downstream tasks. However, this metric might not capture the capacities of biological and chemical language models appropriately. Future work could address these limitations by expanding the training datasets and exploring more efficient architectures tailored to the specific challenges of each domain.

G Conclusions

In this work, we introduced the Bio-xLSTM architecture and demonstrated its effectiveness across three key domains: DNA, protein, and small molecule modeling. In DNA sequence modeling, Bio-xLSTM showed strong performance in both masked and causal language modeling tasks. For protein sequences, Bio-xLSTM clearly outperformed the state-of-the-art Mamba model in benchmarks and large-scale settings, establishing itself as the leading approach for generating homologous proteins. In the domain of small molecule generation, Bio-xLSTM achieved the best Fréchet ChemNet Distance (FCD) in unconditional molecule generation and demonstrated some capacity for in-context learning, showcasing its potential for future developments in conditional molecular design. Overall, Bio-xLSTM offers a versatile and competitive approach to sequence modeling across biological domains. In this work, we brought some clarity to both a) how to tailor xLSTM for biological and chemical sequences and b) how xLSTM-based models compare against other domain-specific LLMs, demonstrating their strong performance across DNA, protein, and chemical sequence tasks.