Caduceus: Bi-Directional Equivariant Long-Range DNA Sequence Modeling

Yair Schiff [1] Chia-Hsiang Kao [1] Aaron Gokaslan [1] Tri Dao [2] Albert Gu [3] Volodymyr Kuleshov [1]

Abstract

Large-scale sequence modeling has sparked rapid advances that now extend into biology and genomics. However, modeling genomic sequences introduces challenges such as the need to model long-range token interactions, the effects of upstream and downstream regions of the genome, and the reverse complementarity (RC) of DNA. Here, we propose an architecture motivated by these challenges that builds off the long-range Mamba block, and extends it to a BiMamba component that supports bi-directionality, and to a MambaDNA block that additionally supports RC equivariance. We use MambaDNA as the basis of Caduceus, the first family of RC equivariant bi-directional long-range DNA language models, and we introduce pre-training and fine-tuning strategies that yield Caduceus DNA foundation models. Caduceus outperforms previous long-range models on downstream benchmarks; on a challenging long-range variant effect prediction task, Caduceus exceeds the performance of 10x larger models that do not leverage bi-directionality or equivariance. Code to reproduce our experiments is available [redacted].

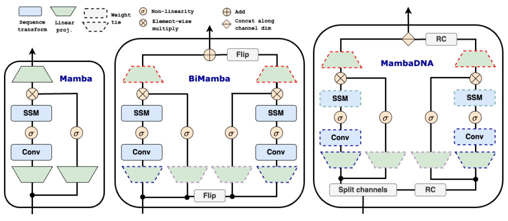


Figure 1. Mamba modules for genomic sequences. *(Left)* **Mamba**: The original left-to-right causal Mamba module proposed in Gu & Dao (2023). *(Middle)* **BiMamba**: A parameter efficient bi-directional extension of the Mamba module. In-projection and out-projection parameters are shared for processing the sequence and its reverse. After processing the reversed sequence, it is flipped again and added to the forward output. *(Right)* **Reverse complementary equivariant Mamba (MambaDNA)**: A module with RC equivariance inductive bias. The input is first split into two along the channel dimension. One split has the reverse complement (RC) operation applied to it. All the parameters of a Mamba module are shared for processing the forward and RC sequence. The reverse sequence has the RC applied once more before being concatenated back with the forward output along the channel dimension.

1. Introduction

Large-scale sequence models have sparked rapid progress in machine learning, bringing about advances that extend beyond natural language processing (NLP) (Achiam et al., 2023; Team et al., 2023) into science, biology, and medicine. In proteomics, these models have enabled predicting protein structures from sequences (Jumper et al., 2021; Lin et al., 2023), deciphering the functions and interactions of amino acids (Rao et al., 2020; Rives et al., 2021), and crafting new molecules (Madani et al., 2023).

[1]Department of Computer Science, Cornell University, New York, NY USA [2]Department of Computer Science, Princeton University, Princeton, NJ USA [3]School of Computer Science Carnegie Mellon University, Pittsburgh, PA USA. Correspondence to: Yair Schiff <yzs2@cornell.edu.>.

Proceedings of the 1st Workshop on Long-Context Foundation Models, Vienna, Austria. 2024. Copyright 2024 by the author(s).

(Zhou & Troyanskaya, 2015; Avsec et al., 2021). Unlike proteins, genomes contain non-coding sequences, which often play an important role in regulating cellular mechanisms, and can thus potentially provide greater insights into cell biology. Understanding non-coding sequences has been a key focus of recent work, including efforts in applying large language models (LMs) to genomes (Ji et al., 2021; Benegas et al., 2023b; Dalla-Torre et al., 2023; Nguyen et al., 2023).

In this paper, we propose architectural components motivated by the above challenges. Our modules build off the long-range Mamba block (Gu & Dao, 2023) and thus naturally handle long sequences of over hundreds of thousands of nucleotides without the quadratic computation cost of attention-based architectures (Vaswani et al., 2017). We extend Mamba to BiMamba, a component that supports bi-directionality, and to MambaDNA, which further adds reverse complement (RC) equivariance. The MambaDNA block can be used as a drop-in replacement in architectures for genome analysis in both supervised and self-supervised contexts.

We then use MambaDNA as the basis of Caduceus[1], a family of bidirectional long-range DNA sequence models that is

[1]Caduceus (⚚) is the staff carried by Hermes in Greek mythology that is adorned by two intertwined serpents. We choose this name to evoke imagery of the double helix structure of DNA and to symbolize bi-directionality using a Mamba sequence operator.

the first to support RC equivariant language modeling. We further introduce pre-training and fine-tuning strategies that yield Caduceus foundation models for a wide range of predictive tasks in genomics. The Caduceus models consistently outperform previous SSM-based models of a similar size (Nguyen et al., 2023) in terms of downstream performance. On many tasks, especially ones that require long-range modeling, Caduceus also outperforms 10x larger Transformer-based models.

We use Caduceus to perform variant effect prediction (VEP), a task that seeks to determine whether a genetic mutation influences a phenotype—gene expression in our case. This task is a natural fit for Caduceus because its pre-training implicitly learns to recognize the effects of evolutionary pressure (e.g., conservation, co-evolution), which is a key source of signal for VEP (e.g., a mutation in a region where mutations are rare likely has an effect and a low probability under the model). On a task derived from a standard dataset of mutations with long-range effects on gene expression (Avsec et al., 2021), Caduceus outperforms existing attention and SSM-based models that do not leverage both bi-directionality and equivariance.

2. Bi-Directional & RC-Equivariant Mamba

2.1. BiMamba

The first extension that we apply to the standard Mamba module is to convert it from causal (left-to-right) to bi-directional. We achieve this by applying the Mamba module twice: once to the original sequence and once to a copy that is reversed along the length dimension. To combine information, the output of the reversed sequence is flipped along the length dimension and added to the forward one.

A naive implementation of this method would double the number of parameters of the module. To avoid this added memory footprint, we instead share projection weights between the 'forward' and 'reverse' Mamba. These projections account for a vast majority of the model's parameters compared to those in the convolution and the SSM sub-modules. We refer to this parameter efficient bi-directional block as **BiMamba**. This module is depicted in the middle schematic of Figure 1.

2.2. MambaDNA

To encode the RC equivariance inductive bias into our modules, we apply a Mamba (or BiMamba) block to a sequence and its RC, with parameters shared between the two applications (Shrikumar et al., 2017; Zhou et al., 2021). Given its relevance to genomics, we dub this block **MambaDNA**.

More concretely, denote a sequence of length T with D channels by $\mathbf{X}^{1:D}_{1:T}$. Our channel splitting operation is then defined as:

$$\mathrm{split}(\mathbf{X}^{1:D}_{1:T}) := \left[\mathbf{X}^{1:(D/2)}_{1:T}, \mathbf{X}^{(D/2):D}_{1:T}\right].$$

We also define the RC operation as follows:

$$\mathrm{RC}\left(\mathbf{X}^{1:D}_{1:T}\right) := \mathbf{X}^{D:1}_{T:1}$$

Finally, letting concat denote the last operation of this module that re-combines the sequences along the channel dimension, our RC equivariant Mamba module, which we denote as $\mathrm{M}_{\mathrm{RCe},\theta}$, can be expressed as follows:

$$\mathrm{M}_{\mathrm{RCe},\theta}\left(\mathbf{X}^{1:D}_{1:T}\right) :=$$
$$\mathrm{concat}\left(\left[\mathrm{M}_\theta\left(\mathbf{X}^{1:(D/2)}_{1:T}\right), \mathrm{RC}\left(\mathrm{M}_\theta\left(\mathbf{X}^{D:(D/2)}_{T:1}\right)\right)\right]\right),$$

where M_θ represents the sequence operator that is parameterized by either the standard Mamba or BiMamba. The MambaDNA module is depicted in the rightmost schematic of Figure 1, with M_θ shown as the standard Mamba.

We claim that MambaDNA satisfies the RC equivariance property that we desire for processing DNA sequences:

Theorem 2.1. *The $\mathrm{M}_{RCe,\theta}$ operator satisfies the property that*

$$\mathrm{RC} \circ \mathrm{M}_{RCe,\theta}\left(\mathbf{X}^{1:D}_{1:T}\right) = \mathrm{M}_{RCe,\theta} \circ \mathrm{RC}\left(\mathbf{X}^{1:D}_{1:T}\right).$$

Proof. See Appendix A.

Similar to BiMamba modules, MambaDNA blocks do not entail significant additional memory footprint, since the wrapped sequence operator that processes the forward and RC sequences is completely shared.

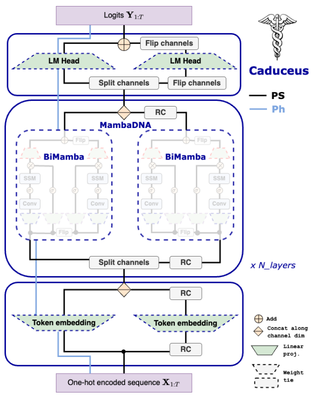


Figure 2. Caduceus Architecture. Bi-directional, RC equivariant Mamba modules are used in conjunction with equivariant word embeddings and language model head to form **Caduceus-PS**. Using only BiMamba blocks with RC data augmentation during pretraining and post-hoc conjoining for downstream task inference yields **Caduceus-Ph**. CADUCEUS IMAGE LICENSE: CREATIVE COMMONS CC0 1.0 UNIVERSAL PUBLIC DOMAIN DEDICATION.

3. Caduceus

Below we describe **Caduceus**, a novel bi-directional DNA LM architecture that enforces RC equivariance. We introduce two versions of this model, each of which maintains equivariance in a different manner: either (1) via parameter sharing (Shrikumar et al., 2017), Caduceus-PS, or (2) via a technique used during downstream tasks, known as *post-hoc conjoining* (Zhou et al., 2021), Caduceus-Ph.

3.1. Caduceus-PS

Architecture For **Caduceus-PS**, we leverage both of the architectural innovations introduced in Section 2. Namely, we wrap a BiMamba module within a MambaDNA block. Additionally, preceding the Mamba blocks of this architecture is an RC equivariant token embedding module. Denoting by Emb_θ the linear projection that takes one-hot vectors $\mathbf{X}^{1:4}_{1:T}$ and produces embeddings in $\mathbb{R}^{D/2}$, the RC equivariant version of this embedding is defined as:

$$\mathrm{Emb}_{\mathrm{RCe},\theta}\left(\mathbf{X}^{1:4}_{1:T}\right) := \mathrm{concat}\left(\left[\mathrm{Emb}_\theta\left(\mathbf{X}^{1:4}_{1:T}\right), \mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathrm{RC}\left(\mathbf{X}^{1:4}_{1:T}\right)\right)\right]\right)$$

Additionally, the logits of the Caduceus model are produced by passing the output of its final MambaDNA block through a RC equivariant language model head. To our knowledge, Caduceus-PS is the first model to incorporate RC equivariance into the LM pre-training paradigm. This can be formalized by first defining a channel flip operator $\text{flip\_chan}\left(\mathbf{X}^{1:D}_{1:T}\right) := \left(\mathbf{X}^{D:1}_{1:T}\right)$. Then, letting LM_θ be the linear projection from sequences with $D/2$ channels to vectors in $\mathbb{R}^4$, we define the equivariant version of the language modeling head as:

$$\mathrm{LM}_{\mathrm{RCe},\theta}\left(\mathbf{X}^{1:D}_{1:T}\right) := \mathrm{LM}_\theta\left(\mathbf{X}^{1:(D/2)}_{1:T}\right) + \text{flip\_chan}\circ\mathrm{LM}_\theta\left(\mathbf{X}^{D:(D/2)}_{1:T}\right).$$

Depicted in Figure 2 with the **black** path, Caduceus-PS enables RC equivariant language model pre-training: the predictions it produces for the RC of a given sequence are equivalent to reversing the predictions of the original sequence along the length dimension and complementing outputs: `A`-`T` and `C`-`G`. We formalize this claim in the following statement:

Theorem 3.1. *Composing* $\mathrm{LM}_{RCe,\theta}\circ\mathrm{M}^{(n)}_{RCe,\theta}\circ\mathrm{Emb}_{RCe,\theta}$*, where* $\mathrm{M}^{(n)}_{RCe,\theta}$ *denotes* n *compositions of different Mamba RC equivariant modules, yields a sequence operator that is RC equivariant.*

Proof. See Appendix B.

Pre-training Given the bi-directionality of this model, we train Caduceus-PS with the masked language modeling (MLM) objective, using the standard masking recipe proposed in BERT (Devlin et al., 2018). The RC equivariant language modeling of Caduceus-PS means that we do not need RC data augmentation at pre-training, since predictions are inherently symmetric with respect to this operation.

Downstream Usage For downstream tasks, since either strand of an assayed sequence will carry the same label, we wish to enforce RC *invariance*. The token embedding parameter sharing in Caduceus-PS means that its intermediate and final hidden states are twice the (channel) dimensionality of a standard Mamba-based language model with an equivalently sized token embedding matrix. To enforce RC invariance at downstream training and inference, final hidden states are split and the two splits are averaged.

3.2. Caduceus-Ph

Architecture The **Caduceus-Ph** model is depicted with the **blue** path in Figure 2. The core of this model is a stack of BiMamba blocks.

Pre-training As with Caduceus-PS, this model is pre-trained using the same MLM objective. However, as the model is not an RC equivariant LM, we instead rely on data augmentation during pre-training.

Downstream Usage In order to make the downstream task representations RC invariant, we leverage a technique called *post-hoc* conjoining (Zhou et al., 2021). Namely, for downstream task *training* the backbone model is unchanged, but we employ RC data augmentation. However, for downstream task *inference*, we apply the model twice, once on the original sequence and once on a corresponding RC sequence, and average the two, effectively performing a version of 'RC ensembling' (Mallet & Vert, 2021).

4. Experiments

4.1. Pre-training

Data We limit the focus of this work to human-genome related tasks. To that end, we perform all pre-training tasks on the human reference genome (Consortium et al., 2009). We use character- / base pair-level tokenization. While other DNA FMs have explored k-mer tokenization, this scheme suffers from the drawback that minor changes to an input sequence can lead to drastically different tokenization outputs (Zhou et al., 2023), which complicates training. Character-level tokenization avoids this issue. For any non-RC equivariant model that we train, including re-training HyenaDNA (Nguyen et al., 2023) models, we employ RC data augmentation during pre-training. For more information on the pre-training dataset and recipes see Appendix C.

Effect of Parameter Sharing on MLM Pre-training Projection parameter sharing in BiMamba enables deeper bi-directional models for similar parameter counts. We compare MLM pre-training loss of BiMamba models to naive bi-directional Mamba models that do not use weight tying and are therefore reduced to half the depth. We find that our parameter efficient implementation of bi-directionality leads to better pre-training loss, as seen in Figure 4b.

Effect of RC Equivariance on MLM Pre-training We also examine the effect of using our proposed RC equivariant LM on pre-training. In Figure 4c, we find that RC equivariant LM leads to better MLM pre-training loss. This is significant because, as described above, performance on the MLM task has grounding in the biology of downstream tasks, such as variant effect prediction.

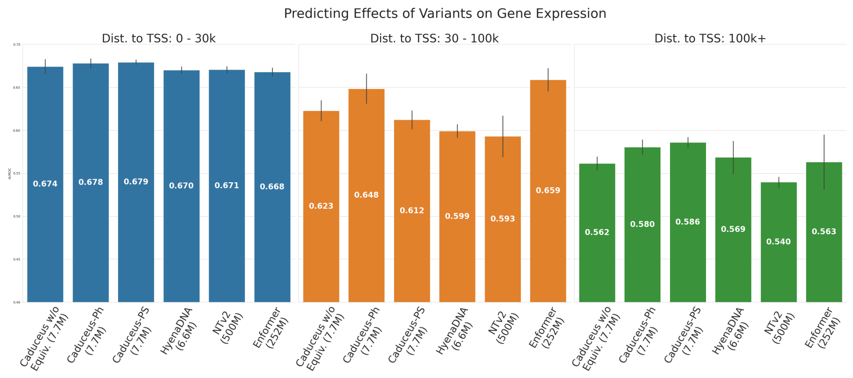


Figure 3. Predicting variant effects on gene expression across varying distances to the nearest Transcription Start Site (TSS). Models compared include Enformer, NT-v2, HyenaDNA, Caduceus w/o RC Equiv, Caduceus-Ph, and Caduceus-PS, with model sizes indicated in parentheses. SSM-based models utilize a 131k sequence length. We show performance at short (0 - 30kb), medium (30 - 100kb), and long-range (100kb+) distances to TSS. Notably, Caduceus-PS consistently demonstrates enhanced predictive accuracy for long-range effects. Error bars represent standard deviation across five SVM classifiers, each fit on different dataset subsets.

4.2. Downstream Tasks

4.2.1. Genomics Benchmark

We begin the evaluation with the Genomics Benchmarks (Grešová et al., 2023), a recently proposed benchmark with eight regulatory element classification tasks. Non-Mamba baselines consist of HyenaDNA and a supervised trained CNN model described in Grešová et al. (2023) . For HyenaDNA and all our Mamba-based models, we take the final hidden state embedding and perform mean pooling on the sequences, which vary from 200 to approximately 2,000 bps in length. We perform 5-fold cross-validation (CV) using different random seeds, with early stopping on validation accuracy and report mean and ± on max/min of the 5 seeds.

As shown in Table 1, Caduceus models attain best performance across all annotations. Of note, Caduceus-Ph is the best performing model overall for this suite of tasks. Other works that examine post-hoc conjoining, similarly find this to be a strong modeling decision (Zhou et al., 2021; Mallet & Vert, 2021).

4.2.2. Nucleotide Transformer Tasks

Next, we benchmark against a collection of 18 datasets introduced in Dalla-Torre et al. (2023) and derived from five peer-reviewed studies (Phaml et al., 2005; Oubounyt et al., 2019; Wang et al., 2019; Scalzitti et al., 2021; Geng et al., 2022). These datasets contain three task types, including histone marker prediction, regulatory annotation prediction, and splice site annotation prediction. In assessing performance, we adhered to the methodology described in Dalla-Torre et al. (2023), using different metrics for the tasks: Matthews Correlation Coefficient (MCC) for all histone marker tasks and enhancer classification, F1 score for promoter regulatory annotations, and splice site annotation tasks, except for the *splice sites all* task, where we report accuracy. We additionally follow Dalla-Torre et al. (2023) in performing 10-fold CV using different random seeds with early stopping on the validation metric. We report mean and ± on max/min of the 10 seeds. We present the results of the benchmark suite in Table 2, where we again find that Caduceus-Ph performs competitively, even beating much larger attention-based models on 8 of 18 prediction tasks. Caduceus models outperform a similarly sized HyenaDNA model on almost all the histone marker and regulatory annotation tasks, with the exception of splice site annotation.

4.2.3. Predicting the Effect of Variants on Gene Expression

The dataset used in this task is derived from the Enformer paper (Avsec et al., 2021) and presented in Trop et al. (2023). From each model, we extract embeddings centered around the SNP location. We stratify the data by distance of the SNP to nearest Transcription Start Site (TSS). For each bucket, we sample 5,000 training points and fit an SVM classifier with an RBF kernel to predict VEP annotations. We report test set AUCROC mean and max/min ranges for classifiers fit on 5 random training subsets. For more details about this experiment, please refer to Appendix D.3. We compare Caduceus to HyenaDNA and Nucleotide Transformer, as well as to the supervised baseline Enformer (Avsec et al., 2021).

As shown in Figure 3, the Caduceus models consistently outperform HyenaDNA, and Caduceus-PS exceeds the performance of the Nucleotide Transformer v2 (with 500M parameters). Of note, on sequences where distance to TSS exceeds 100k, Caduceus even outperforms the Enformer baseline.

5. Conclusion

We introduced architectural innovations to the Mamba module, enabling bi-directional and RC equivariant sequence modeling. We also propose a new DNA foundation model, Caduceus, which outperforms comparably sized uni-directional Hyena-based models and much larger Transformer-based models on biologically relevant tasks.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

Avsec, Ž., Agarwal, V., Visentin, D., Ledsam, J. R., Grabska-Barwinska, A., Taylor, K. R., Assael, Y., Jumper, J., Kohli, P., and Kelley, D. R. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.

Benegas, G., Albors, C., Aw, A. J., Ye, C., and Song, Y. S. Gpn-msa: an alignment-based dna language model for genome-wide variant effect prediction. *bioRxiv*, 2023a.

Benegas, G., Batra, S. S., and Song, Y. S. Dna language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120(44):e2311219120, 2023b.

Blelloch, G. E. Prefix sums and their applications. 1990.

Cao, Z. and Zhang, S. Simple tricks of convolutional neural network architectures improve dna–protein binding prediction. *Bioinformatics*, 35(11):1837–1843, 2019.

Consortium, G. R. et al. Genome reference consortium human build 37 (grch37). *Database (GenBank or RefSeq)*, 2009.

Dalla-Torre, H., Gonzalez, L., Mendoza-Revilla, J., Carranza, N. L., Grzywaczewski, A. H., Oteri, F., Dallago, C., Trop, E., de Almeida, B. P., Sirelkhatim, H., et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pp. 2023–01, 2023.

Dao, T., Fu, D. Y., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Falcon, W. and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL `https://github.com/Lightning-AI/lightning`.

Fishman, V., Kuratov, Y., Petrov, M., Shmelev, A., Shepelin, D., Chekanov, N., Kardymon, O., and Burtsev, M. Gena-lm: A family of open-source foundational models for long dna sequences. *bioRxiv*, pp. 2023–06, 2023.

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

Grešová, K., Martinek, V., Čechák, D., Šimeček, P., and Alexiou, P. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

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

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021a.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34: 572–585, 2021b.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35: 35971–35983, 2022.

Gündüz, H. A., Binder, M., To, X.-Y., Mreches, R., Bischl, B., McHardy, A. C., Münch, P. C., and Rezaei, M. A self-supervised deep learning method for data-efficient training in genomics. *Communications Biology*, 6(1):928, 2023.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Ji, Y., Zhou, Z., Liu, H., and Davuluri, R. V. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Madani, A., Krause, B., Greene, E. R., Subramanian, S., Mohr, B. P., Holton, J. M., Olmos Jr, J. L., Xiong, C., Sun, Z. Z., Socher, R., et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pp. 1–8, 2023.

Mallet, V. and Vert, J.-P. Reverse-complement equivariant networks for dna sequences. *Advances in Neural Information Processing Systems*, 34:13511–13523, 2021.

Martin, E. and Cundy, C. Parallelizing linear recurrent neural nets over sequence length. *arXiv preprint arXiv:1709.04057*, 2017.

Nguyen, E., Poli, M., Faizi, M., Thomas, A., Birch-Sykes, C., Wornow, M., Patel, A., Rabideau, C., Massaroli, S., Bengio, Y., et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023.

Oubounyt, M., Louadi, Z., Tayara, H., and Chong, K. T. Deepromoter: robust promoter predictor using deep learning. *Frontiers in genetics*, 10:286, 2019.

pandas development team, T. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Phaml, T. H., Tran, D. H., Ho, T. B., Satou, K., and Valiente, G. Qualitatively predicting acetylation and methylation areas in dna sequences. *Genome Informatics*, 16(2):3–11, 2005.

Quang, D. and Xie, X. Factornet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. *Methods*, 166:40–47, 2019.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., and Rives, A. Transformer protein language models are unsupervised structure learners. *Biorxiv*, pp. 2020–12, 2020.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

Scalzitti, N., Kress, A., Orhand, R., Weber, T., Moulinier, L., Jeannin-Girardon, A., Collet, P., Poch, O., and Thompson, J. D. Spliceator: Multi-species splice site prediction using convolutional neural networks. *BMC bioinformatics*, 22 (1):1–26, 2021.

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

Smith, J. T., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Tillet, P., Kung, H.-T., and Cox, D. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Trop, E., Kao, C.-H., Polen, M., Schiff, Y., de Almeida, B. P., Gokaslan, A., Pierrot, T., and Kuleshov, V. Advancing dna language models: The genomics long-range benchmark. In *LLMs4Bio Workshop*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, G., Sarkar, A., Carbonetto, P., and Stephens, M. A simple new approach to variable selection in regression, with application to genetic fine mapping. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(5):1273–1300, 2020.

Wang, J., Yan, J. N., Gu, A., and Rush, A. M. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022.

Wang, R., Wang, Z., Wang, J., and Li, S. Splicefinder: ab initio prediction of splice sites using convolutional neural network. *BMC bioinformatics*, 20:1–13, 2019.

Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60): 3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zhou, H., Shrikumar, A., and Kundaje, A. Towards a better understanding of reverse-complement equivariance for deep learning models in regulatory genomics. *BioRxiv*, pp. 2020, 2021.

Zhou, J. and Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

Zhou, Z., Ji, Y., Li, W., Dutta, P., Davuluri, R., and Liu, H. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

A. Related Work

A.1. DNA Language Models

Transformer-based DNA LMs, such as DNABERT, v1 (Ji et al., 2021) and v2 (Zhou et al., 2023), and Nucleotide Transformer (Dalla-Torre et al., 2023) have been restricted by the quadratic scaling of Transformers, with maximum context sizes of up to roughly 12,000 bps. BigBird (Zaheer et al., 2020) (and GENA-LM (Fishman et al., 2023), which uses BigBird as a backbone) use sparse attention to scale context size up to an order of magnitude large.

Notably, GPN (Benegas et al., 2023a;b), uses dilated convolutional layers, which in practice scale to large receptive fields, although a context size of only 512 bps is used when training this model. Benegas et al. (2023b) find that DNA LM are powerful unsupervised variant effect predictors.

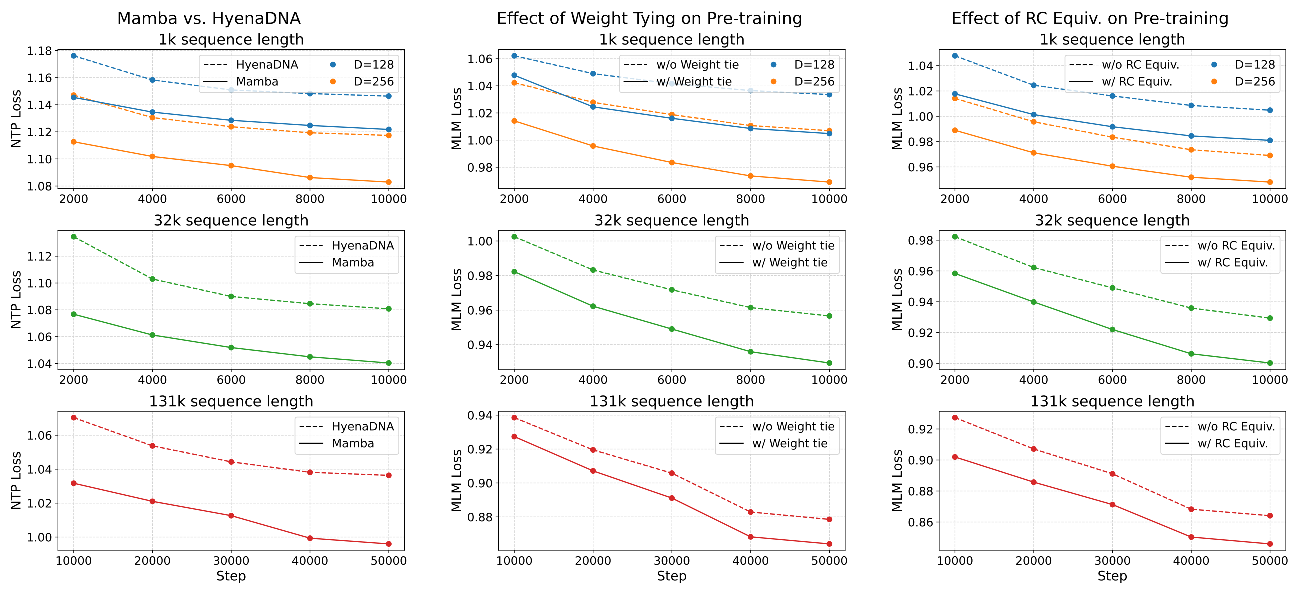
HyenaDNA Most related to our work is the HyenaDNA model (Nguyen et al., 2023), which uses the Hyena operator, derived from the SSM literature, as the building block for a DNA LM. HyenaDNA is able to scale to long-range sequences (up to 1 million bps), but is notably a uni-directional model and not inherently robust to RC inputs.

A.2. Reverse Complement Training for DNA

Cao & Zhang (2019) discuss the importance of RC data augmentation in genomics. Shrikumar et al. (2017) introduce RC Parameter Sharing (RCPS) for convolution, batch normalization, and pooling modules. Mallet & Vert (2021) formalize RC equivariance in the language of Group representations and cast RCPS as a particular decomposition of such representations, exploring other as well. Our implementation of RCPS in the MambaDNA block differs from that proposed in Shrikumar et al. (2017) in that our split operation prevents the channel dimension from doubling when passing a sequence through a given layer.

Zhou et al. (2021) further explore RCPS layers and compare them to a *post-hoc* conjoining baseline, which serves as the inspiration for our Caduceus-Ph model. Zhou et al. (2021) find that post-hoc conjoining is a strong baseline that often outperforms RCPS models on several tasks. We note that Zhou et al. (2021) focus on supervised training regimes, whereas we extend the post-hoc conjoining methodology to include a LM pre-training step as well. Prediction conjoining was also explored in DeepBind (Alipanahi et al., 2015), where max aggregation as opposed to averaging is used, and FactorNet (Quang & Xie, 2019), which performs conjoining during training and inference.

Finally, similar to our setup, Gündüz et al. (2023) explore RC sequences in self supervised pre-training. However, their model uses contrastive learning during pre-training where an encoder is trained to recognize the embeddings of the RC sequence in a given batch.



(a) (b) (c)

Figure 4. Pre-training test set loss. (a) For comparable model size and sequence length, Mamba attains better cross entropy loss than HyenaDNA during pre-training on the human genome. (b) Across sequence lengths, deeper models that use weight tying have better pre-training loss on the human genome. (c) Across sequence lengths, RC equivariance leads to better pre-training loss on the human genome. Note, models with a sequence length of 131k were validated less frequently to reduce overhead during pre-training. By adjusting batch size, we hold number of tokens per batch constant across varying lengths.

A.3. Bi-directional RNNs

Exploiting bi-directionality for pre-training on large datasets was first realized in ELMo (Peters et al., 2017), where forward and backward LSTMs (Hochreiter & Schmidhuber, 1997) were utilized simultaneously to model language context. This laid the groundwork for models such as BERT (Devlin et al., 2018) that replaced recurrent networks with a Transformer backbone. Recently, Wang et al. (2022) explored BERT-style training using SSMs. In concurrent work, Zhu et al. (2024) also extend the Mamba SSM to be bi-directional, similarly combining outputs of forward and backward sequence operators.

B. Background

B.1. DNA Terminology

Deoxyribonucleic acid (DNA) is a polymer that is made up of two complementary strands that wind in ladder / double-helix manner and is comprised of four *nucleotide* bases: *adenine* (`A`), *cytosine* (`C`), *guanine* (`G`) or *thymine* (`T`). The bonds between the nucleotide bases form 'rungs' on the twisted ladder, with `A` bonding with `T` and `C` bonding with `G`. DNA contains the genetic code for forming proteins. In complex organisms, DNA can be billions of nucleotide base pairs (bps) long, but the long strands coil tightly around proteins in the nucleus called *histones*.

Genetic mutations at individual bps, known as *single nucleotide polymorphisms* (SNPs) can account for phenotypic variation across organisms. Evolutionary pressure has forced several genomic regions to be conserved across time and species, with deleterious mutations failing to proliferate in populations. Mutations in conserved regions can therefore have an out-sized effect on phenotype, and models that can identify these regions will likely perform better on variant effect prediction tasks.

Reverse Complement Strands In the double-helix DNA structure, each strand contains semantically equivalent information. The 'reverse complement' (RC) of a given strand is oriented in the opposite direction of its counterpart with bps complemented, `A` converted to `T` and `C` to `G`, relative to the 'forward' strand. In many biological assays, either strand of the DNA can be sequenced with equal probability. However, learning to recognize non-palindromic DNA sequence motifs can be difficult for standard models (Zhou et al., 2021). Therefore, enforcing RC equivariance, loosely defined as model outputs transforming in a manner commensurate with RC-ing an input sequence, is an important desiderata of DNA sequence modeling.

B.2. Structured State Space Models

A recent class of sequence models known as Structured State Space Models (SSMs[2]; Gu et al. (2021a;b; 2022); Gupta et al.

[2]The acronym SSM is commonly used in machine learning communities to refer to this class of models, while in other disciplines it is typically associated to the broader class of state

(2022); Smith et al. (2022); Dao et al. (2022)) have proven to be effective at handling long-range models. At the core of all of these models is a pair of linear differential equations that govern the mapping from input sequences $x(t) \in \mathbb{R}$ to output sequences $y(t) \in \mathbb{R}$ through an intermediate representation $\boldsymbol{h}(t) \in \mathbb{R}^N$:

$$\dot{\boldsymbol{h}}(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t), \quad y(t) = \boldsymbol{C}h(t) + \boldsymbol{D}x(t), \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{B} \in \mathbb{R}^{N\times 1}, \boldsymbol{C} \in \mathbb{R}^{1\times N}$, and $\boldsymbol{D} \in \mathbb{R}$ are the parameters of the system. For multidimensional sequences, $\boldsymbol{x}(t), \boldsymbol{y}(t) \in \mathbb{R}^D$, these dynamics are applied independently to each component.

This differential equation can be discretized with the continuous parameters converted, as follows:

$$\boldsymbol{h}_{t+1} = \overline{\boldsymbol{A}}\boldsymbol{h}_t + \overline{\boldsymbol{B}}x_t, \quad y_{t+1} = \boldsymbol{C}\boldsymbol{h}_t + \boldsymbol{D}x_t, \tag{2}$$

by means of some discretization formula that is a function of continuous parameters $\boldsymbol{A}$,$\boldsymbol{B}$, and an additional time scale parameter Δ. A common discretization used in the SSM literature is the zero-order hold, defined as:

$$\overline{\boldsymbol{A}} = \exp(\Delta\boldsymbol{A}), \quad \overline{\boldsymbol{B}} = \boldsymbol{A}^{-1}(\exp(\Delta\boldsymbol{A}) - \boldsymbol{I})\boldsymbol{B}. \tag{3}$$

Importantly, the linear-time invariance (LTI) of Equation 1 allows us to equivalently formulate Equation 2 as a convolution by unrolling the recurrence, enabling efficient parallel computation during training.

Selection Mechanisms However, the computational efficiency of the LTI formulation comes at the cost of the model not being able to adapt / attend to specific inputs. To alleviate this lack of expressivity, Gu & Dao (2023) introduce a *selective* SSM that enables dependence of the parameters $\boldsymbol{B}$,$\boldsymbol{C}$, and Δ on the input $x(t)$, with:

$$\begin{aligned} \boldsymbol{B}_t = \mathrm{Linear}_{\boldsymbol{B}}(x_t) \quad & \boldsymbol{C}_t = \mathrm{Linear}_{\boldsymbol{C}}(x_t) \\ \Delta_t = \mathrm{softplus}&(\mathrm{Linear}_{\Delta}(x_t)), \end{aligned} \tag{4}$$

where $\mathrm{Linear}(\cdot)$ represents a linear projection and $\mathrm{softplus}(\cdot) = \log(1+\exp(\cdot))$.

While this formulation renders $\overline{\boldsymbol{A}}_t$ and $\overline{\boldsymbol{B}}_t$ time-dependent, the linear recurrence in Equation 2 can be formulated as an associative scan (Martin & Cundy, 2017), which allows us to use an efficient parallel algorithm (Blelloch, 1990) and reduce computation to a logarithmic in sequence length.

Mamba The Mamba block presented in Gu & Dao (2023) is formed by combining a selective SSM sequence transformation and a gated MLP mechanism. This is depicted in the left-most schematic in Figure 1. An incoming sequence is copied and projected to twice the input dimension. One copy is then passed through a causal convolution, followed by the SiLU/Swish non-linear activation (Ramachandran et al., 2017) and then finally through the selective SSM. The other copy has the SiLU non-linearity applied to it and then gates the SSM output. The gated representation is then projected back to the original dimension D. As this is a causal, left-to-right sequence operation, the original models that use Mamba blocks are trained with the next token prediction (NTP) objective during pre-training.

space models widely used in engineering.

Table 1. Genomic Benchmarks. Top-1 accuracy (↑) across 5-fold cross-validation (CV) for pretrained HyenaDNA, Mamba NTP, Caduceus models, and a supervised CNN baseline (trained from scratch). Best values per task are **bolded**, second best are *italicized*. Error bars indicate the difference between the maximum and minimum values across 5 random seeds used for CV.

	CNN (264K)	HyenaDNA (436K)	Mamba (468K)	Caduceus w/o Equiv. (470K)	Caduceus-Ph (470K)	Caduceus-PS (470K)
Mouse Enhancers	0.715 ±0.087	*0.780* ±0.025	0.743 ±0.054	0.770 ±0.058	0.754 ±0.074	**0.793** ±0.058
Coding vs. Intergenomic	0.892 ±0.008	0.904 ±0.005	0.904 ±0.004	0.908 ±0.003	**0.915** ±0.003	*0.910* ±0.003
Human vs. Worm	0.942 ±0.002	0.964 ±0.002	0.967 ±0.002	*0.970* ±0.003	**0.973** ±0.001	0.968 ±0.002
Human Enhancers Cohn	0.702 ±0.021	0.729 ±0.014	0.732 ±0.029	0.741 ±0.008	**0.747** ±0.004	*0.745* ±0.007
Human Enhancer Ensembl	0.744 ±0.122	0.849 ±0.006	0.862 ±0.008	0.883 ±0.002	*0.893* ±0.008	**0.900** ±0.006
Human Regulatory	0.872 ±0.005	0.869 ±0.012	0.814 ±0.211	0.871 ±0.007	*0.872* ±0.011	**0.873** ±0.007
Human OCR Ensembl	0.698 ±0.013	0.783 ±0.007	0.815 ±0.002	0.818 ±0.003	**0.828** ±0.006	*0.818* ±0.006
Human NonTATA Promoters	0.861 ±0.009	0.944 ±0.002	0.933 ±0.007	0.933 ±0.006	**0.946** ±0.007	*0.945* ±0.010

Table 2. Nucleotide Transformer Tasks. Performance (↑) across 10-fold CV for Enformer, DNABERT-2, Nucleotide Transformer v2, HyenaDNA, Caduceus-Ph, and Caduceus-PS. Metrics vary by task: MCC for histone markers and enhancer annotation, F1-score for promoter annotation and splice site acceptor/donor, and accuracy for splice site "all". Best values per task are **bolded**, second best are *italicized*. Given the disparity in model size, we also underline the best value within the SSM-based models. Error bars indicate the difference between the maximum and minimum values across 10 random seeds used for CV.

	> 100M Param. Models			< 2M Param. Models		
	Enformer (252M)	DNABERT-2 (117M)	NT-v2 (500M)	HyenaDNA (1.6M)	Caduceus-Ph (1.9M)	Caduceus-PS (1.9M)
Histone Markers						
H3	0.719±0.048	0.785±0.033	0.784±0.047	0.779±0.037	**0.815**±0.048	*0.799*±0.029
H3k14ac	0.288±0.077	0.516±0.028	0.551±0.021	*0.612*±0.065	**0.631**±0.026	0.541±0.212
H3k36me3	0.344±0.055	0.591±0.020	**0.625**±0.013	*0.613*±0.041	0.601±0.129	0.609±0.109
H3k4me1	0.291±0.061	0.511±0.028	**0.550**±0.021	0.512±0.024	*0.523*±0.039	0.488±0.102
H3k4me2	0.211±0.069	0.336±0.040	0.319±0.045	*0.455*±0.095	**0.487**±0.170	0.388±0.101
H3k4me3	0.158±0.072	0.352±0.077	0.410±0.033	**0.549**±0.056	*0.544*±0.045	0.440±0.202
H3k79me3	0.496±0.042	0.613±0.030	0.626±0.026	0.672±0.048	**0.697**±0.077	*0.676*±0.026
H3K9ac	0.420±0.063	0.542±0.029	0.562±0.040	0.581±0.061	**0.622**±0.030	*0.604*±0.048
H4	0.732±0.076	0.796±0.027	*0.799*±0.025	0.763±0.044	**0.811**±0.022	0.789±0.020
H4ac	0.273±0.063	0.463±0.041	0.495±0.032	*0.564*±0.038	**0.621**±0.054	0.525±0.240
Regulatory Annotation						
Enhancer	0.451±0.108	0.516±0.098	**0.548**±0.144	0.517±0.117	*0.546*±0.073	0.491±0.066
Enhancer types	0.309±0.134	0.423±0.051	*0.424*±0.132	0.386±0.185	**0.439**±0.054	0.416±0.095
Promoter: All	0.954±0.006	*0.971*±0.006	**0.976**±0.006	0.960±0.005	0.970±0.004	0.967±0.004
NonTATA	0.955±0.010	*0.972*±0.005	**0.976**±0.005	0.959±0.008	0.969±0.011	0.968±0.006
TATA	*0.960*±0.023	0.955±0.021	**0.966**±0.013	0.944±0.040	0.953±0.016	0.957±0.015
Splice Site Annotation						
All	0.848±0.019	0.939±0.009	**0.983**±0.008	*0.956*±0.011	0.940±0.027	0.927±0.021
Acceptor	0.914±0.028	*0.975*±0.006	**0.981**±0.011	0.958±0.010	0.937±0.033	0.936±0.077
Donor	0.906±0.027	*0.963*±0.006	**0.985**±0.022	0.949±0.024	0.948±0.025	0.874±0.289

A. Proof of Theorem 2.1

We begin by reiterating the definitions of the different functions that comprise our RC equivariant Mamba module. For an input sequence $\mathbf{X}_{1:T}^{1:D}$ of length T, with D channels, we define:

$$\mathrm{split}(\mathbf{X}_{1:T}^{1:D}) := \left[\mathbf{X}_{1:T}^{1:(D/2)}, \mathbf{X}_{1:T}^{(D/2):D}\right], \tag{5}$$

$$\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:D}\right) := \mathbf{X}_{T:1}^{D:1}, \tag{6}$$

$$\mathrm{concat}\left(\left[\mathbf{X}_{1:T}^{1:(D/2)}, \mathbf{X}_{1:T}^{(D/2):D}\right]\right) := \mathbf{X}_{1:T}^{1:D}, \tag{7}$$

$$\mathrm{M}_{\mathrm{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:D}\right) := \mathrm{concat}\left(\left[\mathrm{M}_\theta\left(\mathbf{X}_{1:T}^{1:(D/2)}\right), \mathrm{RC}\left(\mathrm{M}_\theta \circ \mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right)\right)\right]\right). \tag{8}$$

We also denote the application of the RC operation to a sequence that is 'split' along the channel dimension as:

$$\mathrm{RC}\left(\left[\mathbf{X}_{1:T}^{1:(D/2)}, \mathbf{X}_{1:T}^{(D/2):D}\right]\right) := \left[\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right), \mathrm{RC}\left(\mathbf{X}_{1:T}^{(1:(D/2))}\right)\right] = \left[\mathbf{X}_{T:1}^{D:(D/2)}, \mathbf{X}_{T:1}^{(D/2):1}\right], \tag{9}$$

Note that the RC operation can be 'pulled inside' of a concat operation:

$$\begin{aligned}\mathrm{RC}\circ\mathrm{concat}\left(\left[\mathbf{X}_{1:T}^{1:(D/2)},\mathbf{X}_{1:T}^{(D/2):D}\right]\right)&=\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:D}\right)\\&=\mathbf{X}_{T:1}^{D:1}\\&=\mathrm{concat}\left(\left[\mathbf{X}_{T:1}^{D:(D/2)},\mathbf{X}_{T:1}^{(D/2):1}\right]\right)\\&=\mathrm{concat}\left(\left[\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:(D/2)}\right),\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right)\right]\right)\\&=\mathrm{concat}\circ\mathrm{RC}\left(\left[\mathbf{X}_{1:T}^{1:(D/2)},\mathbf{X}_{1:T}^{(D/2):D}\right]\right)\end{aligned}\tag{10}$$

Additionally, we have that $\mathrm{RC}^{-1}=\mathrm{RC}$ and that

$$\mathrm{RC}\left(\left[\mathbf{X}_{1:T}^{1:(D/2)},\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right)\right]\right)=\left[\mathbf{X}_{1:T}^{(D/2):D},\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:(D/2)}\right)\right].\tag{11}$$

Following Definition 8, we have that:

$$\begin{aligned}\mathrm{RC}\circ\mathrm{M}_{\text{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:D}\right)&=\mathrm{RC}\circ\mathrm{concat}\left(\left[\mathrm{M}_\theta\left(\mathbf{X}_{1:T}^{1:(D/2)}\right),\mathrm{RC}\left(\mathrm{M}_\theta\circ\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right)\right)\right]\right) && \mathit{(8)}\\&=\mathrm{concat}\circ\mathrm{RC}\left(\left[M_\theta\left(\mathbf{X}_{1:T}^{1:(D/2)}\right),\mathrm{RC}\left(\mathrm{M}_\theta\circ\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right)\right)\right]\right) && \mathit{(10)}\\&=\mathrm{concat}\left(\left[\mathrm{M}_\theta\circ\mathrm{RC}\left(\mathbf{X}_{1:T}^{(D/2):D}\right),\mathrm{RC}\left(\mathrm{M}_\theta\left(\mathbf{X}_{1:T}^{1:(D/2)}\right)\right)\right]\right) && \mathit{(11)}\\&=\mathrm{concat}\left(\left[\mathrm{M}_\theta\left(\mathbf{X}_{T:1}^{D:(D/2)}\right),\mathrm{RC}\left(\mathrm{M}_\theta\circ\mathrm{RC}\left(\mathbf{X}_{T:1}^{(D/2):1}\right)\right)\right]\right) && \mathit{(6)}\\&=\mathrm{M}_{\text{RCe},\theta}\circ\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:D}\right) && \square\end{aligned}$$

B. Proof of Theorem 3.1

We begin with the following lemma,

Lemma B.1. *For two RC equivariant sequence operators* F *and* G*, their composition* $\mathrm{F}\circ\mathrm{G}$ *is also equivariant.*

Proof. We have that,

$$\mathrm{F}\left(\mathrm{G}\left(\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:D}\right)\right)\right)=\mathrm{F}\left(\mathrm{RC}\left(\mathrm{G}\left(\mathbf{X}_{1:T}^{1:D}\right)\right)\right)=\mathrm{RC}\left(\mathrm{F}\left(\mathrm{G}\left(\mathbf{X}_{1:T}^{1:D}\right)\right)\right)$$

where each equality follows from the RC equivariance of the operators G and F, respectively. $\square$

Therefore, to prove that the Caduceus-PS is RC equivariant, we need to prove that each operator in $\mathrm{LM}_{\text{RCe},\theta}\circ\mathrm{M}_{\text{RCe},\theta}^{(n)}\circ\mathrm{Emb}_{\text{RCe},\theta}$ satisfies this property.

First, we show that $\mathrm{Emb}_{\text{RCe},\theta}$ is RC equivariant.

$$\begin{aligned}\mathrm{RC}\circ\mathrm{Emb}_{\text{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:4}\right)&=\mathrm{RC}\circ\mathrm{concat}\left(\left[\mathrm{Emb}_\theta\left(\mathbf{X}_{1:T}^{1:4}\right),\mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:4}\right)\right)\right]\right)\\&=\mathrm{concat}\circ\mathrm{RC}\left(\left[\mathrm{Emb}_\theta\left(\mathbf{X}_{1:T}^{1:4}\right),\mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:4}\right)\right)\right]\right) && \mathit{(10)}\\&=\mathrm{concat}\left(\left[\mathrm{Emb}_\theta\left(\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:4}\right)\right),\mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathbf{X}_{1:T}^{1:4}\right)\right)\right] && \mathit{(11)}\\&=\mathrm{concat}\left(\left[\mathrm{Emb}_\theta\left(\mathbf{X}_{T:1}^{4:1}\right),\mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathbf{X}_{1:T}^{1:4}\right)\right)\right]\\&=\mathrm{concat}\left(\left[\mathrm{Emb}_\theta\left(\mathbf{X}_{T:1}^{4:1}\right),\mathrm{RC}\circ\mathrm{Emb}_\theta\left(\mathrm{RC}\left(\mathbf{X}_{T:1}^{4:1}\right)\right)\right)\right]\\&=\mathrm{Emb}_{\text{RCe},\theta}\circ\mathrm{RC}\left(\mathbf{X}_{1:T}^{1:4}\right) && \square\end{aligned}\tag{12}$$

Additionally, we have that $\mathrm{M}_{\text{RCe},\theta}^{(n)}$ is equivariant by Theorem 2.1 and induction using Lemma B.1.

Finally, recall the definition of $\mathrm{LM}_{\text{RCe},\theta}$:

$$\mathrm{LM}_{\text{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:D}\right):=\mathrm{LM}_\theta\left(\mathbf{X}_{1:T}^{1:(D/2)}\right)+\text{flip\_chan}\circ\mathrm{LM}_\theta\left(\mathbf{X}_{1:T}^{D:(D/2)}\right).$$

Note that LM_θ is parameterized by a weight matrix $\boldsymbol{W}_\theta$ and applying LM_θ to a sequence $\mathbf{X}_{1:T}^{1:(D/2)}$ is equivalent to multiplying each of the sequence elements $\mathbf{x}_t^{1:(D/2)}$, for $t=1,\ldots,T$, on the left by $\boldsymbol{W}_\theta$. Therefore if we reverse an input to LM_θ along the length dimension, the output will be reversed along the length dimension as well. We can thus focus on a specific item at position t in a sequence:

$$\mathrm{LM}_{\text{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:D}\right)_t=\boldsymbol{W}_\theta\cdot\mathbf{x}_t^{1:(D/2)}+\text{flip\_chan}\left(\boldsymbol{W}_\theta\cdot\mathbf{x}_t^{D:(D/2)}\right),$$

and we need only show that it is equivariant with the $\mathrm{flip\_chan}$ operation, which we recall merely reverses the channels of given input. We note that $\mathrm{flip\_chan}^{-1} = \mathrm{flip\_chan}$. Now we show that:

$$\begin{aligned}\mathrm{flip\_chan}\left(\mathrm{LM}_{\mathrm{RCe},\theta}\left(\mathbf{X}_{1:T}^{1:D}\right)_t\right) &= \mathrm{flip\_chan}\left(\boldsymbol{W}_\theta \cdot \mathbf{x}_t^{1:(D/2)}\right) + \boldsymbol{W}_\theta \cdot \mathbf{x}_t^{D:(D/2)} \\ &= \mathrm{LM}_{\mathrm{RCe},\theta}\left(\mathrm{flip\_chan}\left(\mathbf{X}_{1:T}^{1:D}\right)\right)_t\end{aligned}$$

This completes the proof. □

C. Pre-training

We provide a more detailed description of the dataset and training methodology used in the human reference genome pre-training task. This dataset is based on the splits used in the previous Enformer study (Avsec et al., 2021). The training split comprises 34,021 segments that we extend to a maximum length of 1,048,576 (2^{20}), collectively covering the genome and amounting to around 35 billion tokens, or nucleotide base pairs.

All the Mamba-based models, including Caduceus, were trained with a learning rate of $8\mathrm{e}^{-3}$. We maintain a constant number of tokens in each batch, using 2^{20} tokens per batch. For example, for sequence lengths of 1,024, batch size is also 1,024 and for sequence lengths of 131k (2^{17}), batch size is 8. All our models, other than Caduceus-PS, are pre-trained with RC data augmentation, where any given sequence is either unchanged or has the RC operation applied to it with equal probability.

Models were trained with cosine decay and the ADAM optimization algorithm (Kingma & Ba, 2014), β_1 and β_2 values of 0.95 and 0.9, respectively.

For bi-directional models, we use the masking recipe presented in Devlin et al. (2018). Namely, we 'mask' 15% of tokens. Of the 'masked' tokens, 80% are replaced with a special [MASK] token, 10% are replaced with a random token from the vocabulary, and 10% are left unchanged.

The various Mamba/Caduceus models that were pre-trained are listed in Table 3. For Figure 4a, we re-pre-train HyenaDNA models on sequence lengths of 1,024, 32k, and 131k. We use the corresponding hidden dimension and depth as those used when these models were originally trained in Nguyen et al. (2023). Other than learning rate, which was set to $6\mathrm{e}^{-4}$, all the other pre-training details used for our models above were used for HyenaDNA pre-training as well.

Table 3. Pre-trained Mamba-based models with corresponding sequence length, depth, hidden dimension, and number of gradient updates.

Seq. Len.	Hidden Dim.	Num. Layers	Gradient Updates	Uni	Bi-directional	Bi-directional RC Equiv.
1k	118	4	10k		✓	✓
1k	128	4	10k	✓	✓	✓
1k	256	4	10k	✓	✓	✓
32k	256	8	10k	✓	✓	✓
131k	256	16	50k	✓	✓	✓

D. Downstream Tasks

D.1. Genomics Benchmark

For the Genomics Benchmark tasks, we deviate from the results presented in Nguyen et al. (2023) in order to maintain 'true' train and test splits. We therefore, elect to use 5-fold cross-validation where we split the training set into 90/10 train/validation splits and perform early stopping on the validation set. Models were fine-tuned for 10 epochs. The HyenaDNA model consists of 2 layers and hidden dimension 128. It is fine-tuned with a learning rate of $6\mathrm{e}^{-4}$ and batch size of 256. Weights for this pre-trained model were downloaded from `https://huggingface.co/LongSafari/hyenadna-tiny-1k-seqlen`. Following Nguyen et al. (2023), we also experiment with adding RC data augmentation for HyenaDNA. The best result of this search is presented in Table 1. The values used for RC data augmentation in each task are presented in Table 4.

The CNN baseline is described in Grešová et al. (2023). It is trained from scratch with a learning rate of $1\mathrm{e}^{-3}$ and batch size of 64. The CNN consists of an embedding layer and convolutional layers with 16, 8, and 4 channels. The first layer is followed by a ReLU non-linearity and all layers are followed by batch normalization and 1D max-pooling. Finally there are two fully connected layers at the end of the network.

The Caduceus and Mamba models were fine-tuned with a batch size of 256. For the learning rate, we performed hyperparameter tuning, searching within $\{1e^{-3}, 2e^{-3}\}$, and present the best result across cross-valildation, as shown in Table 5. Mamba models consist of 4 layers with hidden dimension 128 and Caduceus models consist of 4 layers with hidden dimension 118 (to keep parameter counts roughly equivalent). For both Caduceus-Ph and Caduceus-PS the forward and RC sequence representations are pooled and then averaged. For Caduceus-PS, this averaging is done during both downstream training and inference. For Caduceus-Ph, this is done only during inference.

Table 4. Hyena Hyperparameter Selection for Genomic Benchmarks. The HyenaDNA model, chosen for its top-1 accuracy averaged over 5-fold cross-validation, includes options for using or not using the RC data augmentation during pre-training.

Mouse Enhancers	No RC Augmentation
Coding vs. Intergenomic	No RC Augmentation
Human vs. Worm	RC Augmentation
Human Enhancers Cohn	RC Augmentation
Human Enhancer Ensembl	No RC Augmentation
Human Regulatory	RC Augmentation
Human OCR Ensembl	RC Augmentation
Human NonTATA Promoters	No RC Augmentation

Table 5. Mamba / Caduceus Hyperparameter Selection for Genomic Benchmarks. Learning rate chosen for its top-1 accuracy averaged over 5-fold cross-validation.

	Mamba	Caduceus w/o Equiv.	Caduceus-Ph	Caduceus-PS
Mouse Enhancers	$2e^{-3}$	$2e^{-3}$	$2e^{-3}$	$2e^{-3}$
Coding vs. Intergenomic	$2e^{-3}$	$1e^{-3}$	$2e^{-3}$	$1e^{-3}$
Human vs. Worm	$2e^{-3}$	$1e^{-3}$	$2e^{-3}$	$1e^{-3}$
Human Enhancers Cohn	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$2e^{-3}$
Human Enhancer Ensembl	$2e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$
Human Regulatory	$1e^{-3}$	$2e^{-3}$	$2e^{-3}$	$1e^{-3}$
Human OCR Ensembl	$2e^{-3}$	$2e^{-3}$	$2e^{-3}$	$2e^{-3}$
Human NonTATA Promoters	$1e^{-3}$	$2e^{-3}$	$2e^{-3}$	$2e^{-3}$

D.2. Nucleotide Transformer Tasks

For the Nucleotide Transformer Task, we pull baseline results from `https://huggingface.co/spaces/InstaDeepAI/nucleotide_transformer_benchmark`. For our Caduceus / Mamba-based models we follow the same CV protocol from Dalla-Torre et al. (2023) using a 90/10 train/validation split for each fold. Our models consist of 4 layers and hidden dimension 256, roughly matching the parameter count of the reported HyenaDNA model. Models were fine-tuned for 20 epochs. Hyperparameters for the models reported in Table 2 can be found in Table 6

D.3. Predicting the Effect of Variants on Gene Expression

Labels for this task represent whether a SNP has a causal effect on gene expression. A positive label is assigned if the causal probability, as determined by the SuSiE (Wang et al., 2020) tool, is $> .9$ (see Avsec et al. (2021), where this task was originally proposed, for more details). Chromosomes 9 and 10 are used as the held out test set (see Trop et al. (2023) for more details).

We follow the methodology presented in Trop et al. (2023) and extract embeddings for each model by taking an average of a 1536 bp window centered at the SNP location for both reference and alternative sequences and concatenating along the channel dimension. Based on the tokenization scheme, for each model this window corresponds to a different number of tokens. Namely, for HyenaDNA and Caduceus models, since base-pair-tokenization was used, the window consists of 1536 tokens as well. Since Nucleotide Transformer was trained using 6-mer tokenization, the window corresponds to 256 bps. Finally, for Enformer, the final embedding has a 'receptive field' of 128 bps, hence a window of 12 'tokens' / positions is used. To each embedding we also concatenate the tissue from which the sequence was assayed.

We also use a different input sequence length for each model. For Caduceus and Hyena models, we use inputs of length 131k bps. For Nucleotide Transformer, we use inputs of length 12k bps, which correspond to the input length on which this model was originally trained. For Enformer, we use inputs of 196k bps, which correspond to the input length on which this model was originally trained.

Table 6. Caduceus Hyperparameter Selection for Nucleotide Transformer Tasks. Caduceus-Ph and Caduceus-PS fine-tuning hyperparameters chosen based on best performance averaged over 10-fold cross-validation.

		Caduceus-Ph		Caduceus-PS	
		LR	batch size	LR	batch size
Histone markers	H3	$1e^{-3}$	128	$1e^{-3}$	128
	H3k14ac	$1e^{-3}$	128	$1e^{-3}$	128
	H3k36me3	$1e^{-3}$	128	$1e^{-3}$	128
	H3k4me1	$1e^{-3}$	512	$1e^{-3}$	128
	H3k4me2	$1e^{-3}$	128	$1e^{-3}$	512
	H3k4me3	$1e^{-3}$	512	$1e^{-3}$	512
	H3k79me3	$1e^{-3}$	128	$1e^{-3}$	128
	H3K9ac	$1e^{-3}$	128	$1e^{-3}$	128
	H4	$1e^{-3}$	128	$1e^{-3}$	128
	H4ac	$1e^{-3}$	128	$1e^{-3}$	128
Regulatory annotation	Enhancers	$1e^{-3}$	512	$1e^{-3}$	512
	Enhancers types	$1e^{-3}$	512	$2e^{-3}$	512
	Promoter all	$1e^{-3}$	512	$1e^{-3}$	128
	Promoter no tata	$1e^{-3}$	512	$1e^{-3}$	128
	Promoter tata	$1e^{-3}$	128	$1e^{-3}$	512
Splice site annotation	Splice sites acceptors	$1e^{-3}$	128	$1e^{-3}$	128
	Splice sites all	$1e^{-3}$	512	$1e^{-3}$	512
	Splice sites donors	$1e^{-3}$	128	$1e^{-3}$	128

We then train an SVM classifier with an RBF kernel on these embeddings for each strata of the data, which is separated by distance to nearest TSS. For each bucket of distance to TSS, we randomly select 5,000 training points, fit an SVM with RBF kernel classifier, and record test set AUROC. We repeat this process five times and report mean and +/- of one standard deviation across seeds.

Hyperparameter optimization was performed for each model within each distance category, focusing on the regularization strength. We select this hyperparameter based on highest mean AUROC reported from 5 random seeds. The regularization strength used for each model reported in Figure 3 are listed in Table 7.

Pre-trained weights for HyenaDNA were downloaded from `https://huggingface.co/LongSafari/hyenadna-medium-160k-seqlen-hf`. Pre-trained weights for Nucleotide Transformer were downloaded from `https://huggingface.co/InstaDeepAI/nucleotide-transformer-v2-500m-multi-species`. Pre-trained weights for the Enformer model were downloaded from `https://huggingface.co/EleutherAI/enformer-official-rough`.

Table 7. Hyperparameter Selection for SVM classifier in variant effect prediction task. Inverse of the L_2 regularization weight selected from $\{1,5,10\}$ by evaluating average test set AUROC.

	Distance to Nearest TSS (bp)		
	$0-30$k	$30-100$k	100k+
Enformer	1	1	5
NTv2	1	1	10
HyenaDNA	1	1	5
Caduceus w/o Equiv	1	1	10
Caduceus-Ph	1	5	10
Caduceus-PS	1	1	5

E. Assets

E.1. Datasets

For pre-training we use the HG38 human reference genome (Consortium et al., 2009). The Genomics Benchmark comes from Grešová et al. (2023). The Nucleotide Transformers benchmark is introduced in Dalla-Torre et al. (2023). The variant effect prediction task data was originally proposed in Avsec et al. (2021) and we use the modified version from Trop et al. (2023).

E.2. Software and Libraries

In Table 8, we enumerate the relevant open-source software, and corresponding licenses, used in this work.

Table 8. Open source libraries used in this work, with corresponding licenses.

Library	License
GenomicsBenchmark (Grešová et al., 2023)	Apache 2.0
Enformer PyTorch	MIT
Mamba (Gu & Dao, 2023)	Apache 2.0
HuggingFace (Wolf et al., 2019)	Apache 2.0
Hydra (Yadan, 2019)	MIT
HyenaDNA (Nguyen et al., 2023)	Apache 2.0
NumPy (Harris et al., 2020)	NumPy license
Matplotlib (Hunter, 2007)	Matplotib license
ML Collections	Apache 2.0
OmegaConf	BSD 3-Clause
Pandas (pandas development team, 2020)	BSD 3-Clause "New" or "Revised"
PyTorch (Paszke et al., 2019)	BSD-3 Clause
PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019)	Apache 2.0
Scikit-Learn (Pedregosa et al., 2011)	BSD 3-Clause
Seaborn (Waskom, 2021)	BSD 3-Clause "New" or "Revised"
Triton (Tillet et al., 2019)	MIT

F. Computational resources

Model training and inference were run on GPUs with number of devices and machine type varying by model size during pre-training and downstream tasks. We use 3090, A5000, A6000, V100, and A100 GPUs.