

# A foundational model for joint sequence-function multi-species modeling at scale for long-range genomic prediction

Sam Boshar<sup>\*1</sup>, Benjamin Evans<sup>\*1</sup>, Ziqi Tang<sup>\*1</sup>,

Armand Picard<sup>1</sup>, Yanis Adel<sup>1</sup>, Franziska K. Lorbeer<sup>2</sup>, Chandana Rajesh<sup>1</sup>, Tristan Karch<sup>1</sup>, Shawn Sidbon<sup>1</sup>, David Emms<sup>1</sup>, Javier Mendoza-Revilla<sup>1</sup>, Fatimah Al-Ani<sup>1</sup>, Evan Seitz<sup>1</sup>, Yair Schiff<sup>4</sup>, Yohan Bornachot<sup>1</sup>, Ariana Hernandez<sup>1</sup>, Marie Lopez<sup>1</sup>, Alexandre Laterre<sup>1</sup>, Karim Beguir<sup>1</sup>, Peter Koo<sup>5</sup>, Volodymyr Kuleshov<sup>4</sup>, Alexander Stark<sup>2,3</sup>, Bernardo P. de Almeida<sup>\*\*1</sup> and Thomas Pierrot<sup>\*\*1</sup>

<sup>\*</sup>Equal contributions, <sup>\*\*</sup>Corresponding authors: b.dealmeida@instadeep.com, t.pierrot@instadeep.com, <sup>1</sup>InstaDeep, London, UK,

<sup>2</sup>Research Institute of Molecular Pathology (IMP), Vienna BioCenter (VBC), Vienna, Austria, <sup>3</sup>Medical University of Vienna, Vienna BioCenter (VBC), Vienna, Austria, <sup>4</sup>Cornell Tech, New York, USA, <sup>5</sup>Cold Spring Harbor Laboratory, New York, USA

Genomic prediction and design require models that integrate local sequence features with long-range regulatory dependencies spanning hundreds of kilobases to megabases. Existing approaches have made substantial progress along complementary axes: supervised sequence-to-function models achieve high accuracy for specific assays and organisms, self-supervised genomic foundation models learn transferable representations from large-scale sequence data, and conditional generative models enable principled sequence design guided by functional objectives. However, these strengths are typically realized in isolation—across distinct model classes, architectures, and training regimes—limiting the ability to combine long-context, base-resolution prediction, functional modeling, and controllable generation within a single efficient framework that generalizes across organisms and modalities.

Here we introduce Nucleotide Transformer v3 (NTv3), a multi-species foundation model that unifies representation learning, functional-track and genome-annotation prediction, and controllable sequence generation within a common backbone. NTv3 uses a U-Net-like architecture to enable single-base tokenization and efficient modeling of contexts up to 1 Mb. We pre-train NTv3 on 9 trillion base pairs from OpenGenome2 using base-resolution masked language modeling, followed by post-training with a joint objective that integrates continued self-supervision with supervised learning on ~16,000 functional tracks and annotation labels from 24 animal and plant species. After post-training, NTv3 achieves state-of-the-art accuracy for functional-track prediction and genome annotation across species, outperforming leading sequence-to-function and foundation-model baselines on established benchmarks and on the new NTV3 BENCHMARK, a controlled downstream fine-tuning suite in a standardized 32 kb input / base-resolution output setting. We further show that NTv3 consolidates a shared regulatory grammar across tasks, enabling coherent long-range genome-to-function inference and variant-associated remodeling. Finally, we fine-tune NTv3 into a controllable generative model via masked diffusion language modeling and use it to design enhancer sequences with specified activity levels and promoter selectivity. We validate these designs experimentally, showing that generated enhancers recapitulate the intended activity stratification and achieve the desired promoter-specific activation *in cellulo*. We release the NTv3 model family together with code and practical cookbooks for long-context training, multispecies post-training, fine-tuning, interpretation, and sequence design.

**Keywords:** Foundation models, genomics, multimodal, generative

## 1. Introduction

Decoding the genome by linking primary sequence to gene regulation, cell-state-specific expression programs, and the phenotypic consequences of genetic variation is a central objective in modern quantitative biology and underpins applications ranging from variant interpretation to agricultural optimization. Deep learning has become a key driver of progress in this area by enabling models that learn predictive sequence features directly from data, often at nucleotide resolution, and by providing principled attribution tools that help translate predictive signals into mechanistic hypotheses [1, 2].

A major avenue of this progress has been the development of *sequence-to-function* models that map DNA sequence windows to experimentally measured functional readouts, such as chromatin accessibility, DNA methylation, transcription factor binding, histone modifications, chromatin conformation, gene expression and mRNA processing [3–16]. Convolutional architectures established strong baselines for high-resolution profile prediction and motif-syntax discovery [5, 6, 17], and subsequent work extended these models to substantially longer receptive fields—often by combining convolutional feature extractors with transformer layers—to improve gene-expression prediction through integration of distal regulatory interactions [18]. More recent models have scaled this paradigm to richer output spaces, unifying multiple regulatory readouts and enabling variant-level interpretation across layers of gene regulation [7, 8].

In parallel, the field has rapidly adopted self-supervised *genomic language models*: large models pretrained on unlabeled DNA using self-supervised objectives to learn transferable representations that can be adapted to downstream tasks through fine-tuning [19–26]. Nucleotide Transformer models demonstrated that masked pretraining over both human and multi-species corpora yields robust representations that transfer across diverse molecular phenotype prediction tasks [21, 27]. A complementary direction is to pre-train long-context sequence models that operate at single-nucleotide resolution, including state-space and long-convolution operators that extend context into the hundreds of kilobases or megabases [22, 26]. Beyond representation learning, generative genomic models have emerged that aim to model and design DNA sequences directly; Evo, for example, scales autoregressive training on multi-kingdom corpora and targets genome modeling and design at up to megabase context [23, 28].

Despite these advances, existing approaches remain fragmented across capabilities and training regimes. Sequence-to-function models typically deliver strong accuracy on specific assay families but require supervised training on narrow domains and do not readily generalize across species or modalities without substantial re-curation and retraining [7]. Conversely, self-supervised foundation models provide broad representations but primarily output token-level probabilities or embeddings that are not directly predictive of molecular phenotypes [29–31]: probability outputs can support tasks such as variant-effect scoring or conservation detection, yet lack the task specificity of sequence-to-function models, while embeddings generally require substantial fine-tuning to produce predictions. More fundamentally, learning from sequence alone is frequently insufficient to capture the condition-specific regulatory variation encoded in large-scale functional genomics datasets. As a result, current methods tend to excel either at representation learning from genomes or at supervised functional prediction, but rarely integrate both within a single, scalable framework that jointly learns from genomic sequence and functional tracks across organisms.

A second, closely related fragmentation separates representation learning from generative modeling for genome design. Generative approaches offer a natural route from genome understanding to genome engineering, yet their objectives and interfaces are typically decoupled from supervised sequence-to-function learning, limiting the extent to which a single model can serve as a unifying substrate across genome understanding and genome engineering [28]. Models optimized for sequence likelihood or reconstruction capture statistical regularities of genomic sequences, but do not necessarily internalize the functional constraints required for precise sequence manipulation. A key open challenge is therefore to develop a single, efficient framework that supports long-range, base-resolution modeling while unifying representation learning, sequence-to-function prediction and generative capabilities across organisms and data types.

Here, we introduce *Nucleotide Transformer v3 (NTv3)*, a multi-species genomics foundation model built to unify these capabilities while operating at a million-nucleotide context (Fig. 1A). NTv3 adopts a U-Net–like architecture to enable single-base tokenization and base-resolution outputs at megabase scale with substantially improved computational efficiency relative to long-context genomic language models (Fig. 1B, see also [8]). While U-Net–style architectures have previously been used for supervised regulatory modeling [7, 8, 18], NTv3

shows that they can be successfully pre-trained at scale as genomic foundation models, and we provide practical recipes that make such long-context pre-training tractable and reproducible.

We train NTv3 using a two-stage strategy that unifies long-context representation learning with base-resolution genome interpretation. Models are first pre-trained on OpenGenome2 [28] using masked language modeling (MLM), scaling context up to 1 Mb via a length curriculum. We then introduce a multispecies post-training regime that integrates continued MLM with supervised objectives for functional genomics and genome annotation across ~16k tracks spanning animals and plants (Fig. 1C). Across benchmarks, NTv3 achieves state-of-the-art performance for functional-track prediction and genome annotation, extending accurate base-resolution prediction to phylogenetically distant species, including plants. To support rigorous evaluation and downstream use, we introduce the NTv3 BENCHMARK, a large-scale benchmark of 106 tasks defined on natural long-context inputs with base-pair-resolution outputs and strict data-leakage controls (Fig. 1D). We further extend NTv3 into a controllable generative model using masked-diffusion language modeling, enabling long-context sequence design while retaining base-pair resolution. Using this framework, we design promoter-specific enhancer sequences with targeted activity profiles and experimentally validate 1,000 constructs (Fig. 1E).

Together, these capabilities position NTv3 as a genuinely foundational genomic model—supporting rich representation learning, multi-task annotation and functional prediction, and controllable sequence generation across diverse cell types and species. We release the full NTv3 model family, inference and fine-tuning code, together with practical “cookbooks” covering efficient long-context pre-training, multispecies post-training, interpretation tools, and sequence-design workflows.



**Figure 1 | Overview of model architecture, training procedure and applications for NTv3.** **A)** Different phases of NTv3 training and applications. **B)** Core U-Net-style architecture for 1 Mb, single-bp modeling, comprising convolutional downsampling, a long-range Transformer core, and deconvolutional upsampling. **C)** Multispecies post-training on functional tracks and genome annotations across 24 species using a joint objective that combines supervised track and annotation prediction with continued MLM. **D)** We introduce a new NTv3 BENCHMARK and evaluate various models through fine-tuning. **E)** Generative extension in which NTv3 is adapted via masked diffusion to enable controllable enhancer design with promoter-specific activity.

## 2. Results

### 2.1. Pre-training NTv3 on genomic sequences up to 1 million nucleotides spanning all domains of life

We developed NTv3, a genomics foundation model designed to learn regulatory logic and genome organization across evolutionarily diverse species. NTv3 is first pre-trained on the OpenGenome2 dataset [28], which comprises more than 128,000 species genomes and over 8 trillion nucleotides from all domains of life (Fig. 1A). A central objective when building NTv3 was to construct a scalable architecture capable of modeling single-base-pair tokenization over one-million-nucleotide contexts, thereby enabling the model to capture distal enhancer-promoter interactions, chromatin-scale structure, operon organization, and long-range intergenic dependencies that remain inaccessible to standard genomic architectures.

To make this possible, NTv3 employs a U-Net-inspired architecture (Fig. 2B and Supplementary Fig. A.1) [7, 8], consisting of a convolutional downsampling tower that compresses the input sequence, a transformer bottleneck that integrates global context, and a symmetric upsampling tower that restores predictions at 1 bp resolution (see Methods 4.1.1). Skip connections across resolutions preserve local detail while enabling efficient long-range integration. While U-Net-style architectures have shown strong performance in supervised sequence-to-function models, NTv3 represents the first application of such an architecture to large-scale self-supervised pre-training in genomics. This shift introduces unique optimization and scaling challenges but also establishes a new class of efficient representation-learning models that reduce the computational cost of long-range attention while retaining high-fidelity modeling of local sequence features. We provide a detailed “cookbook” describing practical recipes for pre-training U-Net-based genomic foundation models at scale (see Supplementary Notes).

NTv3 is pre-trained on the OpenGenome2 dataset using a base-resolution masked language modeling (MLM) objective [32], in which a constant fraction of nucleotides is replaced by a mask token and the model learns to reconstruct the original bases. OpenGenome2 spans broad phylogenetic and sequence diversity—including eukaryotic genomes, metagenomes, bacterial and archaeal chromosomes, viral and organellar sequences, and diverse classes of mRNA transcripts [28] (see Methods 4.5). This denoising objective, applied at single-nucleotide resolution across taxa, drives NTv3 to learn bidirectional, context-aware representations that link local sequence determinants to long-range genomic structure, producing features that generalize across phylogeny and capture shared principles of genomic organization.

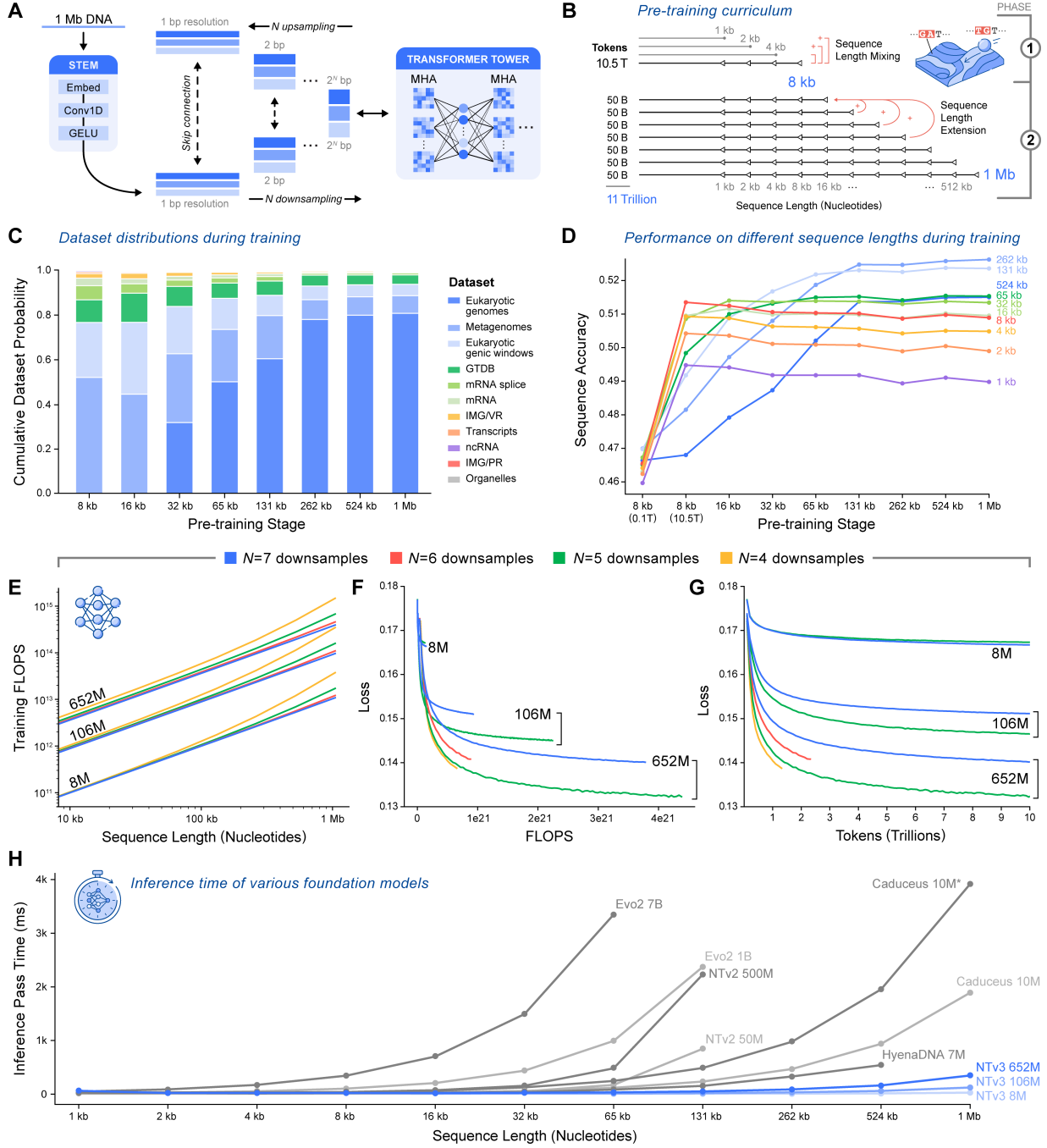
We performed pre-training using a two-stage curriculum to enable efficient scaling to 1 Mb contexts: 8kb-training and sequence length extension (Fig. 2A; see Methods 4.2.2). In the main 8kb-training stage, spanning approximately 10.5 trillion tokens, the model is trained on a mixture of 1–8 kb windows, with sampling weights biased toward genic regions to preferentially learn functional sequence elements. We then transition to sequence length extension, which extends the maximum input length to 1 Mb to capture relationships between regulatory and genic elements across long genomic distances. Sequence length extension proceeds via a sequence of short ~50-billion-token phases in which the context length is doubled at each step, mirroring established long-context training practice in natural language models. Because the OpenGenome2 sub-datasets differ in characteristic sequence lengths, the sampling distribution is adjusted across stages, leading to dynamic shifts in sub-dataset composition over the curriculum (Fig. 2C). Throughout all phases, we retain a controlled mixture of shorter windows, ensuring that short-range competencies are preserved as the receptive field expands and that each sub-dataset remains represented at every stage. Under this curriculum, the model maintains strong reconstruction accuracy across the full range of lengths from 1 kb to 1 Mb (Fig. 2D), and the complete training run spans 11 trillion tokens, constituting the largest trained genomics model to date.

To support different computational budgets and maximize practical adoption, we release three pre-trained NTv3 models that share the same architecture but vary in scale: NTv3 8M (pre), NTv3 100M (pre), and NTv3 650M (pre). This family provides both extremely fast, lightweight models and larger, higher-performing ones. All three display smooth scaling behavior, as shown by our supporting scaling-law analysis.

We additionally performed a systematic exploration of architectural parameters—most notably the number of convolutional downsampling stages—which governs the balance between bottleneck efficiency and retained base-pair detail (Fig. 2E-G). These ablations, together with analyses of depth, width, and compression factors, are reported in Supplementary Note B.2 and are intended as a practical resource for practitioners working with hierarchical genomic architectures.

Despite operating at 1 bp resolution over 1 Mb contexts, even the largest NTv3 model is designed to run





**Figure 2 | Pre-training NTv3 on OpenGenome2.** **A**) NTv3 implements a hierarchical U-Net design comprising a convolutional downsampling tower, a long-range Transformer tower, and a matched deconvolutional upsampling tower that reconstructs nucleotide-resolution representations. Skip connections link corresponding resolution levels to stabilize optimization and preserve fine-scale information. **B**) NTv3 is pre-trained through a two-stage curriculum with mixed-length batches: a main phase of 10.5T tokens on 1–8 kb windows, followed by length extension to 1 Mb through successive 50B-token stages that increase context length by factors of two, while maintaining a mixture of shorter windows throughout to mitigate short-context forgetting. **C**) Relative sampling proportions of OpenGenome2 sub-datasets shift over the curriculum, with shorter-sequence sources emphasized at early stages and whole-genome sources increasingly represented as the maximum context length grows. **D**) NTv3 performance across different sequence lengths on the eukaryotic genic-window sub-dataset. Line plot shows masked-token accuracy at each sequence length, evaluated on a fixed set of held-out tokens at the end of every pre-training stage. **E**) Per-step training FLOPs as a function of sequence length for the NTv3 model suite. **F**) Pre-training loss as a function of cumulative FLOPs for models of different parameter counts, illustrating smooth scaling behavior. **G**) Pre-training loss as a function of total tokens processed, demonstrating stable trillion-token optimization across model sizes. **H**) Single-GPU inference time across sequence lengths for NTv3 and representative long-context foundation models, showing order-of-magnitude faster processing at megabase context for NTv3 at comparable or larger scales.

inference for full megabase windows on a single A100/H100 GPU. Training such models at 1 bp resolution, however, requires substantial parallelization, for which we employ sequence parallelism together with Fully Sharded Data Parallel (FSDP). In mixed precision, NTv3 8M can run inference for sequences of several tens of kilobases on a laptop CPU within seconds, making this class of models broadly accessible. Notably, NTv3 achieves faster inference than all existing representation-learning models we compared against—including full-Transformer architectures [21] and state-space models such as HyenaDNA [22] and Caduceus [26]—and the largest NTv3 model is in fact faster than competing models with up to  $60\times$  fewer parameters (Fig. 2H). Overall, these design choices highlight the efficiency benefits of hierarchical long-context modeling and provide a scalable, easy-to-use family of models for diverse genomic applications.

## 2.2. Post-training NTv3 on multispecies functional data to build a unifying genomics foundational model

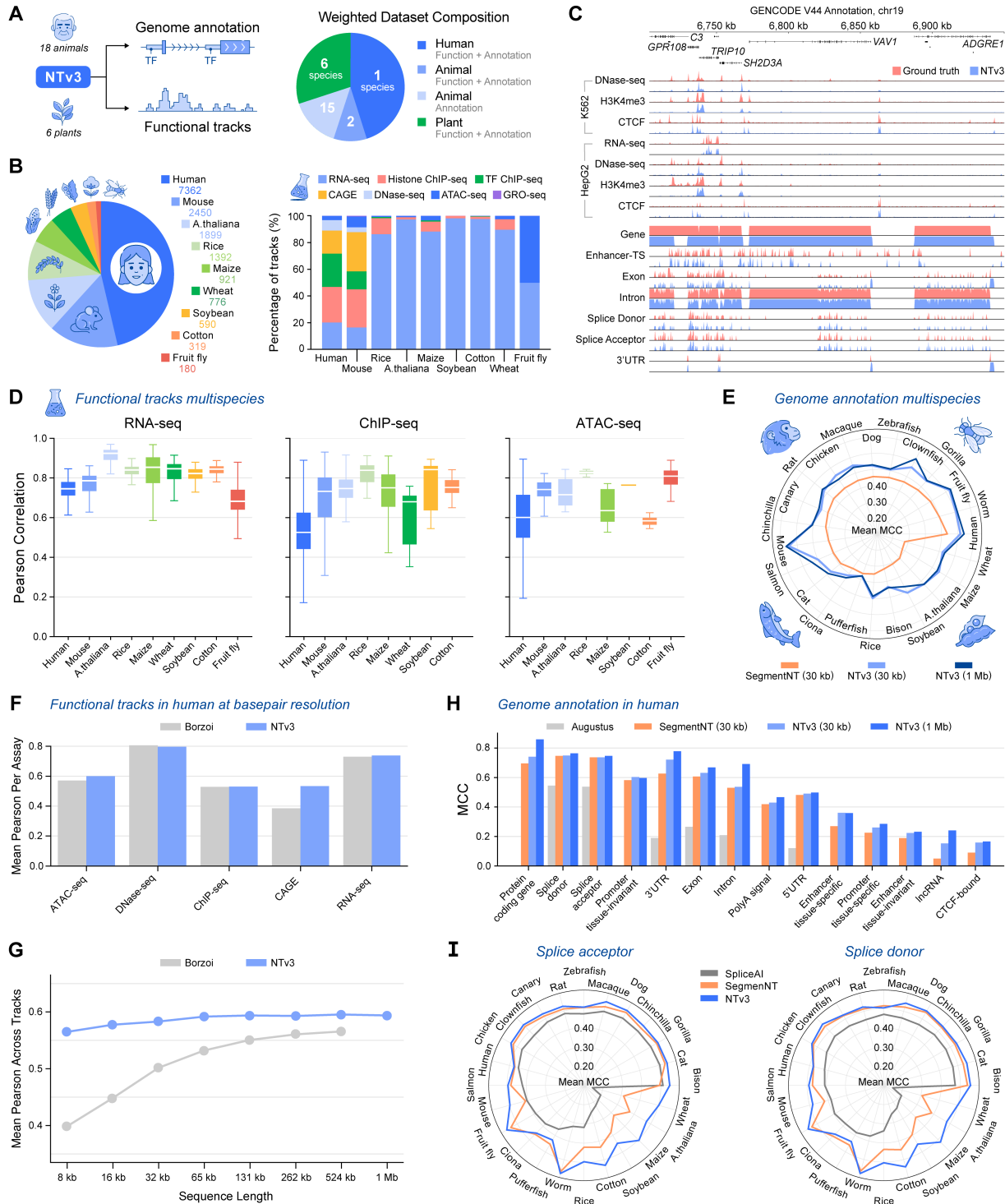
To move beyond learning generic statistical structure from genome sequences and toward modeling how the genome is used across in tissues and cells, we introduce a second training stage (referred to as *post-training*) that extends NTv3 into a multimodal genome-to-function model. In this stage, the model is jointly optimized on one-megabase, base-resolution functional tracks and genome annotations across animals and plants (Fig. 3A; see Methods 4.3). Alongside these supervised objectives, we continue applying the MLM loss to ensure that the language-modeling head remains synchronized with the shared representation space and remains usable after post-training.

This post-training corpus spans 24 species, illustrating the scale and breadth of functional signals incorporated into the model. Whereas previous large-scale sequence-to-function models have been limited primarily to human and mouse, NTv3 is the first model of this scale to integrate functional and annotation data from fruit fly and, critically, to support functional-track prediction for agricultural applications across six key plant organisms.

Post-training integrates supervised objectives over genome tracks spanning 23 modalities of two main types: quantitative functional tracks derived from sequencing assays and binary annotation denoting the location of diverse genomic elements. We use a poisson-multinomial for the former and a per base-pair focal cross-entropy loss [33] for the latter. NTv3 is trained on 15,889 functional tracks for human, mouse, fruit fly, and six plant species, covering chromatin state (DNase, ATAC-seq, histone modifications, transcription factor binding) and transcriptional activity (RNA-seq, CAGE) across diverse tissues, cell types, and experimental contexts (Fig. 3C; Supplementary Table C.1, Supplementary Table C.2). In parallel, we train on genome annotation signals that include core gene-structure elements (protein-coding genes, long non-coding RNAs, 5' UTR, 3' UTR, exons, introns, open reading frames, splice donor and acceptor sites, and start and stop codons) as well as regulatory elements (poly(A) signals, tissue-invariant and tissue-specific promoters and enhancers, and CTCF-bound sites) for the same nine species represented in the functional-track corpus, augmented with an additional 15 animal species for annotations (Fig. 3A, Supplementary Fig. A.7). This design yields both breadth across taxa and depth within species, allowing NTv3 to learn shared regulatory principles while remaining sensitive to lineage-specific genome organization.

To instantiate the post-trained models, we selected the 100M and 650M pre-trained checkpoints as initialization and optimized them on functional-track and genome-annotation objectives, yielding the main NTv3 models: *NTv3 100M (post)* and *NTv3 650M (post)* (Supplementary Fig. A.5A). The 8M model, while included in pre-training, lacked sufficient capacity for stable multispecies post-training—showing degraded performance on several functional tracks—so this stage was restricted to the larger variants. Post-training initializes from the pre-trained backbone and its trained language-modeling head while adding prediction heads that are randomly initialized: species-specific functional-track heads that output the set of tracks available for each species, and a single genome-annotation head shared across all species. We further introduce species conditioning via adaptive layer normalization to enable stable multi-species inference within a single set of weights. For computational efficiency, post-training is performed in two sub-phases: a main phase of 1.2T tokens using sequences up to 131 kb, which does not require sequence parallelism and therefore supports highly efficient data-parallel training; and a second phase of 54B tokens in which we length-extend to 1 Mb contexts enabled by sequence parallelism. Throughout post-training, we use mixed-length batches to promote accuracy across sequence contexts and apply dataset weighting to balance supervision across a highly heterogeneous multispecies corpus (Fig. 3A). Together, the combination of continued MLM with supervised track and annotation prediction produces a unified model that predicts regulatory activity and genome structure at base-pair resolution across phylogenetically distant species.

We observed that genome pre-training provides a clear benefit for supervised sequence-to-function learning during post-training (Supplementary Fig. A.5B). When post-training is initialized from a pre-trained NTv3 650M (pre) checkpoint rather than random weights, we observe substantially faster convergence across both annotation tasks and functional-track prediction. In terms of asymptotic performance, the effect is more nuanced: pre-training yields a large and consistent gain for genome-annotation tasks, whereas for functional tracks the improvement is more modest. This pattern suggests that large-scale genome pre-training is particularly effective at revealing structural and regulatory sequence patterns, while cellular- and tissue-level functional readouts may depend more heavily on modality-specific signals introduced during supervised adaptation.



**Figure 3 | Post-training NTv3 on multispecies functional tracks and genome annotations.** **A)** Multispecies post-training setup. A single NTv3 backbone is post-trained to predict both genome annotations and functional tracks, with dataset weighting across species groups (pie charts) reflecting the relative contribution of human, other track-bearing species, and species with annotation-only supervision. **B)** Number of supervised functional tracks per species (left) and assay-type breakdown across species (right). **C)** Genome-browser example predictions. Representative held-out human interval showing close agreement between experimental coverage profiles and NTv3 650M (post) predictions for multiple assays in K562 (top) and genome-annotation element classes (bottom). **D)** Box plots showing test-set Pearson correlation (log-transformed targets) across nine species, summarized per assay family (RNA-seq, ChIP-seq, and ATAC-seq). **E)** Radar plot displaying mean MCC per species across annotation categories, comparing SegmentNT at 30 kb context to NTv3 at 30 kb and 1 Mb. **F)** Mean Pearson correlation per assay for Borzoi and NTv3 650M (post) evaluated at single-nucleotide resolution on 524 kb sequences (Borzoi original sequence length). **G)** Sequence-length robustness of NTv3 650M (post). Mean Pearson correlation for Borzoi and NTv3 650M (post) as a function of input length evaluated at single-nucleotide. **H)** Bar plot showing MCC on the human SegmentNT benchmark across annotation classes, comparing Augustus and SegmentNT evaluated at 30 kb and NTv3 650M (post) at 30 kb and 1 Mb. **I)** Radar plot displaying mean MCC per species for splice acceptor (left) and donor (right) sites, comparing SpliceAI and NTv3 at 30 kb context.

NTv3 650M (post) demonstrates accurate base-resolution predictions for both quantitative functional assays and discrete genome annotations across species. Genome-browser examples illustrate that NTv3 reproduces fine-grained read-coverage structure with high fidelity on held-out genomic intervals, with predicted profiles closely matching experimental signal shapes and amplitudes across multiple assays (Fig. 3B on human and Supplementary Fig. A.6 on Arabidopsis). In these unseen regions, the model accurately captures both sharp, localized peaks and broader domains of activity, while simultaneously producing coherent annotation masks aligned with gene structure, including exon–intron segmentation and UTR boundaries. This qualitative concordance is reflected in strong quantitative performance across assays and species: NTv3 650M (post) achieves consistently high Pearson correlation for RNA-seq, ChIP-seq, and ATAC-seq tracks, with performance varying by modality and organism but remaining robust across animals and multiple plant species (Fig. 3D). Finally, for genome annotation, NTv3 650M (post) achieves high Matthews correlation coefficient (MCC) across a broad phylogenetic range of 24 species, and further benefits from long-context inference, with the 1 Mb setting yielding gains over 30 kb in human and other selected species (Fig. 3E).

### 2.3. NTv3 achieves state-of-the-art functional track and annotation prediction across species

We compared NTv3 650M (post) with Borzoi [7], the leading sequence-to-function model for genome-wide functional track prediction, under two complementary evaluation settings in human and mouse. We first evaluated both models under Borzoi’s original training regime, fine-tuning NTv3 650M (post) on Borzoi’s human and mouse datasets using the native 32 bp output resolution and target scaling. Under these conditions, NTv3 650M (post) consistently outperforms Borzoi across CAGE, ChIP-seq, ATAC-seq and DNase-seq type experiments, with relative improvements up to 3% depending on the assay (Supplementary Fig. A.10). For RNA-seq we observed a minor reduction in performance up to -0.5%, which we expect to be due to the multi-tasking compromise between different types of tasks.

We next compared the two models under NTv3 650M (post)’s native single–nucleotide-resolution objective. For this base-resolution evaluation, we restricted analysis to tracks shared between the two training corpora and converted Borzoi’s 32 bp outputs to 1 bp resolution by repeating predictions across bins to align with unscaled, nucleotide-level targets. Under this setting, NTv3 650M (post) matches Borzoi on DNase-seq and ChIP-seq prediction while substantially improving accuracy on ATAC-seq, CAGE, and RNA-seq (Fig. 3F). Notably, NTv3 650M (post) maintains stable performance across a wide range of input lengths, including contexts as short as 8 kb—consistent with the mixed-length post-training regime—whereas Borzoi exhibits marked degradation below its training context length of 524 kb. Moreover, NTv3 650M (post) continues to benefit from increasing context length, with the largest gains observed for gene-expression–related modalities, consistent with the influence of distal regulatory elements on transcriptional output (Fig. 3H, Supplementary Fig. A.9). Together, these results show that NTv3 650M (post) captures local regulatory determinants at base resolution while more effectively exploiting long-range genomic context for modalities governed by distal regulation.

We additionally compared our NTv3 650M (post) model with ChromBPNet [34], a base-resolution model for DNase-seq and ATAC-seq profile prediction. On the two DNase-seq experiments included in both training datasets, ChromBPNet reaches Pearson Correlation Coefficient (PCC) values of 0.704 on HepG2 and 0.717 on IMR-90, whereas NTv3 650M (post) achieves 0.753 and 0.755, respectively—substantially outperforming the

ChromBPNet model.

On genome annotation, NTv3 650M (post) achieves state-of-the-art accuracy across a broad phylogenetic range, outperforming AUGUSTUS [35, 36] on common elements between both models, as well as SegmentNT [37] under its native training regime with 30 kb context, further improving when scaling inference to 1 Mb (Fig. 3E,H). These gains are consistent across annotation types and species, indicating that the post-trained model captures both local sequence determinants of gene structure and longer-range contextual cues that disambiguate challenging genomic elements. In human, improvements are most pronounced for categories that are difficult to infer from short context alone, including enhancers, promoters, lncRNAs, and CTCF-bound sites (Fig. 3H). NTv3 also outperforms the specialized splicing model SpliceAI [12] on donor and acceptor sites across species (Fig. 3I). Crucially, the advantage of NTv3 650M (post) extends beyond closely related mammalian genomes to more distant taxa, including multiple plant species, demonstrating that the combination of large-scale genome pre-training and multispecies post-training yields representations of genome organization that generalize across divergent gene architectures and regulatory landscapes. In practice, NTv3 650M (post) complements classical and task-specific annotation approaches—such as *ab initio* gene finders (e.g., AUGUSTUS) and splicing-focused predictors (e.g., SpliceAI)—by providing a single unified model that produces genome-wide, base-resolution annotation masks across species without relying on hand-engineered features, species-matched RNA-seq and homologous protein evidence as other annotations pipelines, or species-specific retraining.

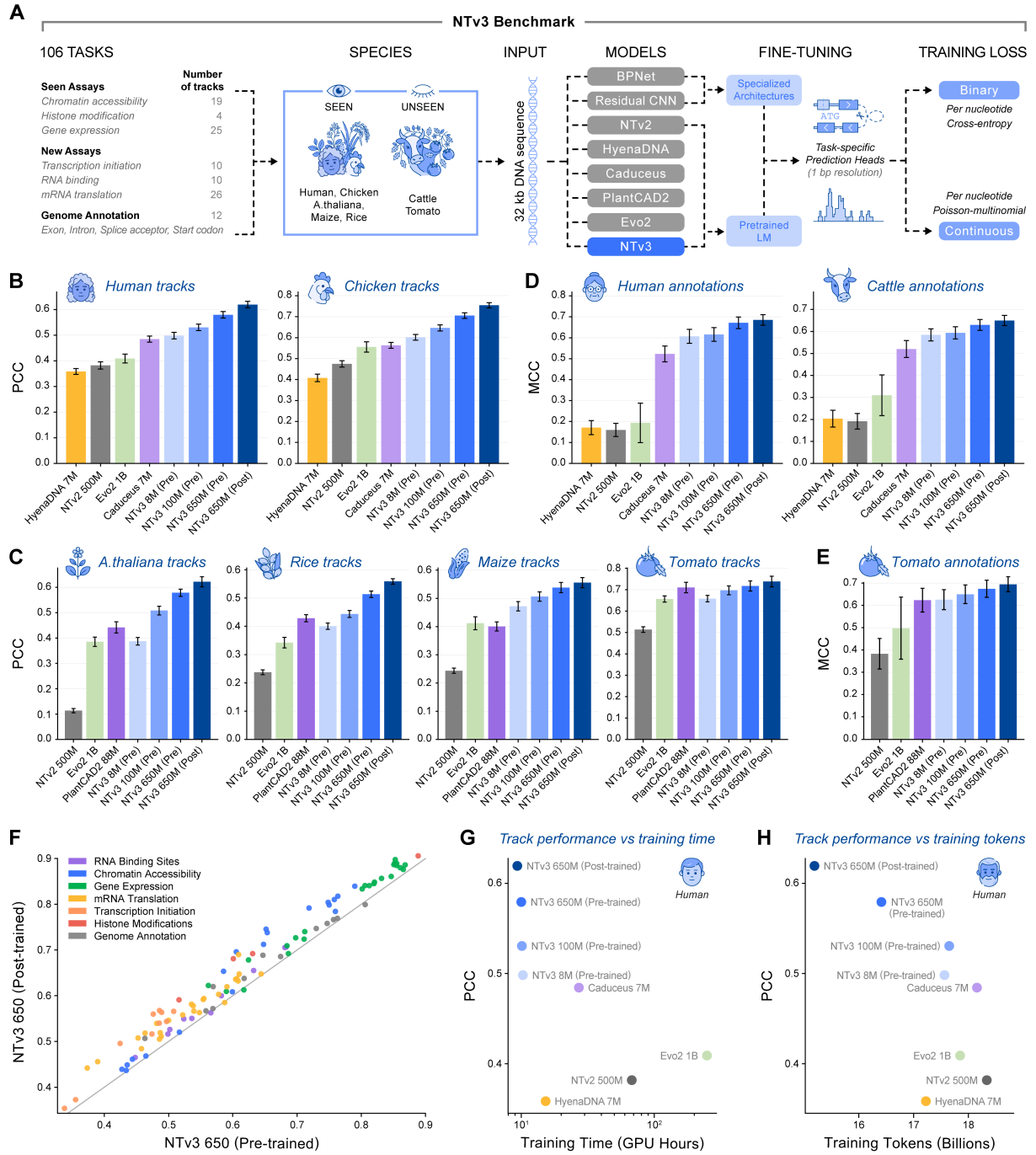
Across both functional track prediction and genome annotation, we observe systematic benefits from longer context and continued length-extension training. For both the 100M and 650M variants, performance improves as the input window increases, and extending training beyond 131 kb yields additional gains at long contexts, with the largest improvements observed at the megabase scale (Fig. 3G, Supplementary Fig. A.8, Supplementary Fig. A.9). This behavior is notable because hierarchical encoder-decoder architectures such as U-Nets are often reported to generalize poorly outside the sequence lengths encountered during training. Our results indicate that the explicit mixture of sequence lengths during both pre-training and post-training mitigates this limitation, producing stable predictions across sequence scales and avoiding the sharp performance degradation seen in competitive models when evaluated away from their training context (including Borzoi [7], AlphaGenome [8] and SegmentNT [37]). Taken together, these improvements position NTv3 as a unified foundation model that delivers highly accurate base-resolution prediction of both functional genomic signals and genome annotations across a broad phylogenetic range, including both animals and plants.

#### 2.4. The NTv3 BENCHMARK: A comprehensive suite for long-range genomic evaluation

To systematically assess downstream utility of NTv3, we introduce the NTv3 BENCHMARK, a comprehensive suite of 106 long-range genomic tasks designed to evaluate genomics models under realistic 32 kb-input, single-base-pair-output settings (Fig. 4A; Table 8 and Supplementary Table C.3). In contrast to existing evaluations that are typically restricted to a single organism or assay family, NTv3 BENCHMARK covers both functional-regulatory prediction and genome-annotation prediction across a phylogenetically diverse panel of species. To ensure a clean separation from post-training data, we curate downstream tasks to avoid data leakage: the benchmark includes new tracks and assays for species present during post-training (human, chicken, arabidopsis, rice and maize) as well as tracks from animal and plant species entirely unseen during post-training (cattle and tomato). Together, these design choices yield a controlled and leakage-resistant benchmark that simultaneously tests long-range sequence-to-function reasoning, generalization across taxa, and transfer to heterogeneous regulatory modalities, including transcription initiation (PRO-cap), RNA binding sites (eCLIP), and mRNA translation (Ribo-seq), which are not represented in prior multispecies post-training corpora.

We designed NTv3 BENCHMARK to isolate differences in representation quality under a standardized fine-tuning protocol. We selected a 32 kb input length as a practical compromise that provides substantial regulatory context while remaining computationally tractable for large-scale ablations and baseline comparisons, and we fine-tune all models to produce single-nucleotide outputs using task-specific prediction heads. The benchmark includes two complementary task families. For genome-annotation prediction, we fine-tune separate single-task models for four universally defined labels—exon, intron, splice\_acceptor, and start\_codon—evaluated by mean MCC. For functional-regulatory prediction, we use a multi-task setup in which all assays available for a given species are grouped into a single multi-output head and jointly fine-tuned, evaluated by mean Pearson correlation across tracks.





**Figure 4 | NTv3 BENCHMARK: a multispecies, long-range fine-tuning benchmark for genomic foundation models.** A) Benchmark overview and evaluation protocol. NTv3 BENCHMARK comprises two task families—genome-annotation prediction and functional-regulatory track prediction—evaluated under a unified 32 kb input / 1 bp output setting. Tasks span species seen during NTv3 post-training (human, chicken, *Arabidopsis*, maize, rice) and species unseen during post-training (cattle, tomato) to probe generalization across taxa. All models are fine-tuned with task-specific prediction heads. B,C) Fine-tuning results on (B) human, chicken, (C) Arabidopsis, rice, maize and tomato functional tracks. Bar plots showing the mean test-set Pearson correlation across all tracks for each species. Error bars denote the mean standard error across three independent fine-tuning runs per model per task. D,E) Genome-annotation fine-tuning results. Bar plots showing the mean test-set MCC on (E) human, cattle and (F) tomato annotation tasks. Error bars denote the mean standard error across three independent fine-tuning runs per model per task. F) Scatter plot showing test-set performance (Pearson correlation or MCC, depending on task) of NTv3 650M (pre) and NTv3 650M (post) models across all 106 NTv3 BENCHMARK task. Points are colored by assay category, and the diagonal indicates parity. G,H) Efficiency of NTv3 models relative to competing genomic foundation models. Functional-track performance (mean Pearson correlation on human tasks) plotted against (G) total training time (GPU hours) and (H) total number of training tokens.

We evaluated both NTv3 pre-trained and post-trained checkpoints to isolate the contributions of genome-scale self-supervision and multispecies post-training, and we include multiple model sizes to characterize scaling behavior. We compare against a broad set of representative baselines spanning both from-scratch convolutional architectures and pretrained genomic foundation models (Table 9). The from-scratch baselines include the bpNet architecture [6, 34] and a large residual convolutional architecture inspired on SpliceAI [12], which represent established supervised backbones for profile prediction and nucleotide-level annotation. The pre-trained models cover the dominant inductive biases explored in genomic foundation modeling and different model sizes, including Transformer-based masked models (NTv2 500M [21]), state-space models (Caduceus 7M for human [26] and PlantCAD2 88M for plants [38]), long-convolution operators (HyenaDNA 7M, the largest one available [22]), and large auto-regressive hybrid models trained on multi-kingdom corpora (Evo2 1B [28]). To ensure comparability, we fine-tune all models using a standardized supervised training regime with matched optimization budgets, identical data pre-processing, consistent evaluation metrics, and uniform checkpoint selection criteria (see Methods B.6), thereby attributing performance differences primarily to representational quality rather than task-specific tuning. We performed three independent fine-tuning runs per model per task to ensure robust results with statistical significance. We provide NTv3 BENCHMARK datasets ([https://huggingface.co/datasets/InstaDeepAI/NTv3\\_benchmark\\_dataset](https://huggingface.co/datasets/InstaDeepAI/NTv3_benchmark_dataset)) and have developed an interactive leaderboard containing results for all models across each task to facilitate comparisons ([https://huggingface.co/spaces/InstaDeepAI/ntv3\\_benchmark](https://huggingface.co/spaces/InstaDeepAI/ntv3_benchmark)).

## 2.5. NTv3 can be efficiently fine-tuned towards various downstream applications

We first evaluated NTv3 on the functional regulatory tasks of the NTv3 BENCHMARK, measuring performance via the Pearson correlation coefficient (PCC) of single-nucleotide predictions against ground-truth signal. Across all evaluated species we observed clear scaling laws across the NTv3 650M (pre) family: as model size increased from 8M to 100M and finally to 650M, performance steadily improved across all species and assays (Fig. 4B,C; see detailed results on human tracks at Supplementary Fig. A.11). NTv3 models consistently outperformed existing baselines. Notably, even the pre-trained-only NTv3 650M (pre) checkpoints surpassed competitive baselines. For instance, our smallest model, NTv3 8M (pre) (8M parameters), achieved higher correlation than the larger 500M-parameter NTv2 and 1B-parameter Evo2 model. Among the foundation model baselines, Caduceus (7M) emerged as the strongest competitor in humans, and PlantCAD2 (88M) in plants; these models likely benefit from their Mamba architecture for single-bp prediction, whereas Evo2 may be hindered by the constraints of causal masking in this setting. The post-trained NTv3 model provided a substantial additional boost in performance, achieving the highest scores across every single species evaluated (Fig. 4B,C). Crucially, this improvement extended to assays (PRO-cap, eCLIP, Ribo-seq) and species (tomato) unseen during the post-training phase, indicating that the model learned a generalized regulatory grammar rather than simply memorizing species-specific features. Additionally, we benchmarked NTv3 against state-of-the-art convolutional architectures trained from scratch, demonstrating that our NTv3 8M (pre) reaches the performance of the largest Residual CNN (44M) supervised baseline despite being approximately  $5\times$  smaller (Table 9).

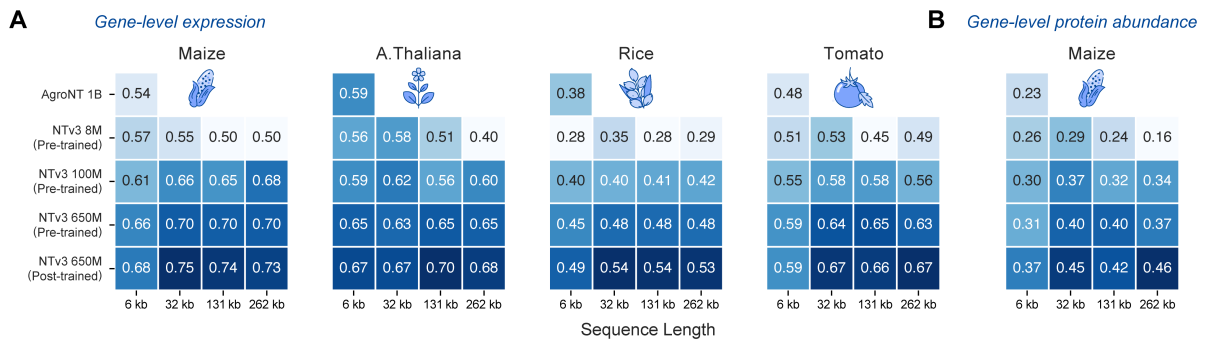
The same results were observed on the genome-annotation tasks. On human annotation tasks, NTv3 650M (pre) models reached the best performance among supervised or foundation model baselines, with NTv3 650M (post) further improving it to an MCC of approximately 0.7 (Fig. 4D). Extending the evaluation to species not present in post-training, cattle and tomato, NTv3 models exhibited equally strong scaling behavior, with the post-trained 650M model maintaining a clear lead over the pre-trained variants and baselines like PlantCAD2 (Fig. 4E). The consistent hierarchy of performance—where  $8\text{M} < 100\text{M} < 650\text{M} < 650\text{M Pos}$ —demonstrates that model capacity and alignment via post-training both contribute for resolving the precise boundaries of genomic elements.

To further verify the necessity of the foundation model approach, we compared our NTv3 pre-trained checkpoints against identical architectures initialized with random weights and fine-tuned for the same duration. Across the human regulatory and genome annotation tasks, models initialized from NTv3 consistently and significantly exceeded the performance of randomly initialized counterparts (Supplementary Fig. A.12). This gap confirms that the high performance of NTv3 is driven by the representations learned from the 9 trillion nucleotides of the OpenGenome2 corpus and subsequent post-training on functional data, rather than solely by the U-Net architecture or the fine-tuning optimization scheme.

We further characterized the benefits of multispecies post-training and the efficiency of the NTv3 family. Direct

per-task comparisons show that post-training consistently improves performance over the pre-trained backbone across all regulatory modalities (Fig. 4F). These improvements indicate that post-training aligns long-range representations with fine-grained regulatory signals rather than merely refining already strong predictions. These gains were also clear in a computational context, revealing that NTv3 achieves superior functional-track performance at substantially lower training cost than competing foundation models (Fig. 4G,H). In particular, NTv3 models occupy a favorable Pareto frontier, delivering higher accuracy per GPU hour and per training token than Transformer-, state-space-, and long-convolution-based baselines. Together, these results demonstrate that multispecies post-training yields systematic accuracy improvements while preserving—and in practice enhancing—the efficiency and scalability of the NTv3 architecture.

## 2.6. Evaluation on gene-level tasks in Agronomics



**Figure 5 | Gene-level evaluation of NTv3 in Agronomics tasks. (A)** Gene expression (4 species) and **(B)** protein abundance (1 specie) prediction from promoter-proximal sequence. Heatmap shows the mean  $R^2$  across tissues as a function of input sequence length for AgroNT (1B) and NTv3 pre-trained model variants (8M, 106M, 650M) evaluated at matching context windows; NTv3 650M (post) denotes the post-trained 650M model.

Moving beyond nucleotide-level tasks, we evaluated NTv3 on gene-level prediction tasks across Agronomics datasets, assessing its ability to map promoter-proximal sequences to quantitative molecular traits (Supplementary Table C.4). Adopting the evaluation framework from the AgroNT study [39], we tested the model’s capacity to predict both gene expression and protein abundance across tissues in four plant species. These tasks provide a stringent test of whether the model captures the regulatory logic governing both transcriptional output and the subsequent protein levels.

NTv3 demonstrated remarkable efficiency, outperforming the domain-specialized, 1-billion parameter AgroNT baseline despite using orders of magnitude fewer parameters (Fig. 5A,B). For example, at the 6 kb context length used by AgroNT [39], our 8M-parameter model achieved a mean  $R^2$  of 0.57 for gene expression in Maize, surpassing the 0.54 achieved by the 1B AgroNT model. This efficiency advantage extended to protein abundance, where the 8M model ( $R^2 = 0.26$ ) similarly outperformed the 1B baseline ( $R^2 = 0.23$ ) at the same sequence length. Our largest 650M pre-trained model, although still smaller than AgroNT, further improved performance on both tasks ( $R^2 = 0.66$  for gene and 0.37 for protein). Similar results were observed for the other species.

Crucially, the architecture of NTv3 enabled us to leverage extended genomic contexts that are computationally intractable for standard Transformers. Extending the input length yielded substantial performance gains. For the 650M pre-trained model, increasing the context from 6 kb to 32 kb drove a significant improvement in Maize gene expression prediction, raising the mean  $R^2$  from 0.66 to 0.70.

Post-training provided the final tier of performance improvement, consistently achieving the highest predictive accuracy across all tasks and lengths. NTv3 650M (post) reached a peak  $R^2$  of 0.75 for Maize gene expression at 32 kb. For protein abundance, the model continued to benefit from extremely long contexts, achieving its best score ( $R^2 = 0.46$ ) at 262 kb. These results underscore the capabilities of NTv3 and how it can capture long-range regulatory information within distal promoter regions that is missed by models restricted to shorter contexts.

## 2.7. NTv3 intrinsically captures multi-scale biological features in genomic sequences

Next, we investigated how NTv3’s unified modeling approach can be used to study genome regulation and the functional impact of genetic variants. As NTv3 progresses from pre-training to post-training, its objective evolves from reconstructing masked nucleotides in genomic sequences to internalizing the regulatory principles that connect DNA sequence to transcriptional output. These principles reflect the inherently multi-scale structure of the *cis*-regulatory code [40]: at the local level, transcription factor binding sites (TFBSs) encode nucleotide-resolved grammar and cooperative interactions [41–43], while at larger scales, these elements are guided by chromatin topology to enable long-range enhancer–promoter communication [44–46]. If NTv3 has learned this hierarchy, then its predictions should manifest not only through strong quantitative predictive performance but as consistent mechanistic hypotheses recoverable from its internal representations.

We hypothesized that NTv3 does indeed internalize this multi-scale regulatory logic, and that its full suite of interpretable modalities—signal profiles, functional annotations, gradient-based attribution maps, and attention-derived interaction maps (see Methods 4.10.1)—could be analyzed collectively to reconstruct the underlying *cis*-regulatory mechanisms (Fig. 6A). Together, these modalities enabled us to validate the model’s ability to resolve complex *cis*-regulatory interactions, such as enhancer–promoter contacts as well as the precise functional impact of single-nucleotide variants (SNVs).

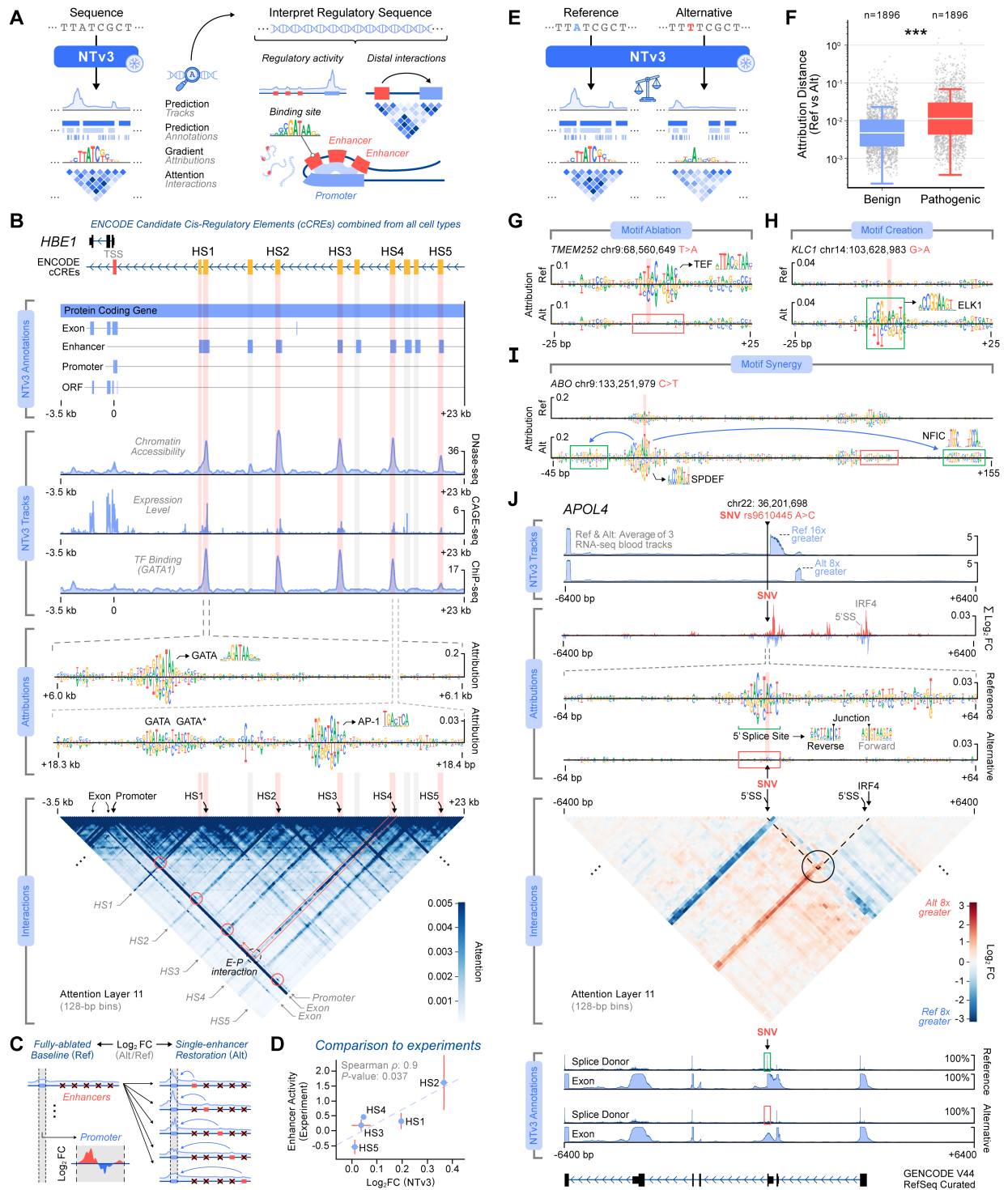
### 2.7.1. Decoding the mechanistic basis of long-range enhancer–promoter interactions

To assess whether NTv3 reconstructs long-range enhancer–promoter interactions, we analyzed the human HBE1 locus—an erythroid regulatory region containing five DNase hypersensitive elements (HS1–HS5) with experimentally characterized enhancer activities [47, 48] (see Methods 4.10.2). This locus provides a controlled test case to evaluate whether NTv3’s predicted modalities converge on the experimentally validated enhancer hierarchy (Fig. 6A).

Using the full 131,072 bp sequence centered on the HBE1 locus, NTv3 produced a consistent multimodal regulatory profile (Fig. 6B). The model’s cell-type-agnostic outputs aligned closely with curated reference databases: predicted intron and exon annotations matched GENCODE transcripts, predicted promoter and enhancer elements coincided with ENCODE candidate cCREs, and the final-layer attention map showed off-diagonal structure linking the promoter token range to the annotated enhancer regions including HS1–HS5, consistent with long-range enhancer–promoter coupling. NTv3’s cell-type-specific functional track predictions further localized regulatory activity to the expected regions: predicted K562 DNase, GATA1 ChIP-seq, and CAGE signals each peaked over the HS1–HS5 enhancer elements, with CAGE signal sharply concentrated at the annotated HBE1 promoter (see Methods 4.10.2). Gradient-based attribution maps computed with respect to the K562 CAGE track highlighted base-pair-level importance across the HS elements, forming motifs corresponding to GATA and its reverse complement.

Together, these multimodal features establish a coherent baseline regulatory architecture against which perturbations can be compared. To quantify enhancer sufficiency in a manner analogous to *in vitro* CRISPRi experiments, we performed targeted *in silico* perturbations using a fully repressed background (Fig. 6C; see Methods 4.10.2). All five hypersensitive elements (HS1–HS5) were simultaneously ablated by dinucleotide-preserving shuffling, and individual enhancers were then restored one at a time, yielding a controlled setting in which each enhancer’s ability to drive transcription could be assessed in isolation. Enhancer sufficiency was quantified by computing the  $\log_2$  fold-change in predicted K562 CAGE signal between each single-enhancer restoration and the fully-ablated baseline within a promoter-proximal window (see Methods 4.10.2). This analysis recovered a clear hierarchy of enhancer strength (HS2 > HS1 > HS4 > HS3 > HS5; see Methods 4.10.2) that was strongly correlated with experimentally measured activities at the HBE1 locus (Spearman  $\rho = 0.900$ ,  $p = 0.037$ ; Fig. 6D), indicating that NTv3 accurately captures the relative regulatory potency of individual enhancer elements using sequence information alone.

Collectively, these results show that NTv3 internalizes the multi-scale *cis*-regulatory architecture of the HBE1 locus, recovering the experimentally validated enhancer hierarchy and demonstrating that the model learns a biologically meaningful representation of long-range enhancer–promoter logic.



**Figure 6 | NTv3 deciphers multi-scale regulatory architecture and variant mechanisms.** **A)** Schematic of the multimodal interpretation framework applied to enhancer–promoter interactions. **B)** Validation at the *HBE1* locus (–3.5 kb to +23 kb from TSS). Top: GENCODE and ENCODE annotations align with predicted outputs. Middle: Attribution maps (computed with respect to K562 CAGE) identify GATA and AP1 motifs (JASPAR: MA0036.2, MA0099.2). Bottom: Attention map highlights promoter contacts with enhancers HS1–HS5 (circled). **C, D)** *In silico* perturbation strategy (**C**) and resulting quantification of predicted CAGE signal (**D**). **E)** Differential framework for assessing the mechanistic impact of SNVs. **F)** Attribution differences between pathogenic and benign GTEx eQTLs (Wilcoxon two-sided test,  $P < 0.001$ ). **G–I)** Attribution maps illustrating distinct variant mechanisms: motif ablation (**G**; TEF, MA0843.2), motif creation (**H**; ELK1, MA0028.2), and distributed regulatory rewiring (**I**; SPDEF, MA0686.2; NFIC, MAI527.1). **J)** Multimodal reconstruction of the pathogenic *APOL4* splice variant rs9610445. The model captures disruptions in splice-donor attribution (canonical U2-dependent 5' site; reverse complement), long-range attention, and genomic annotations consistent with curated transcripts (GENCODE).



### 2.7.2. Decoding the mechanistic basis of functional variant effects on quantitative traits

Having shown that NTv3 reconstructs long-range enhancer–promoter architectures and aligns its annotations and attention maps with known regulatory interactions, we next asked whether this multi-scale understanding carries down to the level of single nucleotide variation. Interpreting non-coding variants requires precisely this form of semantic continuity: pathogenic SNVs typically exert subtle, context-dependent effects that only emerge when regulatory logic is modeled holistically.

Recent work has shown that attribution-based analyses of deep sequence models can surface the motif-level perturbations—such as the disruption or creation of TFBSs—that drive the model’s predicted expression changes [8, 49–51]. Motivated by this link, we developed a differential framework (Fig. 6E) that contrasts the model’s interpretable modalities between reference and alternative alleles (see Methods 4.10.2). Using pathogenic and benign GTEx (v.8) eQTLs from Human Whole Blood [7, 52], we found that gradient-based attribution differences were significantly larger for pathogenic variants (Wilcoxon  $p < 0.001$ ; Fig. 6F), indicating that NTv3’s internal representations reliably register functional *cis*-regulatory perturbations. By contrasting reference–alternative attribution maps, we found that NTv3 resolves many pathogenic SNVs into clear mechanistic classes of regulatory disruption (Fig. 6G,H, and I): (i) motif ablation, where a variant destroys an existing TFBS; (ii) motif creation, where a variant introduces new regulatory grammar; and (iii) motif synergy, where a variant triggers distributed, nonlocal rewiring across broader genomic regions. For detailed biological interpretation of representative variants—including mechanisms implicated in hematopoiesis, cancer suppression, and COVID-19 severity—see Supplementary Note B.7.3.

The motif-synergy class underscores the need to analyze NTv3’s full suite of interpretable modalities jointly as anticipated in our overarching hypothesis. A representative example is the pathogenic variant rs9610445 within the *APOL4* locus. As *APOL4* is a recently evolved, primate-specific gene [53], experimental dissection is challenging, highlighting the utility of foundation-model reasoning at such loci. NTv3 overcomes this by reconstructing a multilayered mechanism that unifies splice-motif loss, long-range architectural reweighting, and cross-track consistency into a single regulatory explanation (Fig. 6J). This SNV ranks among the strongest eQTLs by gradient divergence, and NTv3’s attribution maps show that the alternative allele sharply attenuates the splice-donor motif at the variant site, driving an approximately 16-fold reduction in predicted RNA-seq signal. NTv3’s annotation tracks reinforce this interpretation by removing the Splice Donor label and repositioning adjacent exonic boundaries—consistent with its HIGH-impact classification across 15 *APOL4* transcripts [54]. In parallel, NTv3 identifies coordinated distal remodeling: its attention maps register the strongest  $\log_2$ -fold change nearly 32 kb away, aligning with a secondary splice-donor motif and a nearby IRF4 site emphasized by attribution.

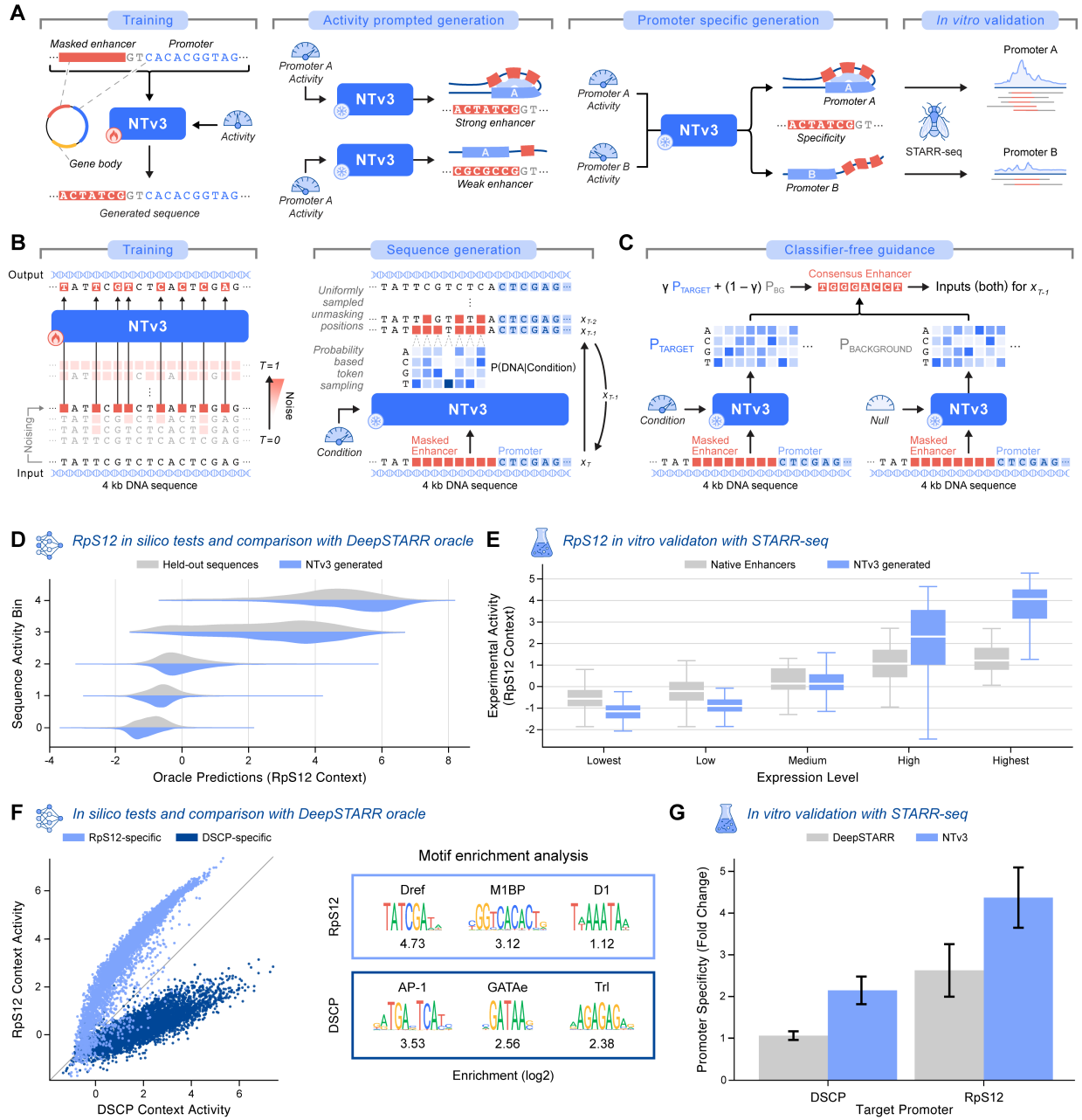
Through this integrated multimodal approach, these results demonstrate that NTv3 captures the multi-scale regulatory principles underlying variant effects, enabling mechanistic reconstruction of pathogenic disruptions from sequence perturbation alone.

### 2.8. NTv3 enables conditional diffusion-based sequence generation

We extend NTv3 beyond prediction by re-purposing its LM head for sequence generation using a discrete diffusion framework. Specifically, we adopt Masked Diffusion Language Modeling (MDLM) [55], which generalizes standard MLM [32]: rather than masking a fixed fraction of tokens per sequence, MDLM samples the number of masked positions according to a diffusion schedule. This approach allows direct reuse of NTv3 pre-trained and post-trained weights for sequence generation without introducing additional parameters or modifying the architecture, relying solely on the existing language-model head (Fig. 7A).

Diffusion provides a natural framework for DNA sequence generation. Unlike autoregressive approaches that impose a left-to-right generation order through causal masking, such as Evo [23, 28], masked diffusion operates bidirectionally and preserves sequence context on both sides of a generated element. This enables in-context generation and infilling, such as designing an enhancer within a fixed plasmid or genomic background (Fig. 7B). In addition, recent work has shown that discrete diffusion models can achieve substantially higher inference efficiency than autoregressive generation, requiring far fewer forward passes per generation [56].

To concretely illustrate these generative capabilities in a biologically meaningful setting, we focused on the design of transcriptional enhancers. Enhancers provide an ideal testbed for generative regulatory modeling: their activity is continuous, highly sequence-dependent, and modulated by the surrounding regulatory context,



**Figure 7 | NTv3 enables controllable enhancer design with experimental validation.** **A)** Starting from NTv3 650M (pre), we adapt the model for conditional enhancer generation, enabling both activity-prompted design and promoter-specific design within a unified pipeline, and validate generated sequences using *in cellulo* STARR-seq. **B)** Masked Diffusion Language Modeling (MDLM) allows generative post-training from NTv3 650M (pre) without altering the native tokenization scheme. During generation, NTv3 conditions on the target activity level as well as the surrounding genomic context, enabling sequence design respecting both functional and sequence contexts. **C)** Classifier-free guidance provides directional control by encouraging the model to match the target condition distribution while avoiding background profiles, producing clearer separation between target and non-target properties and improves the fidelity of generation. **D)** *In silico* activity conditioning evaluated with DeepSTARR[5]. Oracle-predicted activities of generated enhancers for the RPS12 promoter recapitulate the intended activity bins. **E)** *In cellulo* STARR-seq measurements validate that generated enhancers express at the intended activity levels for the RPS12 promoter. NTv3 designs reproduce the expected stratification across activity bins when compared with native enhancers and also extend into a broader activity range, supporting the design of particularly strong regulatory elements. **F)** *In silico* promoter specificity and sequence determinants. Oracle predictions for sequences generated by NTv3 to target the RpS12 or DSCP promoter, together with motif enrichment analysis for each group of sequences. **G)** *In cellulo* STARR-seq validation confirms promoter selective activity. Specificity measured as fold-change between target and background promoter activities. NTv3-generated enhancers show stronger promoter specificity than enhancers selected using the DeepSTARR oracle alone.

including promoter identity. We therefore evaluate NTv3 in the context of enhancer design using a dataset of genome-wide enhancer activity in *Drosophila* S2 cells using STARR-seq [5, 57]. This dataset reports enhancer activity for two distinct promoters—the developmental DSCP promoter and the housekeeping RpS12 promoter—enabling controlled assessment of both activity-level specification and promoter-dependent regulatory compatibility within a unified experimental framework.

Starting from the NTv3 post-trained weights, we fine-tune the model on the enhancer activity dataset using MDLM, where training sequences comprise the full STARR-seq reporter plasmid including the target promoter, and conditioning is provided by the measured enhancer activity value (see Methods 4.11). Conditioning signals are injected into NTv3 through adaptive layer normalization, extending the conditioning mechanisms introduced during multispecies post-training to downstream design tasks. This allows functional attributes and sequence context to be flexibly combined during generation without introducing auxiliary models or classifiers. At generation time, we apply classifier-free guidance (CFG) [58, 59], enabling targeted control over functional properties without auxiliary classifiers (Fig. 7C). We evaluate this framework on two enhancer design tasks described below and experimentally validate 1,000 generated sequences using *in cellulo* STARR-seq.

### 2.8.1. Activity-level specific enhancer generation

We first assess whether NTv3 can generate novel enhancer sequences with specified activity levels for a given promoter. During generation, the full reporter construct containing the promoter is provided as input, and the enhancer region is masked. This setup ensures that enhancer design is explicitly conditioned on promoter context, allowing the model to account for regulatory dependencies between the enhancer and its target promoter. Activity levels are discretized into five bins and represented by dedicated conditioning tokens, which are injected into the model through adaptive layer norm. During sampling, guidance contrasts activity-conditioned and unconditioned predictions under the same promoter context, biasing generation toward sequences that satisfy the desired activity level.

Enhancers are generated for each activity bin under each promoter context. We evaluated the generated sequences *in silico* using the DeepSTARR oracle [5], comparing predicted activities to held-out native enhancers whose experimentally measured activities fall within the same bins. Across all bins, the predicted activity distributions of generated sequences are well-aligned with those of native enhancers, indicating that the model produces designs occupying the correct functional range (Fig. 7D, Supplementary Fig. A.13A). Generated sequences also recapitulate native GC-content and *k*-mer composition distributions, and BLAST searches reveal no evidence of training data memorization.

We have evaluated 600 generated enhancers *in cellulo* using STARR-seq [57]. Across the quintile activity bins, the experimental measurements show clear stratification consistent with the conditioning, validating that NTv3 can reliably design enhancer elements with desired activity levels (Fig. 7E, Supplementary Fig. A.13B). Together, these results demonstrate precise and robust activity-guided enhancer generation by NTv3.

### 2.8.2. Promoter-specific enhancer design

We next tested whether NTv3 can generate enhancers that are selectively active to a specific promoter. In this experiment, sequences are conditioned to have high activity for a target promoter while suppressing activity under an alternative promoter. This is achieved through a tailored classifier-free guidance scheme: the conditioned input specifies high activity together with the target promoter context, while the background input retains the same activity conditioning but replaces the promoter with the alternative promoter. During sampling, CFG contrasts the denoising trajectories of these two inputs, amplifying sequence features favored by the target promoter while suppressing features associated with the competing promoter. This guidance scheme enables directed generation of enhancer sequences that operate preferentially within a desired promoter context.

Using this framework, we generate enhancers specific to the DSCP or RpS12 promoters. To evaluate the promoter specificity of generated enhancers, DeepSTARR-predicted activities under both promoters are plotted against one another for all generated sequences (Fig. 7F). The resulting distributions show clear divergence, with one branch enriched for sequences active under DSCP and the other under RpS12, indicating that NTv3 generates promoter-selective enhancer sets rather than broadly active elements. Motif enrichment analysis further supports this observation: the two generated libraries exhibit distinct motif enrichment patterns that

align with previously described promoter-specific enhancer architectures [60] (Fig. 7F).

For comparison between our framework and oracle-guided strategies, we have also generated a large pool of random sequences using the same computational budget and select candidates based on DeepSTARR predictions. NTv3-generated (150) and DeepSTARR-selected (150) sequences were evaluated using STARR-seq in both promoter contexts. Promoter specificity is defined as the fold-change between activity in the target promoter context and activity in the alternative promoter context. NTv3-generated sequences exhibited stronger promoter specificity, with an average specificity of 2.14-fold for DSCP and 4.37-fold for RpS12. Even when given the same computational budget, oracle-based filtering produces weaker specificity than direct generative conditioning with NTv3 (Fig. 7G).

Together, these results demonstrate that NTv3 enables precise and flexible sequence design at a fixed computational budget. Beyond achieving target activity levels and promoter specificity, the model generates diverse sequence solutions that satisfy the same functional constraints, exploring multiple regions of regulatory sequence space. This ability to produce families of functionally equivalent yet sequence-diverse sequences enables robust downstream design by offering multiple sequence-diverse solutions for a fixed regulatory objective.

### 3. Discussion

A recurring theme in modern genomics is that the strongest models tend to specialize: supervised *sequence-to-function* predictors achieve excellent accuracy on curated assay panels, self-supervised *language models* excel at capturing conserved elements from unlabeled genomes, and *generative models* enable principled sequence design guided by functional objectives. NTv3 unifies these paradigms into a single backbone by combining base-resolution MLM with a multispecies post-training stage that jointly optimizes functional-track prediction and genome annotation across animals and plants, while maintaining a consistent interface for downstream fine-tuning, interpretation and generation. In practice, this means the same representation can support (i) genome modeling, (ii) high-resolution functional readout prediction, (iii) segmentation-style genome annotation, and (iv) a pathway to controllable generation—alleviating the need to maintain separate architectures and training approaches for each “mode” of genomics modeling. This unification is especially valuable for long-range regulation, where the modeling target (variant effect, track signal, annotation boundaries, or designed sequence) changes across applications, but the underlying regulatory grammar and multi-scale dependencies are shared.

Across our evaluations, NTv3 provides strong transfer under supervised fine-tuning already from genome-only pre-training, and multispecies post-training yields a further, consistent boost by aligning these representations with functional and annotation supervision. NTv3 supports both short- and long-range prediction, with outputs that can be configured at nucleotide resolution (e.g., regulatory and annotation tracks) or aggregated to gene-level expression or protein readouts. A central design goal of NTv3 is practical usability: fine-tuning and downstream adaptation require minimal task-specific engineering and modest computational resources, in contrast to other large state-space models like PlantCAD2 [38] or Evo2 [28] that are substantially more difficult to adapt in practice. As a result, NTv3 matches or outperforms those models across multiple tasks while using orders of magnitude fewer parameters and/or significantly less compute. To facilitate broad use, we release the full suite of base-pretrained and post-trained checkpoints across model sizes, together with example notebooks and practical cookbooks covering fine-tuning and evaluation, model interpretation, and controllable sequence design.

A key design decision in our new Nucleotide Transformer version was to move to a U-Net–style architecture. This multi-resolution architecture—now emerging as a common pattern in long-context genomics [7, 8]—retains the parallelism and expressivity of Transformer components while avoiding the computational cost of applying full attention at megabase length. We demonstrate that this architecture can be successfully pre-trained at scale on genomic sequence as foundation models, enabling long-range representation learning while preserving base-resolution outputs. Reaching this regime depended on coordinated advances in training strategy and infrastructure: we used a staged sequence-length curriculum with mixed-length training to preserve accuracy across scales, employed multi-GPU sequence parallelism for the longest windows beyond the data-parallel regime used at shorter lengths, and optimized data serving to sustain accelerator utilization over trillions of tokens. Post-training further required curating a new multispecies supervision corpus with genome annotations for 24 species and ~16k functional tracks across animals and plants, and training a species-conditioned model.

Together, these components provide a practical recipe for scalable multimodal training at megabase context.

We evaluated NTv3 on established benchmarks for track prediction and genome annotation, comparing to sequence-to-function models such as Borzoi [7] as well as annotation-focused baselines including SegmentNT [37], SpliceAI [12], and classical gene-finding pipelines such as Augustus [35, 36]. On functional tracks, NTv3 achieves consistent improvements across major assay families, allowing now to predict experimental readouts from 1Mb context at single-nucleotide resolution. The gains are more pronounced for genome annotation, where long-context, base-resolution modeling and multispecies post-training translate into systematic improvements, particularly for labels requiring precise boundary localization and integration of regulatory context. Notably, these improvements extend robustly to plant genomes, where complex gene structures, larger intergenic regions, and sparser functional annotation have historically limited model performance. While the improvements in human and mouse build upon an already mature ecosystem of high-performing models, the relative impact in the agronomics space is substantially larger: prior approaches were trained on datasets with limited species coverage or assay diversity [38, 61, 62], and provided only restricted support for long-context modeling (typically < 8 kb; [38]), base-resolution outputs, or systematic generalization across diverse plant genomes. By jointly training on animal and plant genomes, NTv3 substantially improves annotation accuracy in agricultural species, highlighting the value of multispecies foundation models for crop genomics and applied breeding contexts.

To further evaluate popular genomics models in real-world genomics tasks, we introduced NTv3 BENCHMARK. Existing benchmarks often focus on either short input windows (< 1,000bp), a single task family, or in-distribution evaluation, which simplifies comparison but limits assessment of generalization across assays, species, and output granularities under practical constraints [21, 29–31]. Moreover, these benchmarks typically rely on heavily processed or abstracted prediction targets; while valuable for controlled evaluation, such targets often obscure the structure, variability, and noise inherent to the underlying biological measurements. In contrast, practical genomics applications require modeling raw or minimally processed signals at base resolution across heterogeneous assays and taxa. NTv3 BENCHMARK is therefore designed to (i) standardize input length and output resolution in a realistic regime (32 kb inputs with base-resolution prediction heads), (ii) span complementary task families (functional-track prediction and genome annotation) and (iii) explicitly test generalization under phylogenetic and assay shift, including unseen taxa and track families relative to model pre-training and post-training. Across this benchmark, NTv3 consistently outperforms representative pretrained baselines and from-scratch convolutional models, with clear gains from model scaling and a further improvement of post-trained checkpoints over genome-only pre-training. Strikingly, the 8M-parameter NTv3 model matches or exceeds, on most tasks, substantially larger or widely used long-context baselines including the 1B-parameter Evo2 [28] and the Caduceus/PlantCAD2 [26, 38] models, while being significantly more efficient.

A major advantage of a unified backbone is interpretability that is coherent across scales: the same model can be interrogated for local motif logic, mid-range regulatory syntax or long-range interactions between genomic elements such as enhancers and promoters. Practically, this allows attributing predictions from multiple heads (functional tracks, and annotations) to the same underlying sequence features and comparing the resulting explanations for consistency—e.g., whether putative motif grammar suggested by an accessibility head is mirrored in promoter / enhancer annotation boundaries. This “multi-view” interpretability is particularly useful in long-range settings, where mechanistic hypotheses often involve both sequence features (motifs) and the spatial organization of regulatory elements, and where single-output models can overfit explanations to one modality. Applying this approach to NTv3 allowed us to decipher multi-scale regulatory architecture and variant mechanisms, recovering unbiasedly many cases that have been reported in the literature.

Finally, NTv3 extends this unified long-context backbone into controllable sequence design via discrete diffusion generation, providing a direct bridge from genome understanding to genome engineering within the same modeling framework. The same long-context representations used for prediction and annotation are fine-tuned for conditional generation, supporting both activity-targeted and promoter-specific enhancer design without auxiliary classifiers or separate generative models. Importantly, this workflow is validated not only with *in silico* oracles but also experimentally, grounding the generative capability in measurable regulatory outcomes. For a fixed target activity and promoter context, NTv3 generates diverse families of sequence-distinct enhancers that satisfy the same functional constraints, enabling robust exploration of regulatory design space rather than reliance on a single solution. The combination of megabase-capable representations, supervised sequence-to-function heads, and experimentally validated conditional generation positions NTv3 as a general framework



for design problems where long-range context and promoter specificity are central, and suggests a path toward iterative, model-in-the-loop regulatory engineering that remains consistent with the predictive interfaces used for variant interpretation and functional annotation.

We anticipate that NTv3 will serve as a scalable substrate for the next generation of genome foundational models—enabling increasingly comprehensive, multi-species, long-range models as functional supervision, perturbation datasets, and generative objectives continue to mature.

### 3.1. Limitations, caveats and future opportunities

Although NTv3 is designed for megabase context, maintaining uniformly strong performance across the full length spectrum remains non-trivial, as we show for models like Borzoi [7]. Extending the maximum receptive field can shift capacity toward long-range dependencies and, without explicit mitigation, degrade short-context accuracy; we address this with mixed-length training during both pre-training and post-training, but residual length sensitivity remains an open area. Further progress will likely require more principled length-invariant curricula and regularizers that explicitly preserve short-range representations while scaling context.

Current supervision remains imperfect for learning the full causal hierarchy from distal sequence to gene-level outcomes. While NTv3 benefits from megabase context, accurately capturing the influence of very distal regulatory elements ( $> 100$  kb) remains challenging because standard functional tracks provide limited causal linkage and only indirect access to 3D genome organization. Moreover, predicting genome-wide tracks is often easier than predicting gene-level responses across diverse assays and contexts, suggesting a need for more gene-centric and multi-modal objectives that better reflect integrated regulatory programs. Finally, we have not yet benchmarked NTv3 for personal-genome prediction, where these models are usually limited [63, 64].

Our multispecies post-training should be viewed as a starting point that can be expanded substantially. Incorporating additional species and richer modalities—particularly single-cell assays, perturbation datasets, and multiome measurements—would provide stronger constraints on context-specific regulatory mechanisms and reduce ambiguities introduced by batch effects across independently generated tracks. Extending supervision to population-scale and personalized functional data would further improve robustness to within-species genetic diversity and broaden translational utility in both medical and agricultural settings.

MDLM provides a principled route to convert a masked LM into a controllable generative model, but it can also be viewed as a natural generalization of MLM. This suggests a future direction in which diffusion-style masked modeling is used consistently across pre-training and post-training, unifying representation learning and generation under a single objective family while preserving base-pair tokenization and long-context scalability. Such a formulation may also offer a convenient interface for richer conditioning (e.g., species, cell type, assay, target activity), strengthening the bridge between genome understanding and targeted sequence design.

# Methods

4.1 NTv3 Model	22
4.2 Pre-training on Genomes	27
4.3 Post-training on Tracks	29
4.4 Performance Infrastructure and Optimization	32
4.5 Pre-training dataset	34
4.6 Post-training Dataset	37
4.7 Evaluation on Sequence-to-Function Benchmarks	41
4.8 NTv3 BENCHMARK	42
4.9 Evaluation on Gene-level Downstream Tasks	47
4.10 Model Interpretation Analysis	47
4.11 Fine-tuning NTv3 for Conditioned Sequence Generation	52

## 4. Methods

### 4.1 NTv3 Model

NTv3 is a large-scale genomics foundation model designed to process DNA sequences of up to 1 million base pairs and to jointly predict heterogeneous outputs at single-nucleotide level, including 1) language-model probabilities, 2) functional regulatory tracks, and 3) genome annotation tracks. The model was trained in two main phases: pre-training on genomes from  $\sim 128,000$  species (referred to as NTv3 (pre) models) followed by post-training on genomes and functional data from 25 animal and plant species (referred to as the main NTv3 (post) models). During pre-training the model is trained through masked language modeling (MLM) and only predicts language-model probabilities for masked tokens. In the post-training phase we introduce additional heads to predict functional and genome annotation tracks, in addition to the MLM objective. During post-training, we also introduce a species-conditioning mechanism, allowing the model to adapt its regulatory and annotation predictions to the species associated with each input sequence.

#### 4.1.1. Model Architecture

NTv3 follows a U-Net [65] encoder-decoder architecture tailored for long sequences, similar to recent models such as Borzoi [7] and AlphaGenome [8]. The model consists of six main components: sequence encoder, central transformer tower, sequence decoder, species conditioning and output heads. We trained NTv3 models of different sizes by scaling the parameters of the different components: NTv3 with 650 million (M) trainable parameters, as well as smaller versions with 8M (NTv3 8M) and 100M (NTv3 100M) parameters. The models were implemented using JAX [66] and Flax NNX [67]. Below we describe the model architecture’s components along with relevant pseudocode.

**Sequence Embedder.** An embedding layer first transforms DNA sequences into sequences of tokens with continuous embeddings. These embeddings are then processed by a stem layer consisting of a 1D convolution with kernel size 15 and stride of 1 followed by a GELU [68] activation. The stem projects the token embeddings into a higher-dimensional embedding that captures local nucleotide context, while preserving sequence length dimension.

---

```

1
2 def sequence_embedder(x: Array) -> Array:
3     # Embedding lookup: alphabet_size -> token_embed_dim
4     x = Embed(num_tokens=alphabet_size, embed_dim=token_embed_dim)(x)
5     # Stem: 1D convolution with kernel 15, stride 1
6     # Projects token_embed_dim -> conv_init_embed_dim
7     x = Conv1D(num_channels=conv_init_embed_dim, width=15)(x)
8     x = GeLU(x)
9     return x
10

```

---

**Sequence Encoder.** The sequence encoder progressively down-samples the input DNA sequence from 1 bp resolution to 128 bp resolution embeddings over 7 convolutional blocks. Each convolutional block contains: (1) a main convolution (kernel size 5) preceded by Layer Normalization [69, 70] and followed by GELU activation, (2) a residual convolution [71] (kernel size 1) with a skip connection, and (3) an average pooling layer (window size 2, stride 2) that halves the sequence length. The number of feature channels is maintained consistently from the stem through all down-sampling layers to maintain high representational capacity. We observed empirically that performance improves when having the highest number of feature channels already at the first convolutional block, rather than increasing the number of channels sequentially over the encoder blocks.

Intermediate representations from each of the 7 stages, corresponding to resolutions of 1 bp, 2 bp, 4 bp, ..., 64 bp (before the average-pooling step), are stored for use as U-net skip connections in the sequence decoder. The final encoder output has a resolution of 128 bp, sequence-axis length 1 Mb/128 bp = 8192, and  $768 + 6 \times 128 = 1536$  feature channels.

---

```

1
2 def conv_block(x: Array, num_channels: int, width: int = 5) -> Array:

```

---

---

```

3     # Normalize channels, apply 1D convolution, then nonlinearity
4     x = LayerNorm(x)
5     x = Conv1D(num_channels=num_channels, width=width)(x)
6     x = GeLU(x)
7     return x
8
9 def residual_downsample_block(x: Array, num_channels: int) -> Array:
10    # Residual path: conv-block with kernel 5 plus conv-block with kernel 1
11    x = conv_block(x, num_channels=num_channels, width=5)
12    x = x + conv_block(x, num_channels=num_channels, width=1)
13    # Downsample sequence length by factor of 2
14    x = AvgPool(pool_size=2, stride=2)(x)
15    return x
16
17 def sequence_encoder(x: Array, embed_dim: int, num_downsamples: int) -> tuple[Array, list[Array]]:
18    residuals = [] # Store skip connections for each downsample stage
19    for _ in Range(num_downsamples):
20        residuals.append(x)
21        x = residual_downsample_block(x, num_channels=embed_dim)
22    # Returns encoded sequence and list of residuals
23    return x, residuals
24

```

---

**Transformer Tower:** Following the encoder, a transformer tower [72] processes the 128 bp resolution sequence embeddings to model long-range interactions across the full 1 mb input context. The transformer tower is composed of multiple multi-head attention layers with rotary positional embeddings (RoPE) [73]. In our implementation of RoPE, we use a base frequency of 10,000, where query and key vectors are split into two halves rather than using consecutive dimension pairs. No rescaling factor is applied. Each layer employs pre-layer normalization [70], multi-head self-attention, and a feedforward network with GLU (Gated Linear Unit) gating [74] and Swish activation [75], both wrapped in residual connections. Attention is computed at the block level, where sequences are divided into blocks of size  $2^{\text{num\_down-samples}}$ , allowing efficient long-range context modeling.

---

```

1
2 def apply_rope(heads: Array, base_freq: int = 10000) -> Array:
3     # Rotary positional embeddings: split heads into halves
4     seq_len, key_size = heads.shape[1], heads.shape[-1]
5     inv_freq = 1.0 / (base_freq ** (arange(0, key_size, 2) / key_size))
6     freqs = einsum('i,j->ij', arange(seq_len), inv_freq)
7     cos, sin = cos(freqs), sin(freqs)
8     x_first, x_second = heads[..., :key_size//2], heads[..., key_size//2:]
9     return concat(x_first * cos - x_second * sin,
10                  x_second * cos + x_first * sin, axis=-1)
11
12 def multi_head_attention(x: Array, attention_mask: Array) -> Array:
13     # Linear projections to queries, keys, values
14     q = Linear(num_heads * key_size)(x).reshape(..., num_heads, key_size)
15     k = Linear(num_heads * key_size)(x).reshape(..., num_heads, key_size)
16     v = Linear(num_heads * key_size)(x).reshape(..., num_heads, key_size)
17     q, k = apply_rope(q), apply_rope(k)
18     # Attention computation
19     attention_logits = einsum('bthd,bThd->bhtT', q, k) / sqrt(key_size)
20     attention_logits = where(attention_mask, attention_logits, -1e30)
21     attention_weights = softmax(attention_logits, axis=-1)
22     out = einsum('bhtT,bThd->bthd', attention_weights, v)
23     return Linear(embed_dim)(out.reshape(..., num_heads * key_size))
24
25 def ffn_block(x: Array, embed_dim: int, ffn_embed_dim: int) -> Array:
26     # Feedforward with GLU gating and Swish activation
27     x1, x2 = split(Linear(2 * ffn_embed_dim)(x), 2, axis=-1)
28     x = Swish(x1) * x2
29     return Linear(embed_dim)(x)
30
31 def transformer_block(x: Array, attention_mask: Array) -> Array:
32     # Pre-LayerNorm self-attention with residual
33     x = x + multi_head_attention(LayerNorm(x), attention_mask)
34     # Pre-LayerNorm feedforward with residual
35     x = x + ffn_block(LayerNorm(x), embed_dim, ffn_embed_dim)
36     return x
37
38 def transformer_tower(x: Array, attention_mask: Array, num_layers: int) -> Array:
39     # Stack multiple transformer blocks
40     for _ in Range(num_layers):
41         x = transformer_block(x, attention_mask)

```

---

---

```

42     return x
43

```

---

**Sequence Decoder.** The sequence decoder mirrors the encoder’s structure but operates in reverse: it progressively upsamples the 128 bp resolution embeddings (output from the transformer tower) back to the original 1 bp resolution, while maintaining the number of feature channels constant. This is achieved through 7 iterative de-convolutional blocks, mirroring the encoder’s architecture using transposed convolutions with kernel size 5, followed by layer normalization and GELU activation, and includes residual transposed convolutions (kernel size 1). As is standard in U-Net architectures [65], at each block the corresponding encoder residual is integrated via U-Net-style skip connections, allowing the decoder to recover fine-grained sequence information lost during down-sampling.

---

```

1
2 def deconv_block(x: Array, num_channels: int, width: int = 5, upsample: bool = True) -> Array:
3     # Normalize channels, apply transposed convolution (upsample if specified), then GeLU
4     x = LayerNorm(x)
5     x = TransposedConv1D(num_channels=num_channels, width=width, stride=2 if upsample else 1)(x)
6     x = GeLU(x)
7     return x
8
9 def residual_upsample_block(x: Array, num_channels: int) -> Array:
10    # Residual path: upsample with kernel 5 plus residual conv with kernel 1
11    x = deconv_block(x, num_channels=num_channels, width=5, upsample=True)
12    x = x + deconv_block(x, num_channels=num_channels, width=1, upsample=False)
13    return x
14
15 def sequence_decoder(x: Array, residuals: list[Array], num_downsamples: int) -> Array:
16    # Iteratively upsample and add skip connections from encoder
17    for residual in reversed(residuals):
18        x = residual_upsample_block(x, num_channels=embed_dim)
19        x = x + residual # U-Net skip connection
20    return x
21

```

---

**Output Heads.** The NTv3 architecture, after processing the input DNA through its encoder, transformer, and decoder stages, produces key internal representations at single nucleotide resolution. The final learned embeddings are projected to three types of output heads that make per-nucleotide predictions.

- **Language model head:** A linear layer that predicts a probability distribution over the DNA token vocabulary at each position.
- **Functional tracks prediction head:** A linear layer that predicts each of the functional tracks for the specified species. For example, when predicting on human 7632 tracks will be predicted for each nucleotide. The linear layer is followed by a softplus activation that ensures a positive prediction value. A layer normalization (separate to the annotation head) is applied to the embeddings before the tracks head. When using multiple species, to enable efficient batched computation across species with different track counts, the outputs are zero-padded to the maximum number of tracks across all species.
- **Genome annotation head:** A discrete classification head that predicts whether annotation labels are present for each nucleotide. This head has a dimension of 21 for the 21 annotation labels described in 4.6.2. The same head is used for all species. A layer normalization is applied to the embeddings before the head.

During the pre-training phase NTv3 uses solely the language model head. The follow-up post-training phase uses all 3 heads simultaneously.

---

```

1
2 def language_model_head(x: Array, embed_dim: int, alphabet_size: int) -> Array:
3     # Predict probability distribution over DNA tokens
4     x = GeLU(x)
5     x = Linear(alphabet_size)(x)
6     return x
7
8 def tracks_head(x: Array, embed_dim: int, num_tracks: int) -> Array:
9     # Predict functional genomic tracks (e.g., 7632 for human)

```

---



---

```

10  x = LayerNorm(x)
11  x = Linear(num_tracks)(x)
12  x = Softplus(x)
13  return x
14
15  def annotation_head(x: Array, embed_dim: int, num_labels: int = 21, num_classes: int = 2) -> Array:
16      # Predict genome annotation labels (21 labels, binary classification)
17      x = LayerNorm(x)
18      x = Linear(num_labels * num_classes)(x)
19      x = reshape(x, x.shape[:-1] + (num_labels, num_classes))
20      return x
21
22

```

---

**Species Conditioning.** The NTv3 architecture incorporates a species-conditioning mechanism that adapts predictions to a specific species using learned, species-specific embeddings. The input species is represented by a learned embedding vector that modulates network activations in two complementary ways:

*Adaptive Layer Normalization (AdaLN).* All layer normalizations in the convolutional, transformer, and deconvolutional blocks are replaced with adaptive layer normalization [76, 77]. The species embedding of dimension  $d_{\text{cond}}$  is mapped via a linear projection to produce shift ( $\beta$ ) and scale ( $\gamma$ ) parameters that modulate the normalized activations. Specifically, a linear layer projects the species embedding to a vector of dimension  $2 \times d_{\text{layer}}$ , which is then split into  $\beta, \gamma \in \mathbb{R}^{d_{\text{layer}}}$ :

$$\text{AdaLN}(x, s) = (1 + \gamma(s)) \odot \text{LayerNorm}(x) + \beta(s)$$

where  $s$  is the species embedding and  $[\beta(s), \gamma(s)] = Ws$  with  $W \in \mathbb{R}^{2d_{\text{layer}} \times d_{\text{cond}}}$ .

*Scaled Residual Connections.* Residual connections in convolutional blocks (encoder and decoder) and transformer feedforward networks are augmented with learned scaling factors [77]. The species embedding ( $d_{\text{cond}}$  dimensions) is projected via a linear layer to  $d_{\text{layer}}$  dimensions to produce a per-channel scaling vector  $\alpha \in \mathbb{R}^{d_{\text{layer}}}$  that modulates the residual branch:

$$x_{\text{out}} = x + (1 + \alpha(s)) \odot f(x)$$

where  $s$  is the species embedding,  $f(x)$  is the residual function (convolution or feedforward), and  $\alpha(s) = Ws$  with  $W \in \mathbb{R}^{d_{\text{layer}} \times d_{\text{cond}}}$ . The  $(1 + \alpha)$  formulation ensures that when  $\alpha = 0$  (at initialization), the scaling factor equals 1, preserving standard residual behavior.

We apply this species conditioning mechanism only at the post-training phase to be able to condition the prediction of functional tracks and genome annotation based on the species of input, enabling a single model to make predictions for multiple species. All modulation parameters (shift, scale, and residual scaling projections) are initialized to zero, ensuring that at initialization the conditioned model behaves identically to the pre-trained unconditioned model. This design enables seamless warm-starting from pre-trained weights during the post-training phase.

---

```

1
2  def adaptive_layer_norm(x: Array, conditions: list[Array], embed_dim: int) -> Array:
3      # Standard layer normalization
4      x = LayerNorm(x)
5      scale, shift = 1.0, 0.0
6      for condition in conditions:
7          # Project condition to shift and scale (initialized to zero)
8          modulation = Linear(2 * embed_dim, init=zeros)(condition)
9          shift_i, scale_i = split(modulation, 2, axis=-1)
10         # Accumulate scale multiplicatively and shift additively
11         scale = scale * (1.0 + scale_i[:, None, :])
12         shift = shift + shift_i[:, None, :]
13     return x * scale + shift
14
15  def conditioned_conv_block(x: Array, conditions: list[Array], num_channels: int, width: int = 5) -> Array:
16      # Convolution with adaptive layer norm and GeLU activation
17      x = adaptive_layer_norm(x, conditions, x.shape[-1])
18      x = Conv1D(num_channels=num_channels, width=width)(x)
19      x = GeLU(x)
20      return x
21

```

---

---

```

22 def conditioned_residual_block(x: Array, conditions: list[Array], num_channels: int) -> Array:
23     # Residual block with species-conditioned gating
24     gate = 1.0
25     for condition in conditions:
26         # Zero-initialized projection ensures gate=1 at init
27         gate_i = Linear(num_channels, init=zeros)(condition)[:, None, :]
28         gate = gate * (1.0 + gate_i)
29     # Apply gated residual to 1x1 convolution
30     return x + gate * conditioned_conv_block(x, conditions, num_channels, width=1)
31
32 def conditioned_downsample_block(x: Array, conditions: list[Array], num_channels: int) -> Array:
33     # Downsample with conditioned conv and gated residual, followed by average pooling
34     x = conditioned_conv_block(x, conditions, num_channels, width=5)
35     x = conditioned_residual_block(x, conditions, num_channels)
36     x = AvgPool(pool_size=2, stride=2)(x)
37     return x
38
39 def conditioned_transformer_block(x: Array, conditions: list[Array], attention_mask: Array) -> Array:
40     # Pre-LayerNorm self-attention with adaptive modulation
41     res = x
42     x = adaptive_layer_norm(x, conditions, embed_dim)
43     x = multi_head_attention(x, attention_mask)
44     x = res + x
45     # Feedforward with gated residual
46     gate = 1.0
47     for condition in conditions:
48         gate_i = Linear(embed_dim, init=zeros)(condition)[:, None, :]
49         gate = gate * (1.0 + gate_i)
50     ffn_out = adaptive_layer_norm(x, conditions, embed_dim)
51     ffn_out = ffn_block(ffn_out, embed_dim, ffn_embed_dim)
52     x = x + gate * ffn_out
53     return x
54
55 def species_conditioned_forward(tokens: Array, species_id: int) -> Array:
56     # Embed species as learned vector
57     species_embedding = Embed(num_species, cond_dim)(species_id)
58     conditions = [species_embedding]
59     # Embedding and stem (unconditioned)
60     x = sequence_embedder(tokens)
61     # Conditioned encoder
62     residuals = []
63     for _ in range(num_downsamples):
64         residuals.append(x)
65         x = conditioned_downsample_block(x, conditions, embed_dim)
66     # Conditioned transformer
67     for _ in range(num_layers):
68         x = conditioned_transformer_block(x, conditions, attention_mask)
69     # Conditioned decoder with U-Net skip connections
70     for residual in reversed(residuals):
71         x = conditioned_upsample_block(x, conditions, embed_dim)
72         x = x + residual
73     return x
74

```

---

#### 4.1.2. Model Scaling

We trained pre-trained models of three parameter sizes: 8M (NTv3 8M (pre)), 100M (NTv3 100M (pre)) and 650M (NTv3 650M (pre)). We performed a post-training phase for the base 100M (NTv3 100M (post)) and 650M (NTv3 650M (post)) models. All model variants can process sequences of any length up to 1 Mb. Scaling of model parameters from 8M to 650M followed the principles of depth, width, and attention scaling. The key parameters are shown in Table 1.

Parameter	8M Model	100M Model	650M Model	Scaling Factor (8M → 650M)
num_layers	2	6	12	6.0×
embed_dim	256	768	1536	6.0×
ffn_embed_dim	1024	3072	6144	6.0×
attention_heads	8	12	24	3.0×
key_size	32	64	64	2.0×

**Table 1** | Key parameters of NTv3 models and their scaling factors.

- **Depth Scaling (num\_layers):** The number of Transformer layers (num\_layers) was systematically increased by a factor of 6.0 $\times$ , scaling from 2 in the 8M model to 12 in the 650M model.
- **Width Scaling (embed\_dim):** The embedding dimension (embed\_dim) was expanded by 6.0 $\times$ , growing from 256 to 1536. Similarly, the dimension of the feed-forward network (ffn\_embed\_dim) maintained the standard ratio  $4 \times \text{embed\_dim}$ , scaling proportionally from 1024 to 6144.
- **Attention Scaling (attention\_heads, key\_size):** The multi-head attention mechanism was scaled along two axes. The number of attention heads (attention\_heads) was tripled (from 8 to 24), and the per-head key size (key\_size) was doubled from 32 to 64. The per-head key size remained fixed at 64 for all models exceeding 100M parameters.

## 4.2 Pre-training on Genomes

NTv3 (pre) was pre-trained on the OpenGenome2 dataset through self-supervised MLM objective. The pre-trained models have as single output head the language model head. We used the same approach as for the Nucleotide Transformer model [21], following the standard BERT masking strategy [32]: for each sequence, 15% of tokens were selected for potential modification; of this subset, 80% were replaced with a [MASK] token, 10% were substituted with a random token from the vocabulary, and the final 10% remained unchanged. For each batch, the loss function was computed as the sum of the cross-entropy losses between the predicted probabilities over tokens and the ground-truth tokens at each selected position.

Pre-training was conducted in two main phases with different sequence lengths for performance optimization: a 8kb-training phase with sequences up to 8,192 token context focused primarily on functional elements, that accounted for  $\sim 95\%$  of all tokens seen during pre-training; and a sequence length extension phase during which we extend the sequence context progressively up to 1M token context length, while continuing to interleave all previously seen sequence lengths, where we included entire genomes in the data distribution. This ensured that the model retained strong performance on shorter contexts. Exact sequence length proportions per phase are detailed in 2. We refer to the models after pre-training but before post-training as NTv3 (pre).

Stage	Sequence length											Tokens (B)
	$2^{10}$	$2^{11}$	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$2^{17}$	$2^{18}$	$2^{19}$	$2^{20}$	
8 kb Pre-training	0.079	0.130	0.132	0.660	0	0	0	0	0	0	0	10500
Length Extension 1	0.052	0.077	0.079	0.394	0.398	0	0	0	0	0	0	50
Length Extension 2	0.056	0.056	0.056	0.225	0.228	0.379	0	0	0	0	0	50
Length Extension 3	0.053	0.053	0.053	0.129	0.130	0.218	0.363	0	0	0	0	50
Length Extension 4	0.052	0.052	0.052	0.076	0.076	0.127	0.212	0.353	0	0	0	50
Length Extension 5	0	0	0.050	0.050	0.050	0.084	0.140	0.233	0.394	0	0	50
Length Extension 6	0	0	0.050	0.050	0.050	0.050	0.079	0.131	0.221	0.369	0	50
Length Extension 7	0	0	0.050	0.050	0.050	0.050	0.050	0.074	0.124	0.207	0.345	50

**Table 2** | Sequence length distribution at each stage of training.

Both pre-training phases used the same training recipe, including all optimizer settings, learning-rate schedule, masking strategy, and batch configuration, only differing in the sequence length schedule. Key hyperparameters are detailed in Table 3. We trained all models using the AdamW optimizer [78] with  $\beta_1 = 0.9$  and  $\beta_2 = 0.98$ . Similar to prior work [28, 79, 80], we found that reducing  $\beta_2$  from the canonical 0.999 was crucial for maintaining stability when training our largest models with the largest effective batch sizes.

In both stages, gradients were accumulated to reach an effective batch size of 8 million tokens. We observed that increasing the batch size further improved training efficiency but led to degraded model performance. The learning rate followed a modified polynomial decay schedule with a linear warm-up. It started at  $\eta_{\text{init}} = 1.0 \times 10^{-4}$ , increased linearly to a peak of  $\eta_{\text{peak}} = 4.0 \times 10^{-4}$  over the first 10.5 billion tokens, or approximately 1250 steps with an effective batch size of 8 million tokens. After reaching a peak the learning rate decayed to a

final value of  $\eta_{\text{final}} = 2.0 \times 10^{-4}$ . Formally, the schedule is defined at the step level as

$$\eta(t) = \begin{cases} \eta_{\text{init}} + \frac{\eta_{\text{peak}} - \eta_{\text{init}}}{t_w} \cdot t, & \text{if } t \leq t_w \\ \eta_{\text{peak}} \cdot \left(\frac{t_w}{t+1}\right)^\alpha, & \text{if } t > t_w \end{cases} \quad (1)$$

where  $t$  is the current step,  $t_w$  is the number of warmup steps, and the decay exponent  $\alpha = \ln(1/f)/\ln(T/t_w)$  is chosen such that the learning rate decays to  $\eta_{\text{final}} = f \cdot \eta_{\text{peak}}$  after  $T$  total steps. In our experiments, we set  $f = 0.5$ .

Parameter	Pre-training 1	Pre-training 2	Post-training 1	Post-training 2
Minimum sequence length	1 kb	1 kb	16 kb	16 kb
Maximum sequence length	8 kb	1Mb	131 kb	1 mb
MLM on genomes		✓		✓
Functional & annotation data		X		✓
Number of training tokens	10T	50B per phase	1.2T	50B
Tokens per Update		8M		1M
Tokens sampled per sequence length		4M		1M
Optimizer		AdamW		AdamW
Optimizer $\beta_1/\beta_2$		0.9 / 0.98		0.9 / 0.999
Weight Decay		0.1		0.1
Number Warmup Tokens	10.5B	0		3.4B
Initial Learning Rate ( $\eta_{\text{init}}$ )	$1.0 \times 10^{-4}$	-		$1.0 \times 10^{-5}$
Peak Learning Rate ( $\eta_{\text{peak}}$ )	$4.0 \times 10^{-4}$	$2.0 \times 10^{-4}$		$5.0 \times 10^{-5}$
Final Learning Rate ( $\eta_{\text{final}}$ )		$2.0 \times 10^{-4}$		$2.5 \times 10^{-5}$

**Table 3** | Training parameters for pre- and post-training phases. Pre-trained phase 1 corresponds to 8kb-training while phase 2 corresponds to sequence length extension. Post-training phase 1 corresponds to the 131 kb phase while phase 2 corresponds to length extension up to 1Mb.

To optimize data loading on our hardware, we structured the sampling process such that 4 million tokens of a single sequence length were yielded consecutively before moving to the next sequence length. Specifically, a sequence length was sampled according to the probabilities specified in Table 2, and mini-batches were yielded until a total of 4 million tokens ( $2^{20}$ ) was collected, with the additional constraint that each mini-batch contained a number of tokens that evenly divided 4 million for all sequence lengths. Sequence lengths were sampled in this manner until the accumulated total reached 8 million tokens, at which point a gradient step was taken. This design allowed each optimization step to include multiple sequence lengths, thereby increasing gradient diversity. Using fewer than 4 million tokens per sequence length could have allowed for an even greater diversity of sequence lengths per accumulation step; however, in later stages with longer sequences, this would have constrained data parallelization, as it would have limited the number of tokens that could be processed per step to be below 4 million.

#### 4.2.1. 8 kb Pre-training

The 8kb-training phase was run for 10.5 trillion (T) tokens on a filtered subset of OpenGenome2 that excludes whole eukaryotic genomes to focus on functional gene-rich regions and was comprised of approximately 1.8 trillion tokens in total. We train on a mixture of sequences of 1,024, 2,048, 4,096, and 8,192 bp to maintain performance across the different context lengths. For this initial pre-training phase, sequence lengths were sampled in proportion to the total number of tokens available at each length, with detailed distributions provided in Table 2. The duration of 8kb-training corresponded to approximately 5.8 epochs over the 8kb-training filtered dataset. A full description of the dataset composition, sampling approach, and tokenization is presented in Section 4.5.

Training exhibited stable dynamics throughout, without the need for gradient clipping or restarts (Fig. Figure 2G). Checkpoints were saved every 100 billion tokens, and the final 10.5T-token checkpoint was used to initialize the subsequent sequence-length extension phase. The loss continued to improve steadily through the end of this stage and training was stopped for time considerations.

#### 4.2.2. Sequence Length Extension

After completing 8kb-training, we implemented sequence length extension to adapt the model for processing of longer sequence contexts. Although our architecture supports long-context processing, we kept sequence length extension shorter than 8kb-training for compute efficiency, following prior work [28]. The efficiency of our architecture also enabled exploration of alternative sequence length schedules and significantly extended training at longer contexts. These additional experiments and ablations are presented in the supplementary section B.4.

We conducted the sequence length extension in seven stages, each consisting of  $\sim 50$  billion (B) tokens. The first stage introduced sequences of 16 kb. At each subsequent stage, the context length was doubled, culminating in sequences of 1 mb. Throughout sequence length extension, we continued to mix in all shorter sequence lengths, maintaining ability to process shorter contexts, and preventing performance degradation on the corresponding distributions of short sequences. At the start of each stage, 40% of sequences were drawn from the newly introduced length, while the remaining probability was distributed among shorter sequences. Additionally, we enforced a minimum of 5% probability per sequence length to ensure the shorted length continued to be seen in training, and re-normalized the weights accordingly. For stages containing sequences  $\geq 262$ kb we required sequence parallelism and as such excluded sequences with length  $\leq 2$ kb for efficiency. Details of the sequence lengths and data distributions across 8kb-training and sequence length extension phases are provided in Table 2. Training hyperparameters were identical to those used in 8kb-training and are detailed in Table 3.

### 4.3 Post-training on Tracks

After pre-training, we selected the 100M and 650M NTv3 pre-trained models and post-trained each on functional track prediction and genome annotation. We refer to the post-trained models as the main NTv3 models: NTv3 100M and NTv3. The main goal of post-training the models was to leverage the previous pre-training on over 128,000 species genomes and bring together functional experimental data and genome annotation. Post-training was performed on a newly curated, large dataset with genome annotations for 24 species and functional track data for 9 of those species (the data is described in subsection 4.6).

The post-trained model is initialized as the NTv3 650M (pre) with its trained language model head, where we add functional tracks and genome annotation heads that are randomly initialized. We add separate functional tracks heads for each species that predicts the number of tracks present for that species. A single genome annotation head is used across all species. In addition, we add the species-conditioning mechanism to enable the model to make predictions for multiple species.

Similar to pre-training, the post-training was performed in two sub-phases for computational efficiency: a first phase of 1.2T tokens where the model is trained on sequences up to 131 kb, where there is no need for sequence parallelism enabling highly efficient training with data parallelism; and a second phase of 54B tokens where we extend the maximum sequence length up to 1 mb. Below we detail the procedure of mixed-length training and dataset distribution for both phases.

#### 4.3.1. Mixed-Length Training

During post-training we used batches of sequences with different lengths to promote high performance across all sequence lengths. Sequence lengths are selected by starting with 16 kb and using successive multiples of 2 up to 1 mb. The sequence lengths are weighted with 10% weight each, with the remainder being given to the longest sequence length. For all sequence lengths used, the same source dataset (containing all species) is used. This ensures that the models sees every combination of sequence-length and species.

In practice, the first phase of post-training lasts for 1.2T tokens and uses four sequence lengths of 16, 32, 65 and 131 kb. Afterwards, the models are length-extended to 1 mb for 54B tokens. During this second phase, sequences of lengths 16 kb, 32 kb, 65 kb, 131 kb, 262 kb and 524 kb have a weight of 10% each, and the 1 mb a weight of 40%. Here sequence parallelism is used to enable training the models on sequences larger than 131 kb.

### 4.3.2. Dataset Weighting

During post-training different portions of the dataset are weighted to expose to the model to a diverse range of data while focusing on important areas. Table 4 shows the weighting used and number of species present for each group. We reasoned that since human has the most tracks we should give it the largest weight. This is followed by the other species (2 animals, and 6 plants) that also have functional tracks. Finally, the species with only annotation data are weighted at 1%. This dataset weighting strategy results in the frequency of each species being independent of the genome length, which is valuable when using highly imbalanced genomes.

Term	Number of species	Individual weight	Total weight
Human	1	45%	45%
Species with functional tracks	8	5 %	40%
Species with only annotations	15	1%	15%
<b>Total</b>	24	-	100%

Table 4 | Weighted composition of data groups used in post-training

### 4.3.3. Genomic Sequence Selection

The training sequences that are selected according to the dataset weights, are calculated by slicing the train set of the genome with a stride of 0.1% of the sequence length. For example, when using a sequence length of 131 kb, the train set would be sliced with a stride of 131bp. The results of this strategy ensure that each nucleotide is seen in many different contexts, which is expected to improve generalization. For validation and evaluation, a stride equal to the sequence length is used so that each nucleotide is seen exactly once by the model.

### 4.3.4. Functional Track Scaling

Due to the large range and skew distribution of functional track data, the values are scaled before calculating the loss. We follow AlphaGenome’s scaling approach of dividing the values by the non-zero mean of the entire track, applying a square-root soft clipping for values above 10 [8]. Additionally, for RNA-seq tracks, we apply a power transformation of  $x^{0.75}$  to the values. The targets are scaled before being used in loss computation and when making predictions, the head output is unscaled to raw values.

```

1 # Global constant: static per-track means and rna-seq mask
2 TRACK_MEANS = Array[...]
3 IS_RNA_SEQ = Array(..., dtype=bool)
4
5 def alphagenome_target_scaling(targets: Array) -> Array:
6     # Normalize by per-track mean
7     scaled = targets / TRACK_MEANS
8
9     # Power transform for RNA-seq tracks
10    scaled = where(IS_RNA_SEQ, scaled ** 0.75, scaled)
11
12    # Soft sqrt clipping for large values
13    scaled = where(
14        scaled > 10,
15        2.0 * sqrt(scaled * 10.0) - 10.0,
16        scaled,
17    )
18    return scaled
19
20 def alphagenome_logit_scaling(targets: Array) -> Array:
21    unclipped = where(x > 10.0, (x + 10.0) ** 2 / (4 * 10.0), x)
22
23    # Reverse power squashing for RNA-seq tracks
24    unsquashed = where(IS_RNA_SEQ, unclipped ** (1.0 / 0.75), unclipped)
25
26    # Reverse normalization by track mean
27    raw_values = unsquashed * TRACK_MEANS
28    return raw_values
29

```



#### 4.3.5. Post-training Objective and Loss Function

The post-training uses a three part loss function, with terms for the functional tracks, genome annotation, and language modeling. For the tracks, the loss is calculated on the center 37.5% of the sequence to ensure that the model has adequate context when making predictions

The functional tracks loss function consists of a Poisson term and multinomial negative loss likelihood term [6–8]. The Poisson loss is calculated on the sequence-wise sum of predicted coverage per track, thus training the model to predict the correct total coverage. The Poisson loss is calculated as

$$\mathcal{L}_{\text{Poisson}}(x, y) = \sum_i x - \sum_i y \log \sum_i x \quad (2)$$

where  $x$  are the labels,  $y$  are the predictions, and  $i$  is the sum over the sequence dimension. The multinomial term is calculated on the difference between the true and predicted distributions, thus training the model to predict the correct coverage shape.

$$\mathcal{L}_{\text{multinomial}} = - \sum_i y_i \log p_i, \quad \text{where } p_i = x_i / \sum_i x \quad (3)$$

The Poisson (or scaled loss) is down-weighted by 1/5th relative to the shape term to calculate the total loss.

---

```

1 def poisson_loss(predictions: Array, targets: Array) -> float:
2     return predictions - targets * log(predictions + epsilon)
3
4 def multinomial_loss(
5     logits: Array,
6     targets: Array,
7     shape_loss_coefficient: float = 0.2,
8 ) -> tuple[float, float, float]:
9     # Scale (Poisson) loss on total coverage
10    sum_pred = sum(logits, axis=seq_dim)
11    sum_true = sum(targets, axis=seq_dim)
12    scale_loss = mean(poisson_loss(sum_pred, sum_true))
13
14    # Shape (multinomial) loss on distribution
15    p_pred = logits / sum(logits, axis=seq_dim, keepdims=True)
16    shape_loss = -sum(targets * log(p_pred + epsilon)) / num_valid_positions
17
18    total_loss = shape_loss + shape_loss_coefficient * scale_loss
19
20    return total_loss
21
22

```

---

The genome annotation tracks are trained using a focal loss, which is a cross-entropy loss with weighted to focus the model on difficult to predict cases. The focal loss uses the recommended  $\gamma$  value of 2.0 [33] in the equation,

$$\mathcal{L}_{\text{focal}} = - \sum_i (1 - p_i)^\gamma \log p_i \quad (4)$$

where  $p_i$  is the predicted probability of the true class. During post-training, we continue to train the language modeling head using the same 15% BERT masking as during pre-training, with the aim of regularizing the model.

The three loss terms are summed to form the total training loss as,

$$\mathcal{L}_{\text{total}} = (\mathcal{L}_{\text{Poisson}} \times 0.2 + \mathcal{L}_{\text{multinomial}}) + \mathcal{L}_{\text{focal}} + \mathcal{L}_{\text{BERT}} \quad (5)$$

---

```

1 def focal_loss(logits: Array, targets: Array, gamma: float) -> float
2     # Compute log_probs and probs
3     log_probs = log_softmax(logits, axis=-1)
4     probabilities = exp(log_probs)
5
6     # Reshape for loss computation
7     num_classes = probabilities.shape[-1]
8

```

---

---

```

9   probabilities = reshape(probabilities, (-1, num_classes))
10  log_probs = reshape(log_probs, (-1, num_classes))
11  targets = reshape(targets, (-1,))
12
13  # Compute focal loss per position
14  loss = -sum(
15      take_along_axis(
16          (1 - probabilities) ** gamma * log_probs,
17          expand_dims(targets, axis=-1),
18          axis=-1,
19      ),
20      axis=-1,
21  ) # shape: (total_positions,)
22  loss = loss / len(loss)
23
24  return loss
25
26
27 def post_training_loss(
28     model: Model,
29     tokens: Array,
30     tracks_targets: Array,
31     annotation_targets: Array,
32 ) -> float:
33     # Model forward pass
34     tracks_logits, annotation_logits, _ = model(tokens)
35
36     # Functional tracks loss
37     scaled_targets = alphagenome_target_scaling(tracks_targets)
38     tracks_loss = multinomial_loss(tracks_logits, scaled_targets, shape_loss_coefficient=0.2)
39
40     # Genome annotation loss
41     annotation_loss = focal_loss(annotation_logits, annotation_targets, gamma=2.0)
42
43     # MLM forward pass and loss
44     masked_tokens, loss_mask = bert_masking_fn(tokens)
45     _, _, mlm_logits = model(masked_tokens)
46     mlm_loss = cross_entropy_loss(mlm_logits, tokens, loss_mask)
47
48     return tracks_loss + annotation_loss + mlm_loss
49

```

---

#### 4.3.6. Training Parameters

We describe the training parameters used during post-training, listed in [Table 3](#). For post-training we use the AdamW optimizer with a weight decay of 0.1. The learning rate is increased linearly for the number of ‘warmup tokens’ and then decreased using a square decay over the number of training steps to the final value. For the post-training, a single learning rate decay is implemented across the first and second subphases, and we used a lower learning rate compared to pre-training to retain the knowledge of pre-training in the model. We found that using a smaller batch size of 1M tokens improved performance on track prediction, relative to larger batch sizes like 2M or 8M, similar to previous work [7].

#### 4.3.7. Evaluation Procedure

After training, the post-trained model is evaluated on the held-out test set. For each species, the model is evaluated on the whole test set and the Pearson correlation is calculated. For the human species, the models were evaluated at each sequence length that was used during training ranging from 16 kb to 131 kb. For functional tracks, the model predictions are unscaled to raw values, and both the raw predicted values and ground truth labels are log transformed,  $\log(1 + x)$ , before calculating the Pearson correlation. For genome annotation, the Matthews Correlation Coefficient (MCC) for each track is calculated and reported, in line with the guidance of Chicco and Jurman [81].

### 4.4 Performance Infrastructure and Optimization

#### 4.4.1. Data Loading for Pre-training

To efficiently manage large-scale genomic datasets comprising millions of individual FASTA files, we implemented a scalable data loading pipeline designed to maintain a low memory footprint and minimize disk I/O,

independent of dataset size. The system enables direct, byte-level random access to genomic subsequences without loading full files or global indices into memory.

The approach consists of two stages: (1) a single-pass metadata indexing stage and (2) a byte-addressable data retrieval stage.

During the metadata indexing stage, each FASTA file is scanned once to extract attributes required for random access, including total sequence length, header byte length, and per-line sequence length. To eliminate the memory burden associated with tracking millions of FASTA files, we serialize all file paths and metadata into disk-backed, memory-mapped `*array_record*` databases, ensuring that metadata lookup does not scale RAM usage with dataset size. This design externalizes metadata management from system RAM and supports efficient, persistent lookup across arbitrarily large datasets.

In the retrieval stage, the complete corpus is treated as a virtual contiguous sequence that is partitioned into fixed-length segments used as atomic training units. Each segment is addressable by a unique global index. Given a segment index  $i$ , the corresponding source file is identified via a binary search over a precomputed array of cumulative segment counts. The global coordinates of the segment are then mapped to nucleotide offsets within the identified file. These offsets are translated into absolute byte positions using the metadata. A single read operation is issued to retrieve only the necessary byte range, which is then decoded directly into sequence data.

#### 4.4.2. Data Loading for Post-training

The post-training data comprises of three sections, file assemblies, genome annotations and functional tracks. The file assemblies and genome annotations are moderately sized and can be easily loaded for training. In contrast, the functional track dataset is large with data from over 17,000 tracks.

The ‘tensorstore’ Python package is used to store and load the functional track data. The data is stored on s3 bucket as tensorstore arrays with datatype of float16. A separate tensorstore is used for each species. The data is compressed with zstd level 3 compression, which takes advantage of the sparsity of elements. To enable fast loading of all the tracks in the tensorstore, the data is compressed in chunks of 8192 bp by the full number of tracks used for that species.

Each target example for training the functional track head is substantially large. For instance, a single human sample comprising 7,326 functional tracks over a 393 kb window corresponds to approximately 6 GB of data in memory. To efficiently stream these large tensors during training, we employ the grain Python framework with parallel data-loading across multiple worker processes. This design minimizes input latency and sustains high-throughput training by ensuring that model computation is rarely idle while awaiting sample retrieval.

#### 4.4.3. Training Optimizations

Training was conducted using multi-node setups on the Kyber cluster and additionally on TPUv6 hardware when available. Distributed training was used for scaling to long sequences, with the largest models requiring sequence parallelism across one or more nodes of 8 GPUs. We applied a series of optimizations to accelerate training while preserving performance. We provide an overview of these optimizations, details can be found in Section B.1.

**Mixed-precision Training.** We utilized mixed-precision to accelerate training while maintaining stability. Key compute-intensive components—including the stem, all down- and up-convolutions and transformer blocks (attention projections and feed-forward networks) were executed in `bfloat16`, while precision-sensitive components remained in `float32`, including all model parameter, embeddings, LayerNorm layers, and the final language model head. Loss calculations were retained in `float32` to ensure the gradient was computed with higher precision, and in ablations we found this to be important for training stability.

**Efficient Up-sampling Implementation.** To accelerate the up-sampling path, we replaced conventional transposed convolutions (Conv1DTranspose) with a **repeat+conv** approach. In this implementation, the input sequence is first up-sampled using nearest-neighbor repetition along the sequence length, duplicating each position by the up-sampling factor. A standard 1D convolution is then applied to the repeated sequence to mix features locally. Nearest-neighbor repetition is computationally cheap because it involves only memory replication, and the subsequent convolution operates at stride 1. This design reduces the cost of up-sampling

while maintaining the function of conventional transposed convolutions.

**Sequence Parallelism.** To scale our models to very long sequences, we implemented sequence parallelism, in which each batch is split along the sequence dimension across multiple devices. This approach allows each device to process a contiguous portion of the sequence, reducing memory requirements while enabling training on sequences up to 1 mb.

Certain operations require additional communication between devices to maintain correctness. For convolutional layers, we perform a collective permute to share the boundary tokens at the edges of each local sequence partition. For attention layers, the key and value tensors are all-gathered across devices, while the query tensor remains sharded. This allows each device to compute attention locally for its portion of the query, while still attending to all keys and values across the full sequence.

This setup allowed us to train our largest models up to 1Mb by splitting a single batch on the 8 GPUs of a single DGX so we could benefit from the high interconnect between devices on the same machine without exceeding per-device memory limits.

#### 4.4.4. Training Infrastructure

Model	Pre-training 8 kb 10.5T tokens	Length ext. 1-4 200B tokens	Length ext. 5-7 150B tokens	Post-training 1 1.2T tokens	Post-training 2 54B tokens
8M	1,608	40	40	NA	NA
100M	3,000	40	48	1,536	192
650M	6,736	88	80	1,824	192

**Table 5** | H100 GPU-hours required for all training stages of each model size. Pre-training was done with 8 kb sequences, while length extension was split into stages 1-4 (up to 131 kb, full data parallel) and stages 5-7 (262 kb to 1 mb, requiring sequence parallelism). Post-training stage 1 used full data parallelism, whereas stage 2 required sequence parallelism across a node. Tokens processed during each stage are indicated in the column headers.

All models were trained on either Cloud TPUs(v6) through the TPU Research Cloud program or On-Premise infrastructure. Jax was used to be able to easily target these two accelerator only configuration changes on our training codebase. Our On-Premise infrastructure is composed of machines with 8 H100 GPUs with 80GB of memory connected via NVLink. The machines are connected with RoCE to allow for direct GPU to GPU communication across nodes. We run our experiments using AIchor, our internal ML training platform platform on top of Kubernetes (<https://aichor.ai/>).

**Pre-training Compute.** The first pre-training phase accounted for the vast majority of compute. For the largest NTv3 650M (pre) model, processing the full 10.5 trillion tokens required over nine months of H100 GPU-hours. By leveraging a 4-node (32 GPU) setup, the end-to-end wall-clock time was reduced to just under nine days (Table 5). Sequence processing throughput for each model was approximately 3.5M tokens/sec for the 650M model, 7.8M tokens/sec for the 100M model, and 14.5M tokens/sec for the 8M model. In the later stages of sequence length extension we achieved lower throughput due to longer sequence lengths and, in the case of the final stages, sequence parallelism. Despite this, the shorter length extension phases, meant that the length extension stages were fast with across all models.

**Post-training Compute.** Post-training was split in two subphases to maximize efficiency and hardware usage. In the first subphase, models were trained on sequence lengths up to 131 kb using full data parallel. Using data parallel enabled faster throughput due to reduced communication between GPUs. In the second subphase, the models were trained on longer sequences up to 1 mb using sequence parallelism across all 8 GPUs. Using sequence parallelism reduced training throughput due to the higher communication demands. In both subphases, the 100M model training was bottlenecked by the dataloading of the functional tracks data.

## 4.5 Pre-training dataset

#### 4.5.1. OpenGenome2

We pre-trained NTv3 on a subset of the OpenGenome2 dataset introduced in Evo2 [28], which originally comprises 8.84 trillion non-redundant nucleotides spanning diverse taxa and genomic contexts. Specifically, the dataset includes 357 billion nucleotides from prokaryotic genomes, 6.98 trillion nucleotides from eukaryotic genomes, 854 billion nucleotides from metagenomic sequencing data, 2.82 billion nucleotides from organellar genomes, and 602 billion nucleotides from targeted subsets of eukaryotic sequences enriched for functional regions centered on protein-coding genes. The original dataset can be found here: <https://huggingface.co/datasets/arcinstitute/opengenome2>.

For pre-training NTv3, we applied two filtering steps to this dataset. First, we removed all sequences shorter than 1 kb, which primarily excluded promoter regions and accounted for a negligible number of nucleotides. Second, within the `ncbi_eukaryotic_genomes` subset, we retained only sequences longer than 32 kb; this filtering removed approximately 800 billion nucleotides. After these steps, the total number of unique nucleotides used for pre-training was approximately 8 trillion composed of 6.2 trillion from the `ncbi_eukaryotic_genomes` subset, and 1.8 trillion from all other subsets (Table 13).

#### 4.5.2. Data Processing and Augmentation

To train the model, we utilized the original OpenGenome2 splits for train, test and validation and divided each fasta file into fixed-length genomic windows without overlap. In the pre-training of NTv3 we sought to avoid padding as padded tokens can introduce distributional artifacts, reduce the proportion of informative positions contributing to the loss, and create boundary effects that are incorporated into the receptive fields and can be particularly detrimental for U-Net architectures.

To eliminate the need for padding, we restricted training to sequence segments longer than the model’s input length at each stage. Since genomic sequences rarely have lengths that are exact multiples of the training window, a single fixed tiling would leave parts of each sequence uncovered. To ensure complete coverage, we generated two complementary tilings for every FASTA file. The first divides the sequence into non-overlapping windows starting from the first base and discards any leftover bases at the end. The second performs the same operation in reverse, starting from the final base and discarding any incomplete window at the beginning. By mixing segments from both tilings during training, the model sees every genomic region in at least one full-length window, entirely avoiding padding.

#### 4.5.3. Pre-training Data Composition and Sampling

Sequence Length	Eukaryotic Genic Windows	GTDB v220	IMGPR	IMGVR	Metagenomes	mRNA	mRNA Splice	ncRNA	Organelles	Transcripts
1,024 - 2,048 bp	2027	1516	0	124	132013	12747	9410	3057	0	603
2,048 - 4,096 bp	4899	12000	27	520	184436	17940	38018	2397	0	2146
4,096 - 8,192 bp	11943	26433	352	7911	169243	13256	29753	1682	0	3563
≥8,192 bp	395278	314785	5370	29748	344121	7294	14277	4971	2784	2267

**Table 6** | Token count by sequence length and OpenGenome2 sub-dataset. This dataset consisted of all sequences longer than 1 kb. Sequences longer than 8 kb are chunked to 8 kb. The total number of tokens is 1,808,930.

During the initial pre-training phase, which used sequences up to 8,192 bp, we followed prior work [28] and trained on all subsets of the OpenGenome2 dataset, with the exception of whole eukaryotic genomes (`ncbi_eukaryotic_genomes`). These genomes, though comprising 6.98 trillion nucleotides, are dominated by long intergenic regions. Segmenting them into 8 kb windows would often yield sequences with minimal functional content. We therefore deferred their inclusion until the model was able to process longer contexts. This approach leaves a pre-training dataset of approximately 1.8 trillion nucleotides, enriched for gene-proximal functional regions.

To ensure meaningful input for the transformer, we further filtered sequences shorter than 1,024 bp, which correspond to fewer than eight tokens in the transformer. This filtering removes the Promoters subset of OpenGenome2, but its contribution to the total token count is negligible.

For 8kb-training, sequences were partitioned into length-specific data sources, where `datasourcei` contained all sequences  $s$  such that:

$$2^i \leq \text{len}(s) < 2^{i+1}$$

for  $i = 10, 11, 12$  (1 kb, 2 kb, 4 kb). All sequence  $\geq 8$  kb were placed in a final bin. In total, this yielded four distinct data sources, collectively partitioning all eligible tokens in OpenGenome2 while excluding whole genomes and sequences shorter than 1,024 bp.

Training batches were sampled from these bins following the strategy in Section 4.5.2. Each sequence was segmented twice at its target length: once in a left-to-right pass and once in a right-to-left pass. The resulting paired segmentations were then combined and shuffled, yielding four data sources of fixed-length sequences. This procedure ensures full coverage of all pre-training tokens. Most tokens appear in both views, while only a small fraction near the sequence boundaries are included once.

In training we specifying sampling weights for each sequence length (data source), and, in this way, maintain precise control over the fraction of tokens seen at each sequence length. For 8kb-training, these weights were set proportional to each bin’s share of the total token count. The final sampling weights for the 1 kb, 2 kb, 4 kb, and 8 kb bins are presented in Table 7.

Sequence length range	% of total tokens
[1024, 2048)	0.079
[2048, 4096)	0.130
[4096, 8192)	0.132
[8192, $\infty$ )	0.660

**Table 7** | Token distribution in the pre-training phase by sequence length.

#### 4.5.4. Sequence Length Extension Distribution

After 8kb-training, we extended the model’s context length from 8,192 bp to 1 Mb through a staged curriculum. Approximately every 50B tokens, we introduced a new maximum sequence length, doubling it at each stage (16 kb, 32 kb, 64 kb, ..., 1 Mb). Throughout this process, all shorter sequence lengths remained in the training mixture, allowing the model to acquire long-range dependencies gradually while retaining strong performance on short contexts.

As in 8kb-training, we created a distinct data source for each sequence length. For subsets composed of inherently short sequences (e.g., transcripts, metagenomes, and prokaryotic genomes), sequences were assigned to  $\text{datasource}_i$  if their length satisfied

$$2^i \leq \text{len}(s) < 2^{i+1}.$$

As the maximum context length grew through the stages, the set of active bins expanded from  $\text{datasource}_{14}$  up to  $\text{datasource}_{20}$ .

Whole eukaryotic genomes (`ncbi_eukaryotic_genomes`) required a separate strategy, as they constitute the majority of tokens and are often extremely long, with many sequences exceeding one megabase. We introduce them earlier in training, at the 32-kb stage, to allow the model more exposure, particularly to smaller chunks that may be more representative of practical downstream applications. We note that 705 of the total 15,032 distinct genomes in `ncbi_eukaryotic_genomes` that are of length less than 32 kb. These genomes account for 0.8T of the 6.98T tokens, leaving a total of 6.2T tokens in whole genome distribution (`ncbi_eukaryotic_genomes`) for the remaining length extension phase. We chose not to include these shorter length whole genome sequences in the the early stages of training to focus short context training on functionally enriched areas.

Starting at the 32 kb stage, each eukaryotic genome was subdivided into non-overlapping windows matching the stage’s target length, and all such windows were added to the corresponding  $\text{datasource}_i$ . In other words, for stage  $i$  with target length  $2^i$ , every  $2^i$ -bp chunk from genomes at least  $2^i$  bp long was included. This ensured that the model saw whole-genome sequences progressively, integrating both shorter and longer contexts.

For example, during the 64 kb stage,  $\text{datasource}_{16}$  included (i) all non-eukaryotic sequences between 64 kb and 128 kb, and (ii) all 64 kb chunks extracted from eukaryotic genomes at least 64 kb in length, including



megabase-scale chromosomes. This staged inclusion allowed the model to encounter whole-genome structures earlier in training rather than only in the final stage.

At each stage, the primary focus was to adjust the model to the newly introduced longest sequences. To maintain robust performance across all context lengths, we dynamically adjusted the sampling weights of each data source. The newly introduced longest sequence length was initially assigned 40% of the sampling probability, while the remaining 60% was rescaled among all shorter bins. To prevent shorter contexts from being underrepresented, we enforced a minimum sampling probability of 5% for every bin; if rescaling caused a bin to fall below this threshold, it was set to 5%, and the remaining weights were re-normalized. The evolution of these sampling distributions is detailed in Table 2. We provide below the basic implementation for generating this sequence length extension schedule, given an initial distribution over datasets, a number of stages to extend for, and the new weight with which to fold in the longest sequence.

---

```

1
2 def enforce_minimum_and_normalize(x: Array[N], min_value: float) -> Array[N]:
3     # Iteratively raise all entries below min_value and rescale others
4     mask = (x < min_value)
5     while Any(mask):
6         deficit = Sum(min_value - x[mask])
7         above = x[~mask]
8         available = Sum(above)
9         x[mask] = min_value
10        x[~mask] = above * (available - deficit) / available
11        mask = (x < min_value)
12    return x
13
14 def calculate_sequence_length_extension_schedule(
15     num_stages: int,
16     init_dist: Array[K],
17     new_weight: float
18 ) -> Array[num_stages + 1, K + num_stages]:
19     # init_dist is the distribution from 8k pre-training [0.079, 0.130, 0.132, 0.660]
20     # Stage 0 distribution; pad with zeros to length K + num_stages (pad = append zeros)
21     weights = init_dist
22     schedule = [ weights + [0.0] * (K + num_stages - Length(weights)) ]
23     for t in Range(num_stages):
24         scale = 1.0 - new_weight
25         weights = weights * scale # shrink existing datasets
26         weights = Concat(weights, [new_weight]) # add new dataset at stage t
27         weights = enforce_minimum_and_normalize(weights, min_value = 0.05)
28         padded = weights + [0.0] * (K + num_stages - Length(weights)) # zero-pad for future stages
29         schedule = Append(schedule, padded)
30    return schedule
31

```

---

## 4.6 Post-training Dataset

Our post-training dataset combines *functional tracks*—capturing regulatory and transcriptional activity—with *genome annotation tracks* across 24 different animal and plant species. The functional tracks include RNA-seq, ATAC-seq, DNase-seq, and ChIP-seq data across nine species, totaling 15,889 tracks (human (*Homo sapiens*): 7,362; mouse (*Mus musculus*): 2,450; fly (*Drosophila melanogaster*): 180; Arabidopsis (*Arabidopsis thaliana*): 1,899; rice (*Oryza sativa*): 1,392; maize (*Zea mays*): 921; wheat (*Triticum aestivum*): 776; soybean (*Glycine max*): 590; cotton (*Gossypium hirsutum*): 319). In addition, we compiled genome annotation tracks for those species plus 15 additional ones covering various model organisms, encompassing diverse gene elements (protein-coding genes, lncRNAs, 5'UTR, 3'UTR, exon, intron, splice acceptor and donor sites) and regulatory elements (polyA signal, tissue-invariant and tissue-specific promoters and enhancers, and CTCF-bound sites) when available for a given species. Below, we describe the data sources and pre-processing steps for each species and data type.

### 4.6.1. Functional Tracks

The animal functional tracks combine all human and mouse regulatory datasets used in Borzoi [7]—including ENCODE v3, FANTOM5 CAGE, and GEO RNA-seq, ATAC-seq, DNase-seq, and ChIP-seq—together with the *Drosophila* embryogenesis single-cell atlas of Calderon et al. [82].

Plant functional data were obtained from large public RNA-seq, ATAC-seq, and ChIP-seq collections from NCBI repositories (e.g., SRA/GEO; <https://www.ncbi.nlm.nih.gov/>), using stringent quality filters and retaining only well-annotated wild-type samples across six major species.

**Human and Mouse ENCODE data.** For both human and mouse, we used ENCODE v3 BigWig signal tracks corresponding to the exact sets included in Borzoi [7]. Data were retrieved directly from the ENCODE portal. When consortium-provided merged-replicate BigWigs were available, we used them preferentially; otherwise, we merged individual replicates ourselves to obtain unified signal tracks. All human ENCODE files were accessed on February 24, 2025, and all mouse ENCODE files on July 20, 2025.

**Human and Mouse CAGE data.** CAGE datasets for both species were obtained from the FANTOM5 database, restricted to the standardized, reprocessed collections to ensure consistency across phases and assemblies. Specifically, we used the hg38 human CAGE data from [https://fantom.gsc.riken.jp/5/datafiles/reprocessed/hg38\\_latest/basic/](https://fantom.gsc.riken.jp/5/datafiles/reprocessed/hg38_latest/basic/), and the mm10 mouse data from [https://fantom.gsc.riken.jp/5/datafiles/reprocessed/mm10\\_latest/basic/](https://fantom.gsc.riken.jp/5/datafiles/reprocessed/mm10_latest/basic/). All files were accessed on July 24, 2025.

For both species, we used the `ctss.bed` files. Replicates were identified based on distinct `experiment_name` (e.g., CNhs1015) and `donor_id` (e.g., donor1, donor2). Each strand (+ and -) was processed independently. After download, we applied counts-per-million (CPM) normalization separately to each strand of each replicate. Replicates were then merged by averaging their normalized signal values, and the resulting processed files were converted to BigWig.

**Human and Mouse GEO data.** For human and mouse functional tracks from GEO, we collected RNA-seq, ATAC-seq, DNase-seq, and ChIP-seq datasets and processed them using matching pipelines across species. GEO accessions were mapped to their corresponding SRA runs, and raw reads were retrieved and aligned using `bowtie2` [83] with default parameters. Human reads were aligned to GRCh38, and mouse reads to GRCm38/mm10. Resulting BAM files were coordinate-sorted using `samtools` [84]. Genome-wide coverage profiles were computed and normalized to CPM without additional filtering or duplicate removal. Only samples with complete metadata and successful processing were retained.

**Drosophila Single-cell RNA- and ATAC-seq.** For *Drosophila melanogaster*, we used the single-cell RNA-seq and ATAC-seq atlas of embryogenesis generated by Calderon et al. [82]. This dataset profiles nearly 1.5 million nuclei across the full 0–20 h developmental window and, through neural-network-based age inference, provides a continuous trajectory of chromatin accessibility and gene-expression dynamics with high temporal resolution. We extracted the published cell-type and developmental-age annotations and produced genome-wide RNA-seq and ATAC-seq coverage tracks by aggregating nuclei belonging to each annotated population. Following Calderon et al., who consider NNLS mixture coefficients above 0.1 to represent high-confidence correspondence between RNA- and ATAC-derived cell states, we restricted our dataset to the 90 cell types exhibiting the strongest RNA-ATAC agreement. These high-quality, modality-matched tracks provide a temporally resolved view of transcriptional and regulatory activity throughout *Drosophila* embryogenesis.

**Plant RNA-seq Tracks.** The NCBI Short Read Archive (SRA) was queried for all RNA-Seq Experiments for six plant species: *Arabidopsis thaliana*, *Zea mays*, *Glycine max*, *Gossypium hirsutum*, *Triticum aestivum* and *Oryza sativa*. The metadata for the experiments was downloaded using Run Selector. Only experiments with “Treatment” and “Genotype” metadata were kept. To ensure the quality of the sequencing data experiments were filtered for “LibraryLayout” equal to “PAIRED”, “Bases” greater than or equal to  $2.5 \times 10^9$  and “AvgSpotLen” greater than or equal to 200 base pairs. Additional metadata in NCBI for the experiments was obtained by additionally querying for the biosample metadata associated with each experiment and for the bioproject data for each set of experiments. After the filtering for complete metadata and sequencing quality the number of experiments for each species was: *A. thaliana*: 8712, *Z. mays*: 3326, *G. max*: 2422, *G. hirsutum*: 902, *T. aestivum*: 2436 and *O. sativa*: 4410.

The NCBI experiments were organised into BioProjects, the experimental design of each BioProject was determined, and the experiments were filtered to select only a single biological/technical replicate per biological condition and to exclude mutants. Mutants were excluded since the genotype would differ from the genome DNA track for which the model would be trained to predict gene expression coverage. Additionally, the separate RNA sequencing of the mutant vs control indicated that the original experimenter expected the mutation to cause the gene expression to differ from the wild-type. To perform this filtering a datacard was created for each BioProject listing the BioProject Title and Description and each Experiment. For each BioSample the

tissue, treatment, Sample Name, Experiment ID, BioSample ID, cultivar, replication, genotype and dev\_stage in YAML format. A system and user prompt were created to submit together with each BioProject data card individually to the OpenAI model o3. The prompt requested one replicate of each biological condition with mutants excluded. To sense-check the results the model was asked to also return a concise matrix summary of the BioProject study design, its reasoning for its selection of the experiments and a confidence score from 1-5, with a semantic interpretation of the scoring scheme provided in the prompt. All BioProjects for which the analysis was less confident than 4 out of 5 (“data essentially complete, minimal inference to interpret any abbreviations”) were discarded. As an additional sense check, 10 processed Soy BioProjects were selected at random and were curated by an expert to ensure agreement with the experiments selected by the model. After this final filtering step, the numbers of RNA-seq experiments were: *A. thaliana*: 1639, *Z. mays*: 813, *G. max*: 576, *G. hirsutum*: 286, *T. aestivum*: 762 and *O. sativa*: 1356.

Each RNA-seq experiment was aligned to the reference genome and corresponding GTF annotation file of the species using STAR [85]. To generate the genome index, the following options were used:

```
STAR --runMode genomeGenerate \
      --genomeFastaFiles GENOME_FILE \
      --limitGenomeGeneratorRAM 300000000000 \
      --sjdbGTFfile GTF_FILE \
      --sjdbOverhang 299 \
      --genomeSAindexNbases VALUE
```

where VALUE was set to the recommended  $\min(14, \log_2(\text{GenomeLength})/2 - 1)$ .

FASTQ files were trimmed using Trim Galore [86] with the options “-quality 20 -length 20”. The trimmed FASTQ files were then aligned to the indexed genome using STAR with default parameters and the option “-outSAMtype BAM SortedByCoordinate”.

Coverage tracks were generated using bamCoverage [87] with the options “-normalizeUsing CPM -binSize 1”.

**Plant ATAC- and ChIP-seq Tracks.** Similarly to the RNA-seq track processing, ATAC- and ChIP-seq tracks and their associated metadata were queried from the NCBI SRA. A comparable set of filters was applied, including “LibraryLayout” equal to “PAIRED,” “Bases” greater than or equal to  $6 \times 10^9$ , and “AvgSpotLen” greater than or equal to 300 base pairs. As with RNA-seq track processing, mutants were excluded by selecting only wild-type samples from the “Genotype” (or “Cultivar”) field. For *A. thaliana*, this included any entry containing the string “Columbia 0”; for *Z. mays*, the string “B73”; for *G. max*, the string “Williams 82”; for *G. hirsutum*, the string “G. hirsutum acc. J668”; for *T. aestivum*, the string “Chinese Spring”; and for *O. sativa*, the string “japonica.” In addition, any sample annotated as “wild type” or similar variations (e.g., “wt”) was also included. Given that the number of ATAC- and ChIP-Seq experiments is much smaller than those of RNA-Seq, we manually curated the missing metadata entries for all experiments that passed the initial filters to obtain reliable tissue, and for ChIP-seq experiments, transcription factor or histone mark annotations.

Then each ATAC- and ChIP-seq experiments were processed using the following pipeline:

1. FASTQC [88]: to assess the quality of raw sequencing reads.
2. Trim Galore [86] with the options -q 20 -phred 30 -stringency 3 -length 20 -e 0.1: to trim adapter sequences and low-quality bases.
3. Bowtie2 [83] with the options -X2000: to align reads to the reference genome, allowing for paired-end mapping with a maximum fragment length of 2000 bp.
4. Python script: to count the number of reads mapping to nuclear versus organellar genomes, enabling assessment of contamination from mitochondrial or chloroplast DNA.
5. SamTools [84] with the options -f 2 -F 1804 -q 30: to filter for properly mapped, paired reads with MAPQ higher or equal than 30, followed by fixmate -r to correct mate information, and a second filtering with -f 2 -F 1804 to remove orphan or inter-chromosomal reads.
6. Bash script: to compute library complexity, which estimates the number of unique fragments and evaluates sequencing depth sufficiency.
7. Picard’s MarkDuplicates [89]: to identify and remove PCR duplicates, preventing bias from overrepresented fragments.

8. Python script: to calculate fragment length distribution on deduplicated reads, which is particularly important for ATAC-Seq to infer nucleosome positioning.

We note that the Python and Bash scripts mentioned above were obtained from the documentation of ENCODE’s ATAC-Seq pipeline. After processing, the following quality control filters were applied: experiments with "Basic Statistics" and "Per Base Sequence Quality" marked as "Pass" in the FASTQC report; experiments with a Bowtie2 alignment rate  $\geq 0.8$ ; experiments with an organelle read fraction  $< 0.5$ ; experiments meeting library complexity criteria (PBC1  $\geq 0.5$ , PBC2  $\geq 1$ , NRF  $\geq 0.5$ ); experiments with a duplication rate  $< 0.4$ ; experiments showing the presence of both NFR and mono-nucleosome peaks in the fragment length distribution; and experiments passing a read-depth filter to ensure sufficient sequencing coverage, using species-specific thresholds scaled from the ENCODE human guideline of 25 million mapped reads according to genome size. After this final filtering step, the numbers of ATAC-Seq and ChIP-Seq experiments were: *A. thaliana*: 260, *Z. mays*: 100, *G. max*: 14, *G. hirsutum*: 33, *T. aestivum*: 14 and *O. sativa*: 36.

For ATAC-seq experiments, a Tn5-shifted BAM file was generated to accurately represent insertion sites. Specifically, the BAM file was processed with alignmentSieve [87] using the `-ATACshift` option to adjust reads according to the Tn5 transposase cut site. The resulting BAM was then sorted and indexed using SamTools to prepare it for downstream analysis. An insertion signal track with 1-bp resolution was generated without centering or extending the reads. Finally, bamCoverage [87] was used to create a normalized BigWig track (`-normalizeUsing CPM`) at 1-bp bin size, representing the Tn5 insertion profile across the genome. For ChIP-seq experiments, two types of coverage tracks were generated depending on the target. For transcription factors or other proteins producing narrow, well-defined peaks, reads were centered using bamCoverage (`-centerReads`) and CPM-normalized at 1-bp bin size. For histone modifications, the full fragment coverage was used without centering and similarly CPM-normalized at 1-bp bin size.

**Data Normalization.** All functional targets are represented as genome-wide BigWig signal tracks. For each track, we computed genome-wide summary statistics across all assembled chromosomes and used the resulting mean signal value  $\mu_t$  as the primary normalization factor. During training, each target track was normalized by its mean to stabilize optimization,

$$\tilde{y}_{b,\ell,t} = y_{b,\ell,t} / \mu_t.$$

RNA-seq tracks, which exhibit heavy-tailed signal distributions, received an additional sublinear power transform ( $x^{0.75}$ ) together with smooth clipping of extreme values to reduce the influence of outliers. At evaluation time, model outputs were mapped back to the original scale by analytically inverting the clipping and power transformation (for RNA-seq) and multiplying by  $\mu_t$ . This procedure was introduced by Avsec et al. [8], and was proven to yield numerically stable training targets while preserving interpretability in the raw BigWig signal space.

#### 4.6.2. Genome Annotation Tracks

We followed the same methods described in SegmentNT [37] to annotate the human genome; however, we extended the elements to include stranded UTRs, start and stop codons, open reading frames (ORFs), skipped exons, and always-on exons. Specifically, UTRs were explicitly separated by strand to preserve directionality, resulting in four distinct UTR feature classes. ORFs, start and stop codons were extracted from CDS annotations, where phase information was used to adjust start and end coordinates. Exon annotations were also extended to include isoform-level data: exons present in all isoforms of a gene were annotated as always-on exons, while exons subject to alternative splicing—identified by their presence in some transcripts but absence in others—were annotated as skipped exons.

Similar to Segment NT [37], the human annotation files were extracted from GENCODE V44. Annotations were also filtered to exclude level 3 transcripts, ensuring all training data was manually annotated. Thus, annotations for the human genome included a total of 21 elements (protein coding gene, lncRNA, exon, intron, splice donor and acceptor sites, CTCF bound, polyA signal, tissue specific and invariant enhancers, tissue specific and invariant promoters, 5UTR+, 5UTR-, 3UTR+, 3UTR-, skipped and always on exons, start and stop codons, ORFs).

Annotations were generated for all other species following the same methods, but their corresponding GFF3 and GTF files were retrieved from Ensembl databases. These datasets were restricted to the following 15

gene-structural features: protein coding gene, lncRNA, exon, intron, splice donor and acceptor sites, 5UTR+, 5UTR-, 3UTR+, 3UTR-, skipped and always on exons, start and stop codons, and ORFs.

#### 4.6.3. Species Genome Data Splits

Post-training of NTv3 was done on genomes and functional data from the following 24 species and reference genomes: human (*hg38*), mouse (*mm10*), fly (*dm6*), Arabidopsis (*TAIR10*), maize (*Zm-B73-REFERENCE-NAM-5.0*), rice (*IRGSP-1.0*), soybean (*Glycine-max-v2.1*), wheat (*IWGSC*), cotton (*Gossypium-hirsutum-v2.1*), chicken (*galGal6*), zebrafish (*danRer11*), worm (*ce11*), clownfish (*AmpOce1*), bison (*Bison-UMD1*), long-tailed chinchilla (*ChiLan1*), cat (*Felis-catus-9*), sea vase (*KH*), pigtail macaque (*Mnem1*), trout (*fSalTru1*), dog (*ROS-Cfam-1*), canary (*SCA1*), pufferfish (*TETRAODON8*), rat (*mRatBN7*), gorilla (*gorGor4*). For each species genomes we created train, validation and test splits DNA sequences controlled for sequence similarity as described below.

**Human and Mouse.** To remain comparable with existing models, the folds used by Borzoi [7] are used to define the training, validation and test sets. Specifically, fold3 is used for test, fold4 for validation and the rest are used for training. In contrast with previous approaches that directly use these sequences, we merge the sequences for each set into continuous regions. These regions are then split using a stride of 0.01% of the sequence length. This approach enables high sequence diversity, ensuring that the model seldom sees repeat samples.

**Other Species.** To control for data leakage within a species, we used a homology-based clustering approach. Continuous spans on the genome were divided into 1 mb gene-aware windows; that is, each window was expanded to include the last gene that starts within its given span, ensuring that no gene was split across windows. Chromosomes or scaffolds shorter than 1 mb were treated as their own windows. Ensembl BioMart was used to extract homologous gene pairs. The average base pair size of the homologous genes across windows was then used to weight the edges of a symmetric window-to-window connectivity matrix. The Markov Clustering (MCL) algorithm [90] was used to create clusters among the connected windows. Inflation rates ranging from 1.5 to 5.0 were evaluated in increments of 0.5. The optimal rate was selected based on the highest modularity score [91]. The clusters were then stratified into train, validation, and test splits balancing gene and no gene containing windows and according to a predefined target validation and test size. For species where the algorithm could not meet the predefined target validation and test set sizes, edges representing a shared homology of less than 10% of the window size were pruned. These species included *A. thaliana*, *C. elegans*, *C. intestinalis*, and *O. sativa*.

## 4.7 Evaluation on Sequence-to-Function Benchmarks

### 4.7.1. Benchmarking Against Borzoi Track Predictions

NTv3 650M is compared with Borzoi in two contexts, firstly, Borzoi is evaluated at single nucleotide resolution predictions on our tracks, secondly, NTv3 is finetuned on Borzoi’s 32bp data and evaluated on their context. Both evaluations are done with Borzoi’s model replicate 0, on fold4 of Borzoi’s data (their and our test split).

Firstly, we compare our model to Borzoi’s ability to predict functional tracks at single base pair resolution. To do this, the Borzoi 32bp predictions are unscaled (following their scaling methodology) and repeated 32 times to create a prediction for each nucleotide. The unscaling uses the assay-specific scaling factor, soft and hard clips and aggregation function described in Borzoi’s repository. We compare the models at varying sequence lengths by merging overlapping regions and then splitting the regions into sequences of the selected length. Since both models are trained to predict the inner 37.5% of the sequence, we use a stride of 0.625 (calculated as  $1 - 0.375$ ). This ensures that the metrics are calculated over every nucleotide, irrespective of sequence length. We used all the tracks predicted by NTv3, except the GTex tracks, because they had been renormalised, making direct comparison unfair. The raw predictions and targets are log transformed (as with other functional track results), and the transformed Pearson Correlations are reported. We compare these results to our own results and report the mean per assay in Fig. 3F, G.

For the comparison at 32bp, the NTv3 model is finetuned on Borzoi’s exact data. Borzoi’s dataset is downloaded as TFrecords, and stored (with no modifications) in a tensorstore for finetuning. An additional (randomly



initialized) head is added to NTv3 to predict all of their human and mouse tracks. NTv3 is then finetuned on their data and the best checkpoint, selected via validation performance on the validation set (fold4), is used for test set (fold3) evaluation. Both NTv3 and Borzoi’s fold0 model are evaluated on the full test set and the average performance per assay is reported.

#### 4.7.2. Benchmarking Against SegmentNT Genome Annotation

We benchmark NTv3 650M against SegmentNT on genome annotation in the two categories of human genome annotation and multispecies genome annotation. For human genome annotation, we use human SegmentNT and for multispecies annotation we use Multispecies SegmentNT. For both evaluations we evaluate both the baseline model and NTv3 on the intersection of the baseline hold out set and NTv3 hold out set. We remove any sequences with N’s in them since SegmentNT was not trained on sequences with N’s. We use a sequence length of 29,952 nucleotides since it is the largest number under 30 kb, that is divisible by 6 (SegmentNT requirement) and 128 (NTv3 requirement). We calculate the MCC over the inner 37.5% of the sequence and use an overlap of 62.5% to ensure that every nucleotide is included in the calculation. Additionally, we evaluate at NTv3 at 1 mb to show the impact of using longer sequence lengths. The results from the human comparison are in [Figure 3H](#), and the multispecies comparison in [Figure 3E](#).

#### 4.7.3. Benchmarking Against Augustus

NTv3 is benchmarked against the Augustus gene finder on genome annotation tracks, exon, intron, 5UTR, 3UTR, splice site donors and splice site acceptors. We evaluate Augustus using all isoforms across the whole chromosome. Augustus was run with the following settings:

- AUGUSTUS\_CONFIG\_PATH=path –strand=both,
- outfile=out.gff,
- and –gff3=on –introns=on –UTR=on –species=human sequence.fasta.

Augustus was evaluated on the intersection of the SegmentNT and NTv3 test sets so that the results could be directly compared. As with the other genome annotation comparison, we use MCC as the performance measure. The results are included with the SegmentNT comparison in [Figure 3H](#).

#### 4.7.4. Benchmarking Against SpliceAI

We benchmark NTv3’s ability to predict splice sites with SpliceAI. We evaluate all 24 species that NTv3 is trained on at 29,952bp using the intersection of the SegmentNT and NTv3 test splits. We use SpliceAI to make predictions for the full sequence. Since SpliceAI was only trained on genes in the forward direction, we identify regions with regulatory elements in the reverse direction and use the reverse complement of the sequence and correspondingly reverse the predictions. The results for all species for donor and acceptor site are presented in ([Fig. 3I](#)).

### 4.8 NTv3 BENCHMARK

To evaluate the usefulness of the representations acquired during post-training, we introduce the NTv3 BENCHMARK, a unified suite of downstream fine-tuning tasks spanning both the animal and plant kingdoms. The goal of this benchmark is to measure how effectively post-trained representations transfer to *supervised fine-tuning tasks*, across both genome-annotation and functional-regulatory prediction settings.

The construction of the benchmark follows two simple principles. First, for species that were present during post-training (e.g., human, chicken, Arabidopsis, rice, maize), we include only regulatory tracks that did *not* appear in the post-training supervision. This ensures that fine-tuning performance reflects the quality of the learned representations rather than any reuse of supervised post-training signals. Second, for species that were absent during post-training (tomato, cattle), all available tracks constitute genuinely new fine-tuning tasks.

Together, these principles yield a clean and controlled collection of tasks that isolates the contribution of representation learning, avoids supervision overlap across stages, and provides a broad, phylogenetically diverse setting for benchmarking genomic foundation models.

#### 4.8.1. Downstream tasks

Species	Track Type	Kingdom	Species in Post-training	Tracks	Tissues / Cell lines	Notes
Human	Chromatin Accessibility	Animal	Yes	5	5	New tracks
Human	Histone modifications	Animal	Yes	4	2	New tracks
Human	Transcription initiation (PRO-cap)	Animal	Yes	10	5	New assay
Human	RNA binding sites (eCLIP)	Animal	Yes	10	1 (K562)	New assay
Human	Gene Expression	Animal	Yes	5	3	New tracks
Human	Genome annotation	Animal	Yes	4	NA	Exon/Intron/Splice/Start
Chicken	Chromatin Accessibility	Animal	Yes	7	7	New tracks
Chicken	Gene Expression	Animal	Yes	7	7	New tracks
Cattle	Genome annotation	Animal	No	4	NA	Exon/Intron/Splice/Start
Arabidopsis	mRNA translation (Ribo-seq)	Plant	Yes	4	3	New assay
Rice	mRNA translation (Ribo-seq)	Plant	Yes	14	7	New assay
Maize	mRNA translation (Ribo-seq)	Plant	Yes	8	5	New assay
Tomato	Chromatin Accessibility	Plant	No	7	2	New tracks
Tomato	Gene Expression	Plant	No	13	2	New tracks
Tomato	Genome annotation	Plant	No	4	NA	Exon/Intron/Splice/Start

**Table 8** | Summary of fine-tuning tasks included in the NTV3 BENCHMARK. Columns report the number of tracks, distinct tissues or cell lines, and whether the species was seen during post-training.

The NTV3 BENCHMARK encompasses two complementary classes of fine-tuning tasks: genome-annotation prediction and functional-regulatory prediction. Together, they provide a controlled and taxonomically broad setting for assessing how well the learned representations support both structured sequence-annotation tasks and heterogeneous regulatory assays.

For genome annotation, we fine-tune models on four universally available labels—**exon**, **intron**, **splice\_acceptor**, and **start\_codon**. These labels capture essential elements of gene structure and form a consistent cross-species evaluation set. Each label is trained in a *single-task* configuration: one model per annotation type, with a binary-classification head predicting the presence of the feature at each nucleotide. This results in four independent fine-tuning runs per species.

Functional-regulatory prediction follows a complementary *multi-task* design. For each species, all available regulatory assays—spanning chromatin accessibility (ATAC/DNase), histone-mark ChIP-seq, PRO-cap, eCLIP, RNA-seq, and Ribo-seq—are grouped into a single multi-output head, and a single model is fine-tuned jointly across all tracks. We motivate this joint-training strategy both as a practical choice—substantially reducing the number of ablation experiments required—and as a principled one: in Supplementary Section B.6.2, we show that joint training yields similar asymptotic performance as training separate models for each individual track.

Below, we describe the data collected and processed for each species included in the NTV3 BENCHMARK.

**Human Functional Tracks.** For human fine-tuning, we selected a diverse panel of regulatory assays from the ENCODE consortium, including chromatin accessibility (ATAC/DNase), histone-mark ChIP-seq, PRO-cap, eCLIP, and RNA-seq. These datasets span multiple cellular contexts—such as GM18861, BLaER1, MCF10A, motor neuron, ureter, Caco-2, Calu3, and K562 cells, as well as selected GTEx tissues—providing broad coverage of transcriptional activity, chromatin state, and protein–DNA or protein–RNA interactions. Only assays that were not used during post-training were retained, ensuring that all human functional tracks represent genuinely new fine-tuning tasks.

**Chicken ATAC- and RNA-seq Tracks.** ATAC-seq and RNA-seq data were derived from an atlas of regulatory elements in chicken (*G. gallus*), as reported by Pan et al. [92]. This study provided paired ATAC-seq and RNA-seq data across seven tissues: trachea, testis, bone marrow, uterus, heart, kidney, and thymus. Raw data were obtained from the NCBI SRA under BioProject accession number PRJEB55656. Each ATAC-seq and RNA-seq experiment was processed using the same pipelines described above for plant species, and the same post-processing quality control filters were applied to retain only high-quality experiments. For tissues with multiple replicates, only the experiment with the highest read count was retained. As with the plant ATAC-seq data, Tn5 shifting was performed using the `-ATACshift` option from `alignmentSieve` [87], followed by `bamCoverage` [87] to generate a normalized CPM BigWig track at 1-bp resolution, representing the Tn5 insertion profile across the genome. Similarly, for RNA-seq tracks, `bamCoverage` [87] was used to generate CPM BigWig tracks at 1-bp resolution.

**Tomato ATAC- and RNA-seq Tracks.** For tomato (*S. lycopersicum*), ATAC-seq and RNA-seq data were sourced from a study of regulatory regions during fruit development [93]. The tissues included: leaf, fruit (3–10 days post-anthesis), and anthesis fruit. Raw data were obtained from the NCBI SRA (BioProject PRJNA1228166). Data processing followed the same pipelines and quality filters as for chicken and plant experiments. When multiple replicates were available for a tissue, the replicate with the highest read count was retained. ATAC-seq reads were Tn5-shifted using the `-ATACshift` option, and both ATAC-seq and RNA-seq tracks were generated as normalized CPM BigWig files at 1-bp resolution using `bamCoverage`.

**Maize, Rice, and Tomato Ribo-seq tracks.** Plant protein abundance data were derived from Ribo-seq assays for three species: *Z. mays*, *O. sativa*, and *A. thaliana*. The experiments on these plant species were conducted across multiple tissues and developmental stages. Raw data for all species were accessed via the NCBI SRA under the BioProject accession number PRJNA637713.

Each Ribo-seq experiment was subsequently processed using the following pipeline:

1. FASTQC [88]: to assess the quality of raw sequencing reads.
2. Trim Galore [86] with the options `-q 20 -phred33 -stringency 3 -length 20 -e 0.1`: to remove adapter sequences and low-quality bases from the reads.
3. Bowtie2 [83]: to deplete reads mapping to rRNA and tRNA, retaining only putative mRNA reads.
4. HISAT2 [94] with default parameters: to align ncRNA-depleted reads to the reference genome.
5. Samtools [84] with the options `view -bS -q 30`: to filter for high-quality alignments.

After running the pipeline, only experiments that passed FASTQC quality filters, had an alignment rate  $\geq 0.5$ , and contained more than 2 million reads were retained. As with the ATAC-Seq and RNA-Seq tracks, `bamCoverage` [87] was used to generate a normalized CPM BigWig track at 1-bp resolution for all Ribo-seq experiments that passed quality filters.

#### 4.8.2. Models Evaluated

Table 9 summarizes all baseline models included in our evaluation. We distinguish between two broad model classes. First, *from-scratch convolutional baseline architectures*—represented by the bpNet and SpliceAI families—are trained solely on the downstream prediction task and do not involve any form of self-supervised pretraining. In this context, SpliceAI and bpNet refer strictly to the convolutional or dilated-convolutional architectural backbones originally introduced for profile prediction or splice-site detection, without any associated pretraining corpus. Second, we evaluate a set of *pretrained language-model (LM)-based foundation models*, including NTv2, HYENADNA, CADUCEUS/ PLANTCAD2 (SMALL), and EVO2. These models rely on large-scale self-supervised training—either masked nucleotide modeling or autoregressive next-token prediction—performed on multi-species, human, or plant genomes before downstream task adaptation. Together, the two classes span CNNs, dilated-CNNs, state-space models, long-convolution operators, and Transformer architectures, providing a comprehensive sampling of contemporary genomic sequence modeling approaches.

**Convolutional Baseline Architectures.** For the convolutional baselines, we evaluate two variants of the bpNet model, BPNET (SMALL) and BPNET (LARGE), and two variants of the SpliceAI model, RESIDUAL CNN (SMALL) and RESIDUAL CNN (LARGE).

BPNET (SMALL) is the original convolutional profile-prediction architecture introduced by Avsec et al. [6]. BPNET (LARGE) follows the hyperparameter configuration of ChromBPNet [34], a model designed to deconvolve assay-specific biases and learn regulatory sequence determinants of chromatin accessibility. In our work, we use only the bpNet backbone defined in ChromBPNet—excluding the auxiliary noise-prediction head—and adopt its architectural scaling, which expands the original bpNet from approximately 100k parameters to 6M while maintaining its 2,114 bp input and 1 kb output profile resolution.

RESIDUAL CNN (SMALL) corresponds to the original dilated-CNN architecture of Jaganathan et al. [12], which employs a 32-layer dilated convolutional stack to model up to 10 kb of flanking sequence and accurately predict splice donor and acceptor sites directly from primary sequence. We additionally include RESIDUAL CNN (LARGE), the 44M-parameter variant used in Segment-NT [37], which extends the receptive field and model capacity and has shown strong performance on long-context genomic annotation tasks. As all four baselines are fully convolutional architectures, their parameter counts are independent of the input sequence length.

Name	Model class	Architecture	# Params	Seq. length	Pre-training
BPNET (SMALL)	From-scratch CNN	bpNet CNN	100k	1k	–
BPNET (LARGE)	From-scratch CNN	ChromBPNet CNN	6M	2k	–
RESIDUAL CNN (SMALL)	From-scratch CNN	SpliceAI dilated CNN	700k	10k	–
RESIDUAL CNN (LARGE)	From-scratch CNN	SpliceAI dilated CNN	44M	30k	–
NTv2	Pretrained LM	Transformer	500M	12k	MLM
HYENADNA	Pretrained LM	Hyena long-convolution	7M*	1M	Next-token
CADUCEUS	Pretrained LM	RC-equivariant SSM	7M	131k	MLM
PLANTCAD2 (SMALL)	Pretrained LM	Mamba2 SSM (plant)	88M	8k	MLM
Evo2 (1B)	Pretrained LM	StripedHyena2 hybrid	1B	8k	Mixed

**Table 9 | Summary of the baseline models evaluated.** “From-scratch” models correspond to supervised CNN or dilated-CNN architectures trained directly on the downstream task, while “Pretrained LM” models denote genomic foundation models obtained through large-scale masked or autoregressive self-supervision prior to fine-tuning. MLM: masked language modeling. \*For HyenaDNA we used the largest HuggingFace model (*LongSafari/hyenaDNA-large-1m-seqlen-hf*); while in HuggingFace it is mentioned 54M parameters, we obtain 6.6M with `sum(p.numel() for p in model.parameters())`, similar to what the paper reports [22].

**Foundations Models.** We evaluate five pre-trained model baselines, chosen to represent the main inductive biases currently explored in genomic foundation modeling: NTv2, HYENADNA, CADUCEUS/ PLANTCAD2 (SMALL) and Evo2. These models differ in architectural principles (Transformer attention, reverse-complement-equivariant state-space models, long-convolution operators), pretraining corpora (multi-species, human, plant-specialized), and objectives (masked modeling or autoregressive prediction). Table 9 summarizes their parameterization, taxonomic scope, maximum supported context length, and pretraining objectives. Below, we outline the core architectural features and pretraining settings of each model.

NTv2 [21] is a multi-species masked DNA language model based on an encoder-only Transformer with 6-mer tokenization. Pretrained on a broad multi-species genome corpus, it provides contextualized representations that transfer effectively across organisms.

HYENADNA [22] employs the Hyena operator—implicit long convolutions with hierarchical gating—to model up to 1 mb sequences with subquadratic complexity. It is pretrained autoregressively on human genomic data, providing a complementary pretraining signal relative to masked models. We used the largest HYENADNA model available.

CADUCEUS [26] introduces reverse-complement-equivariant modeling through BiMamba and MambaDNA blocks, enforcing strand-consistent representations while enabling efficient long-range reasoning over 131 kb contexts. It is pretrained on human (hg38) genomes with a masked modeling objective.

PLANTCAD2 (SMALL) adapts the Caduceus architecture for plant genomes by replacing Mamba blocks with Mamba2 modules [38], improving memory efficiency for long contexts. It is pretrained on an angiosperm-wide corpus (65 species) with an 8 kb masked nucleotide modeling objective and serves as a domain-specialized baseline for plant datasets. We focus on the *Small* variant of the model, as its 88M parameters already places it in a comparable regime to our NTv3 100M, allowing a more balanced comparison in terms of model capacity.

Evo2 [28] is a large-scale genomic foundation model based on the StripedHyena 2 architecture, which combines convolutional, gating, and implicit long-range operations to efficiently model very long DNA sequences. It is pretrained autoregressively on a multi-kingdom corpus comprising billions of nucleotides from bacteria, archaea, and eukaryotes, using a next-nucleotide prediction objective. The model supports context windows up to 1 mb, capturing distal regulatory and structural dependencies, and provides biologically meaningful representations suitable for a wide range of downstream genomic tasks. We focus on the 1B-parameter variant, which is closest in scale to NTv3-650M, offering a balanced trade-off between capacity and computational cost.

#### 4.8.3. Supervised Training Pipeline

This section describes the supervised fine-tuning protocol used to train all baseline models listed in Table 9 on the downstream tasks defined by the NTv3 BENCHMARK (Table 8). The procedure is designed to ensure that all models—regardless of architecture or pretraining strategy—are evaluated under comparable optimization

budgets, data-processing conventions, and selection criteria.

**Supervised Training Regime, Prediction Heads, and Training Objectives.** All models—whether pretrained or trained from scratch—are trained for **10 epochs**, where one epoch corresponds to a full pass over the species-specific training genome. Validation is performed every **half-epoch**. The checkpoint with the highest validation score is then evaluated on the held-out test set. The exact token budget and the size of the validation and test set are provided in Supplementary Table 14. We fine-tune all model parameters.

The NTV3 BENCHMARK defines two supervised task families, each associated with a distinct output format, prediction head, and evaluation metric:

- *Functional-regulatory prediction (regression)*. Includes chromatin accessibility, histone marks, PRO-cap, eCLIP, RNA-seq, and Ribo-seq. Predictions are produced through a regression head (layer normalization + linear projection with `softplus`). Model selection and final evaluation both use the *mean Pearson correlation* across tracks.
- *Genome-annotation prediction (classification)*. Includes the four universal labels: `exon`, `intron`, `splice_acceptor`, and `start_codon`. Predictions are produced through a linear projection yielding per-position logits. Model selection and final evaluation both use the *Matthews Correlation Coefficient (MCC)*.

**Training frameworks and checkpoints.** Different model families require different execution frameworks, reflecting their native implementations, but all share identical preprocessing, batching, loss definitions, and metric computation.

*NTv3 and convolutional baselines.* These architectures are trained using a JAX (Flax-NNX) implementation that provides functional model definitions, just-in-time compilation via `jax.jit`, and distributed data-parallel execution using `pjit` and `shard_map`. NTV3 models are initialized from the pretrained checkpoints described in Section 4.2 and subsequently fine-tuned on the supervised tracks described in Section 4.3. For fine-tuning on species present in post-training we use the respective species token as input, while for new unseen species we use the mask token. Unless otherwise indicated, reported results correspond to the final post-trained NTV3 checkpoints.

*Other pretrained models.* These architectures are trained using a PyTorch implementation with DeepSpeed ZeRO-2 optimization and `bfloat16` precision. Only the prediction head is replaced; the pretrained backbone is left unchanged. We use publicly released checkpoints hosted on Hugging Face:

- InstaDeepAI/nucleotide-transformer-500m-multi-species for NTV2;
- LongSafari/hyenadna-large-1m-seqlen-hf for HYENADNA;
- kuleshov-group/caduceus-ph\_seqlen-131k\_d\_model-256\_n\_layer-16 for CADUCEUS;
- kuleshov-group/PlantCAD2-Small-124-d0768 for PLANTCAD2 (SMALL).

*Evo2.* Evo2 models are trained using NVIDIA’s `bionemo-framework` [95], which provides native support for the Evo2 architecture and NeMo2 checkpoint format. All experiments use the `evo2/1b-8k-bf16:1.0` model, a BF16-optimized fine-tuned variant of `evo2/1b-8k:1.0`. The underlying base model (`arcinstitute/savanna_evo2_1b_base`) contains 1B parameters and was selected because its scale is closest to NTV3–650M while remaining computationally tractable. Preliminary experiments indicated improved optimization stability with the BF16 variant.

**Training procedure.** To ensure alignment between nucleotide positions and model outputs, model-specific special tokens are removed from the final hidden states before the prediction head is applied. CADUCEUS and PLANTCAD2 (SMALL) require removal of the EOS token; HYENADNA requires removal of both CLS and EOS tokens; NTV2 requires removal of the CLS token.

All models use an effective batch size of 32 implemented via gradient accumulation. For all architectures except Evo2, optimization uses AdamW with weight decay 0.01 and default momentum parameters ( $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ), and the same modified polynomial decay learning rate schedule used for pretraining and post-training: we use an initial learning rate of  $1 \times 10^{-5}$  and a 3% of training tokens budget linear warmup increasing to a peak learning rate of  $5 \times 10^{-5}$  before decay. CNN-based models (BPNet and SpliceAI) use a higher learning rate of  $1 \times 10^{-3}$ . Following the post-training methodology, we compute losses only on the center 37.5% of sequences. Functional track regression tasks use a Poisson-multinomial loss with a shape-loss coefficient of 5.0, and genome annotation classification tasks use a weighted focal loss with  $\gamma = 2.0$ . DeepSpeed training uses ZeRO-2 with `bfloat16` precision.



Evo2 follows the constraints of the bionemo-framework: Adam (not AdamW), BF16 parameter storage with FP32 gradient accumulation, and gradient clipping with a maximum norm of 1.0, mirroring the Evo2 pretraining configuration.

## 4.9 Evaluation on Gene-level Downstream Tasks

### 4.9.1. Tasks

**Gene-level Expression in Plants.** Plant data on gene expression was based on that compiled for the Plant Genomic Benchmark presented in the Agronomic Nucleotide Transformer (AgroNT) study [39]. Unlike the functional track prediction tasks, this task aims to predict gene expression profiles across tissues at the gene level, i.e., producing predictions per gene rather than per base pair. Briefly, the dataset was composed on bulk RNA-seq data from four plant species, namely *A. thaliana*, *O. sativa*, *S. lycopersicum*, and *Z. mays* across multiple tissues, including a variety of organs and developmental tissues. Unlike the original study, which used DESeq2-normalized counts, our downstream task relied on transcripts per million (TPM). To compute TPM values, we used the same pre-computed gene counts and applied gene length correction based on collapsed exon lengths for each gene. TPM values were then log-transformed using a pseudo-count of 1 to avoid taking the logarithm of zero.

For model training, we extracted promoter-proximal sequences centered on annotated transcription start sites (TSS). To evaluate the benefit of providing a larger genomic context, we constructed seven versions of each promoter sequence by extending the flanking regions symmetrically around the TSS. To enable comparison with previously published methods, we used the following flanking lengths: 256 bp, 3 kb, 4 kb, 16 kb, 65 kb, and 131 kb. The total input sequence length for each version is therefore twice the respective flanking length. To prevent data leakage, promoter-proximal sequences were assigned according to the homology-based splits described above. The training, validation, and test sets comprised 95%, 5%, and 5% of the data, respectively.

For *Z. mays*, models were trained on all context sizes (512 bp, 6 kb, 8 kb, 32 kb, 131 kb, and 262 kb). For *A. thaliana*, *O. sativa*, and *S. lycopersicum*, training was restricted to sequence lengths of 6 kb, 8 kb, and 131 kb.

**Gene-level Protein Abundance in Plants.** Plant protein abundance data were obtained from Ribo-seq assays using the dataset and pipeline described above. As in the gene-level expression task, protein abundance was formulated as a gene-level prediction problem. Specifically, protein abundance values were derived using StringTie [96] with the options `-e -A`. These options enabled transcript-level quantification in TPM from the high-quality alignments and produced per-gene abundance estimates based on the reference annotation, providing a proxy for protein synthesis. Models were fine-tuned using a regression objective, with input sequences constructed using the same TSS-centered flanking regions used for the gene expression task. Protein abundance prediction was evaluated across all flanking-region sizes: 512 bp, 6 kb, 8 kb, 32 kb, 131 kb, and 262 kb.

### 4.9.2. Evaluation procedure

NTv3 models were compared with AgroNT 1-billion parameter model [39] on these tasks. Training used an effective batch size of 16, the Adam optimizer with a fixed learning rate of  $5 \times 10^{-5}$  and no weight decay. Training proceeded for up to 200,000 update steps with early stopping based on validation performance. Validation was performed every 2,000 steps on the complete validation set. Early stopping was triggered if validation performance did not improve for 10 consecutive validation evaluations. All experiments used a fixed random seed of 0 for reproducibility. Final model evaluation was conducted on the held-out test set using the checkpoint with the best validation performance.

The post-trained model uses species-specific tokens in Adaptive Layer Normalization (AdaLN) layers for conditioning. For *Solanum lycopersicum*, which was not included in the post-training dataset, we used the `<mask>` conditioning token to enable generalization to unseen species.

## 4.10 Model Interpretation Analysis

To understand the regulatory mechanisms captured by NTv3, we leveraged four complementary interpretability

modalities: (i) predicted functional tracks, which report regulatory activity at base-pair resolution across thousands of assays; (ii) predicted genome annotations, which denote gene-structural and regulatory elements at base-pair resolution; (iii) gradient-based attribution maps, which identify at base-pair resolution the sequence positions most influential for the model’s predictions; and (iv) transformer attention maps, which summarize how the model integrates information across long genomic distances at a resolution determined by the model’s downsampling architecture. Together, these outputs support multi-scale interpretation of regulatory logic, enabling analysis from individual nucleotides to enhancer elements and full gene loci. To note, all interpretability results presented in the main text and described below were obtained using the NTv3 650M post-trained model (131 kb context); for this configuration, the attention maps described in (iv) have a resolution of 128 bp per token, corresponding to seven downsampling stages ( $131,072 \text{ bp input} \div 1,024 \text{ tokens}$ ).

#### 4.10.1. Interpretability Methods

**Predicted Functional Tracks.** Functional track predictions are cell-type and assay specific. For each input sequence, the functional-track module produced per-base predictions for all species-matched regulatory assays. Consistent with the post-training configuration, supervision for functional tracks is applied only to the central 37.5% of the input sequence. Accordingly, all analyses used predictions extracted from this supervised region to ensure that every interpreted position had full receptive-field coverage.

Functional-track predictions were obtained directly from the model output tensor of shape (`sequence length`, `number of tracks`) without additional smoothing or normalization. For visualization (e.g., Fig. 6), predictions for assays such as K562 DNase-seq, K562 GATA1 ChIP-seq, K562 CAGE-seq, and GTEx RNA-seq were retrieved by indexing into this tensor using the track metadata.

**Predicted Genome Annotations.** Genome annotation predictions are cell-type agnostic. The genome-annotation head produced a probability vector over 21 annotation classes at each position. These annotations can be viewed in continuous form (pre-thresholded probabilities; e.g., Fig. 6J) or in binarized form by thresholding softmax probabilities at  $p \geq 0.5$  to produce genome annotation tracks (e.g., Fig. 6B). These labels were used for visualization and qualitative interpretation of regulatory structure (e.g., promoters, exons, introns, splice sites, enhancers; see Supplementary Note B.7.1 for discussion of isoform-related annotation ambiguity).

**Gradient-Based Attribution Maps.** Attribution maps are cell-type specific, as they are computed with respect to a chosen functional track. Gradient-based attribution provides a first-order, base-pair-resolved estimate of how NTv3’s predictions depend on the underlying DNA sequence. We employ gradient-based saliency maps [97], which quantify the local sensitivity of a predicted regulatory signal to infinitesimal changes at each input base. In regulatory genomics this is well motivated: TFBSs, splice junctions, and other *cis*-regulatory elements often depend on a small number of informative nucleotides, such that single-base substitutions can meaningfully alter regulatory output. As a result, large-magnitude gradients arise at positions with high predictive influence, enabling identification of the motif-level sequence features used by the model.

To compute these attributions, we differentiated model outputs with respect to the one-hot encoded DNA input [98, 99]. Because NTv3 represents DNA using an 11-token vocabulary (A, C, G, T, N, and special tokens), gradients cannot be taken with respect to discrete token IDs. Instead, we used a differentiable embedding override: for a sequence of length  $L$ , the one-hot matrix

$$x \in \{0, 1\}^{L \times 11}$$

was multiplied by the learned embedding matrix to produce continuous token embeddings. These embeddings were supplied directly to the model, ensuring that  $\partial y / \partial x$  is well-defined and enabling gradients to flow to every position.

For each selected functional track, the model outputs logits of shape (`batch`, `sequence length`, `number of tracks`). To form a scalar suitable for differentiation, we aggregated the logits over sequence positions and (if applicable) over a subset of tracks, followed by a log transform for numerical stability:

$$y = \log \left( \sum_{i \in \mathcal{W}} \sum_{t \in \mathcal{T}} \text{logits}_{i,t} + \varepsilon \right), \quad \varepsilon = 10^{-6}.$$

Here,  $\mathcal{W}$  denotes the set of sequence positions included in the supervised output window (e.g., the central

49,152 bp for functional tracks), and  $\mathcal{T}$  denotes the subset of functional-track indices used for the attribution calculation.

We then computed

$$A = \frac{\partial y}{\partial x},$$

yielding an attribution tensor  $A \in \mathbb{R}^{L \times 11}$  encoding the local sensitivity of the aggregated prediction to perturbations at each input token.

For base-pair-resolution interpretation, we retained only the A/T/C/G channels (vocabulary indices 6–9), discarding non-nucleotide channels. These channel-resolved hypothetical gradients correspond to counterfactual substitutions at each base and form the basis for motif discovery, local sequence-grammar analysis, and allele-specific interpretation.

**Transformer Attention Maps.** Attention maps are cell-type agnostic, reflecting the model’s internal context aggregation independent of any specific output track. Transformer self-attention provides a complementary interpretability modality by revealing how NTv3 integrates information across distal regions of the input sequence. For genomic inputs, attention scores measure how sequence information at one downsampled token (128 bp per token; see above)—influences the contextual representation of other tokens across the locus. Because these scores are computed after substantial sequence compression, they offer a coarse-grained but informative view of long-range dependencies not captured by base-resolution gradient methods.

Multi-head self-attention computes pairwise similarity between token representations via learned query–key interactions, producing attention weights that modulate how information at one position influences the contextual encoding of others. Although attention weights are not direct measures of biological causality, they provide a useful diagnostic of the long-range dependencies the model has learned. We therefore use attention maps as a post-hoc interpretability modality to examine which distal regions NTv3 emphasizes when forming its predictions.

To analyze these dependencies, we extracted multi-head self-attention weights from the final transformer block and averaged them across heads to obtain a single  $L' \times L'$  attention matrix (with  $L' \approx 1024$  for 131 kb inputs). This follows prior work demonstrating that transformer attention can highlight structured token–token interactions in language models [100] and genomic models such as Borzoi [7] (see Supplementary Note B.7.1 for discussion of layer- and head-specific analyses).

To standardize interpretation and mitigate asymmetries in directional attention, matrices were symmetrized as  $(A + A^T)/2$ . Lower-triangular entries were masked, and matrices were visualized using a 45° rotated triangular layout, which highlights long-range interaction bands as pyramidal structures. For variant-centered or enhancer-focused analyses, attention was cropped to windows surrounding the locus of interest and aligned to functional-track outputs, genome-annotation labels, and attribution maps.

**Interpretability Output Alignment.** All interpretability outputs were aligned to a shared genomic coordinate system. Base-pair coordinates map directly to the supervised window of functional-track and annotation outputs. Token-resolution outputs (e.g., attention maps) were aligned by converting token indices to base-pair positions using the fixed downsampling factor (128 bp per token) as described above. This alignment enabled direct comparison across modalities: attention maps consistently displayed strong diagonal structure reflecting local context aggregation, while off-diagonal features highlighted long-range dependencies including enhancer–promoter interactions and splice-site coordination. These coarse-resolution patterns complemented base-pair-level attribution maps by identifying which distal regions NTv3 attends to when forming its predictions.

#### 4.10.2. Interpretation Tasks

**Differential Comparison Across Modalities.** To compare reference and alternative sequences in both the VEP and enhancer–promoter tasks, we computed differential profiles across prediction modalities produced by NTv3, including functional tracks, genome annotations, attribution maps, and attention maps. Because different modalities exhibit distinct numerical properties (non-negativity, discreteness, symmetry, or sign ambiguity), each required a modality-appropriate transformation while maintaining interpretability and compatibility across analyses.

**Predicted functional tracks.** For RNA-seq, DNase-seq, CAGE-seq, and ChIP-seq-like prediction heads, which produce non-negative per-base signals,  $\log_2\text{FC}$  was computed elementwise as

$$\log_2\text{FC}_i = \log_2 \left( \frac{\text{alt}_i + \varepsilon}{\text{ref}_i + \varepsilon} \right),$$

with a numerical stabilizer  $\varepsilon = 10^{-6}$ . Positive values correspond to increased predicted activity under the alternative allele, and negative values correspond to reduced activity. This formulation matches established genomic signal-comparison pipelines used for model interpretation.

**Predicted genome annotations.** For genome-annotation predictions, which produce per-position logits over annotation categories, we computed a discrete fold difference rather than  $\log_2\text{FC}$ . Logits were converted to presence probabilities via softmax and thresholded at  $p \geq 0.5$  to yield binary presence masks  $B^{\text{ref}}, B^{\text{alt}} \in \{0, 1\}^{L \times K}$ , where  $K$  denotes the number of annotation categories. The fold difference was computed as

$$\Delta_{i,k} = B_{i,k}^{\text{alt}} - B_{i,k}^{\text{ref}},$$

yielding values in  $\{-1, 0, +1\}$ :  $+1$  indicates an annotation gained under the alternative allele,  $-1$  indicates a loss, and  $0$  indicates no change.

**Gradient-based attribution maps.** To compare attribution maps between reference and alternative alleles, we computed  $\log_2\text{FC}$  after shifting both matrices by their joint minimum to ensure positivity. The resulting  $L \times 4$   $\log_2\text{FC}$  matrix (positions  $\times$  nucleotide channels) is difficult to interpret directly; we therefore collapsed it to a one-dimensional bidirectional profile by separately summing positive and negative values across channels at each position:

$$\text{profile}_i = \sum_c [\log_2\text{FC}_{i,c}]^+ + \sum_c [\log_2\text{FC}_{i,c}]^-,$$

where  $[\cdot]^+ = \max(\cdot, 0)$  and  $[\cdot]^- = \min(\cdot, 0)$ . This bidirectional aggregation preserves the magnitude of both gains and losses at each position, enabling visualization of positions with increased (positive) or decreased (negative) attribution under the alternative allele. This operation is denoted  $\sum \log_2\text{FC}$  in Fig. 6J.

**Transformer attention maps.** Let  $A^{\text{ref}}$  and  $A^{\text{alt}}$  denote symmetrized attention matrices for the reference and alternative sequences, respectively. We computed  $\log_2\text{FC}$  attention maps using

$$\log_2\text{FC}(i, j) = \log_2 \left( \frac{A_{ij}^{\text{alt}} + \varepsilon}{A_{ij}^{\text{ref}} + \varepsilon} \right), \quad \varepsilon = 10^{-8}.$$

The resulting matrices highlight long-range interactions whose strength changes between alleles, with positive values indicating gains in attention and negative values indicating attenuation.

**Enhancer–Promoter (EP) Task.** To evaluate whether NTv3 captures experimentally validated enhancer–promoter relationships, we analyzed the human *HBE1* locus, a well-characterized erythroid regulatory region containing five DNase hypersensitive elements (HS1–HS5). These enhancers have been extensively assayed in erythroid cells, and their relative strengths have been quantified using massively parallel reporter assays and CRISPR-based perturbations [47, 48]. This locus therefore provides a biologically grounded test case for assessing long-range sequence–to–function modeling.

All analyses were performed on a 131,072 bp input window centered on the *HBE1* interval (chr11:5,202,859–5,333,931; hg38). Within this window, enhancer positions were defined using the hypersensitive-site coordinates reported by Agarwal [47]:

- HS1: 5,275,890–5,276,090
- HS2: 5,280,660–5,280,860
- HS3: 5,284,690–5,284,890
- HS4: 5,288,180–5,288,380

- HS5: 5,291,330–5,291,530

To contextualize enhancer activity, we used matched erythroid functional assays from the post-training dataset: K562 DNase-seq (ENCSR000EKN), K562 GATA1 ChIP-seq (ENCSR000EFT), K562 CAGE-seq (CNhs11250). These provide chromatin accessibility, TF binding, and promoter-proximal transcriptional output in the same cellular context in which HS1–HS5 enhancer activities were experimentally validated.

For the HBE1 enhancer–promoter analysis, attribution maps were computed on the reference sequence with respect to the K562 CAGE track. To evaluate enhancer function, we performed *in silico* perturbation experiments in which enhancer sequences were systematically ablated and restored. Dinucleotide-preserving shuffling was used to ablate regulatory content while maintaining local sequence composition, using the HS1–HS5 enhancer coordinates defined above.

A fully ablated background sequence was first constructed by shuffling all five enhancer elements simultaneously. Enhancer sufficiency was then assessed by restoring individual enhancers one at a time while keeping the remaining enhancers shuffled. For each enhancer, five independent shuffled replicates were generated for both the fully ablated background and each single-enhancer restoration condition.

All perturbed sequences were passed through NTv3 to extract predicted K562 CAGE signal. Track outputs were restricted to the supervised central 49,152 bp region of the output sequence.

To quantify enhancer sufficiency, we measured the change in predicted promoter-proximal CAGE signal when a single enhancer was preserved relative to the fully ablated background. Specifically, we extracted a 384 bp promoter window centered at the HBE1 transcription start site (chr11:5,269,945; hg38;  $\pm 192$  bp). Within this window, we computed the  $\log_2$  fold-change at each base position and then averaged these values across positions to obtain a single enhancer-effect score. More positive values correspond to greater enhancer sufficiency, reflecting the ability of that element alone to rescue promoter activity above the fully ablated baseline. Predicted sufficiency scores were: HS2 ( $0.366 \pm 0.014 \log_2\text{FC}$ ), HS1 ( $0.196 \pm 0.011$ ), HS4 ( $0.047 \pm 0.009$ ), HS3 ( $0.037 \pm 0.037$ ), and HS5 ( $0.012 \pm 0.013$ ). This perturbation-based sufficiency metric parallels experimental measurements reported in massively parallel enhancer assays [47].

**Variant Effect Prediction (VEP) Task.** To evaluate whether NTv3 captures the local regulatory disruptions caused by SNVs, we performed a variant effect prediction (VEP) analysis using expression quantitative trait loci (eQTLs) from the Genotype–Tissue Expression (GTEx) project. We focused on variants annotated for *Whole Blood* and used GTEx-reported allelic effects to define pathogenic and benign sets. A total of 3,792 variants were retained after mapping to the hg38 reference genome. Variants were excluded if the 131,072 bp centered window extended beyond chromosome boundaries.

Genomic coordinates were intersected with the GENCODE v44 annotation set to provide transcript- and splice-aware metadata for downstream interpretation. These annotations were used solely for visualization of gene structure and regulatory elements (e.g., exon boundaries, splice sites, promoter intervals) and were not used to filter variants or adjust model outputs.

For each variant, we extracted a 131,072 bp window centered on the SNV. Two allele-specific input sequences were constructed per variant: a *reference* sequence containing the hg38 allele at the center position, and an *alternative* sequence in which the reference base was replaced by the SNV’s alternative allele. All sequences, annotations, and predictions were handled in the hg38 coordinate system.

Variant effects were quantified using the GTEx Whole Blood RNA-seq prediction head from the functional-track module (GTEx-114MK-0002-SM-EZ6M9.1, GTEx-1LB8K-0005-SM-DIPED.1, GTEx-1OKEX-0006-SM-DKPQ2.1). These tracks correspond to bulk RNA expression aggregated across donors and matched to the training metadata. For each allele-specific sequence, NTv3 produced a tensor of predicted RNA-seq values across the supervised central region of the output. These predictions were used directly for downstream attribution-based analysis. No post-hoc smoothing, normalization, or additional transformations were applied. This preprocessing pipeline yielded matched reference and alternative predictions for every variant, enabling computation of base-pair-resolution attribution maps and allele-specific divergence metrics.

To provide interpretable summaries of attribution signals, we extracted windows around each SNV and retained the A/T/C/G channels. These nucleotide-resolved gradients were visualized using the Logomaker package to generate hypothetical-gradient sequence logos [51, 101], preserving gradient directionality and enabling motif-level inspection.



This workflow identified fine-scale *cis*-regulatory determinants including ETS/ELK1 motifs, ZNF784 motifs, IRF-family motifs, and canonical splice donor/acceptor patterns. Comparing reference and alternative attribution profiles enabled classification of mechanistic changes into motif ablation, motif creation, or distributed multi-motif effects.

To translate these model outputs into actionable variant prioritizations, we developed a multi-stage ranking framework that integrates (i) quantitative divergence metrics across modalities, (ii) genomic context and functional annotations, and (iii) mechanistic synthesis with literature validation. For a detailed description of this framework—including distance-metric selection, window-length effects, Ensembl VEP integration [54], and a modular Python toolkit for visualization—see Supplementary Note B.7.2.

## 4.11 Fine-tuning NTv3 for Conditioned Sequence Generation

### 4.11.1. Model Training

To enable conditional diffusion-based sequence generation, we fine-tuned NTv3 with masked diffusion language modeling (MDLM) objective [55]. At a high level, transitioning from a standard pre-training MLM model to the generative MDLM one entails training with a variable amount of noise, i.e., masked tokens, applied to each sequence in a batch, as opposed to the fixed percentage applied during BERT-style [32] MLM pre-training. We formally introduce MDLM training below.

Let  $\mathbf{x}$  and  $\mathbf{z}$  represent one-hot vectors over some vocabulary  $\mathcal{V}$ , i.e.,  $\mathbf{x}, \mathbf{z} \in \{0, 1\}^{|\mathcal{V}|}$  and  $\mathbf{x}, \mathbf{z} \in \Delta^{|\mathcal{V}|}$ , where  $\Delta^{|\mathcal{V}|}$  is the simplex over  $|\mathcal{V}|$  categories. We denote a full sequence of  $L$  tokens as  $\mathbf{x}^{1:L}$ , with  $\mathbf{x}^\ell$  as the  $\ell^{\text{th}}$  token in the sequence. In MDLM, we use a special [MASK] token and denote its one-hot representation by  $\mathbf{m}$ .

MDLM is derived from the diffusion paradigm for generative modeling [102, 103], initially introduced for continuous signals, and later adapted for discrete data [104]. These models are characterized by a pre-defined corruption process  $q$  that transforms signal into noise and a learned reverse process  $p_\theta$  that is trained to recover the original signal. Specifically, the forward process adds noise to samples drawn from some data distribution  $\mathbf{x} \sim q_{\text{data}}$  to produce latent variables  $\mathbf{z}_t \sim q(\cdot | \mathbf{x})$ , where the subscript  $t \in [0, 1]$  denotes time in the diffusion process. The latents  $\mathbf{z}_t$  become progressively more corrupted as  $t$  goes from 0 to 1, with the fully corrupted final latent variable  $\mathbf{z}_1$  drawn from a pre-defined prior distribution. In MDLM, this prior is equal to the special mask token one-hot  $\mathbf{m}$ . That is, at the end of the forward process, all of the signal from the data sample has been removed, as all tokens have been replaced with the [MASK] symbol. Let  $\text{Cat}(\cdot; p)$  denote a categorical distribution over  $\mathcal{V}$  with probabilities given by  $p \in \Delta^{|\mathcal{V}|}$ . A widely used forward process for discrete signals, is one that interpolates between data and the limiting distribution [104], and we adopt this process for MDLM as well [55, 105–107]:

$$q(\mathbf{z}_t | \mathbf{x}) = \text{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x} + (1 - \alpha_t) \mathbf{m}), \quad (6)$$

where  $\alpha_t := \alpha(t) \in [0, 1]$  denotes the noise schedule, which is monotonically decreasing in  $t$ , with  $\alpha(1) = 0$  and  $\alpha(t) \rightarrow 1$  as  $t \rightarrow 0$ . Here,  $\alpha_t$  represents the fraction of the original signal that remains at time  $t$ , while  $1 - \alpha_t$  corresponds to the fraction replaced by the [MASK] token (i.e., the noise level). For a discrete-time process, we set some number of diffusion steps as  $T$  and let  $t = i/T$  for  $i \in 0, \dots, T$ . Letting  $s < t$  denote some time earlier in the diffusion process, the marginals defined in (6) yield the following form for the true posterior distributions [55]:

$$q(\mathbf{z}_s | \mathbf{z}_t, \mathbf{x}) = \begin{cases} \text{Cat}(\mathbf{z}_s; \mathbf{z}_t) & \text{if } \mathbf{z}_t \neq \mathbf{m} \\ \text{Cat}(\mathbf{z}_s; \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \mathbf{x} + \frac{1 - \alpha_s}{1 - \alpha_t} \mathbf{m}) & \text{if } \mathbf{z}_t = \mathbf{m}. \end{cases} \quad (7)$$

Importantly, the noising process  $q$  is assumed to factorize independently across the sequence length dimension, i.e.,  $q(\mathbf{z}_t^{1:L} | \mathbf{x}^{1:L}) = \prod_{\ell=1}^L q(\mathbf{z}_t^\ell | \mathbf{x}^{1:L})$  and  $q(\mathbf{z}_s^{1:L} | \mathbf{z}_t^{1:L}, \mathbf{x}^{1:L}) = \prod_{\ell=1}^L q(\mathbf{z}_s^\ell | \mathbf{z}_t^\ell, \mathbf{x}^{1:L})$ .

The goal of diffusion modeling is to train a denoising process  $p_\theta$  that can generate samples that resemble those from the original data distribution. Ideally, we could perfectly recover the exact posterior in (7), however, when generating samples, we do not have access to the true data sample  $\mathbf{x}$  which is being generated. Therefore, a widely-used parameterization for  $p_\theta$  is to define a denoising network  $\mathbf{x}_\theta \in \Delta^{|\mathcal{V}|}$  that takes in noisy (partially masked) inputs and returns a probability over the clean token for each position. We then plug this prediction



into (7). This gives the following form for the reverse process [55]:

$$p_{\theta}(\mathbf{z}_s | \mathbf{z}_t) = \begin{cases} \text{Cat}(\mathbf{z}_s; \mathbf{z}_t) & \text{if } \mathbf{z}_t \neq \mathbf{m} \\ \text{Cat}(\mathbf{z}_s; \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \mathbf{x}_{\theta} + \frac{1 - \alpha_s}{1 - \alpha_t} \mathbf{m}) & \text{if } \mathbf{z}_t = \mathbf{m}. \end{cases} \quad (8)$$

As above, we assume that this process factorizes along the sequence length:  $p_{\theta}(\mathbf{z}_s^{1:L} | \mathbf{z}_t^{1:L}) = \prod_{\ell=1}^L p_{\theta}(\mathbf{z}_s^{\ell} | \mathbf{z}_t^{\ell})$ . To mimic the behavior of the forward process, two additional restrictions can be placed on the denoising model: 1) we do not place any probability mass on the mask position:  $\langle \mathbf{x}_{\theta}, \mathbf{m} \rangle = 0$  and 2) for any tokens that are not noised, we simply ‘copy them over’:  $\langle \mathbf{x}_{\theta}, \mathbf{z}_t \rangle = 1$ , if  $\mathbf{z}_t \neq \mathbf{m}$ , where  $\langle \cdot, \cdot \rangle$  denotes an inner product.

The denoising diffusion model is trained to minimize a variational upper bound on the log-likelihood of the training dataset. Crucially, taking  $T \rightarrow \infty$  produces a continuous-time formulation whose variational bound simplifies to the following [55, 106, 107]:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x} \sim q_{\text{data}}} \mathbb{E}_{\mathbf{z}_t^{1:L} \sim q(\cdot | \mathbf{x}^{1:L})} \sum_{\ell=1}^L \int_{t=0}^{t=1} \frac{-\alpha'_t}{1 - \alpha_t} \log \langle \mathbf{x}_{\theta}^{\ell}(\mathbf{z}_t^{1:L}), \mathbf{x}^{\ell} \rangle dt, \quad (9)$$

where  $\alpha'_t$  denotes the time derivative of the noise schedule. Examining this objective, we see that it represents a weighted average of cross-entropy MLM losses for input sequences with variable amounts of masking rates. Hence, the standard MLM pre-training, which uses only a single, fixed masking rate, provides a useful model to fine-tune using the MDLM objective.

#### 4.11.2. Enhancer Design Use-Case

One of the use cases we are interested in with a generative genomic sequence model is the design of regulatory elements with defined functional properties. In particular, we aim to generate enhancer sequences that drive a specified expression profile within a given promoter or gene context. Achieving this requires a model that can capture both local regulatory sequence pattern and the higher order interactions between promoters and enhancers. Therefore, we developed a generative framework that internalizes regulatory interactions from large scale self supervised pretraining while supporting the incorporation of conditioning signals to specify the desired activity profile during sequence design.

To achieve this goal, NTv3 was fine-tuned on a UMI-STARR-seq dataset generated in *Drosophila* S2 cells in a previous study [5]. The original dataset contains 242,026 samples of 249 bp sequence windows from the dm3 genome, each paired with a quantitative activity measurement obtained with the DSCP and RpS12 core promoters. To place the tentative enhancer sequences in the context of the promoter and reporter gene, we constructed input sequences by inserting the 249 bp sequences into the standard *Drosophila* STARR-seq vector [57], containing either the DSCP or the RpS12 core promoter, and then cropping the resulting constructs to a length of 4,096 bp. By constructing the input enhancers with experimental plasmid sequences, we enabled the model to leverage its knowledge from self-supervised pretraining, and identify promoter-enhancer interaction patterns from the input sequences. For each input sequence, we discretized the activity measurement into five equal width bins spanning the full activity distribution for the corresponding promoter, represented by integer labels 0-4. Setting the bin boundaries by numerical value distance led to imbalanced data distributions, but we didn’t want the condition to rely heavily on the distribution of training samples. This procedure yielded 484,052 sequences of length 4,096 bp, each associated with a discrete activity label defined in its promoter specific sequence context. The train, validation, and test splits followed the partitioning used in the original study.

After training, we conducted two complementary experiments to evaluate the in-context generative capabilities of NTv3. First, we used our model to design enhancers of various activity levels in context of DSCP and RpS12 promoters. This experiment tests whether the model can generate sequences with a desired activity profile and therefore captures the relationship between sequence composition and regulatory output. Second, we designed enhancers that are only active in one of the two promoters. This is to assess whether NTv3 incorporates promoter-specific context during generation, and to demonstrate that the model can be used with very complex conditioning to achieve a wide range of functional design objectives.

#### 4.11.3. Training Parameters

Adaptive layer norms [77] were used to incorporate the discrete activity labels as conditioning inputs. During training, MDLM masking with a linear noise schedule (i.e.,  $\alpha_t = 1 - t$ ) was applied to the 249 bp region where enhancer sequences are inserted into the vector. NTv3 also included an additional multiclass classification head that predicts the activity level of the input sequence. For 10% of training samples, activity labels are randomly masked, which enables joint supervised training of the classification head and the generative MDLM objective. NTv3 was trained with the AdamW optimizer with weight decay of 0.01 and a modified squared decay learning rate schedule. The initial learning rate was  $5.0 \times 10^{-5}$  with a linear warmup over 3 billion tokens, reaching the peak learning rate of  $1.0 \times 10^{-4}$ , and a subsequent decay to  $5.0 \times 10^{-5}$ . The model was trained with an effective batch size of 32 across 2 GPUs. Exponential moving average (EMA) with a decay rate of 0.9999 was applied, and the final EMA weight were used for all experiments. We found that the use of EMA parameters improved stability and also led to performance improvements over the raw parameters.

#### 4.11.4. Conditioned Enhancer Sequence Generation

When generating sequences with NTv3, we used the guidance algorithm of classifier-free diffusion guidance (CFG) [58] adapted for discrete models [59].

Formally, let  $y$  denote some property or condition of interest. Controlled generation in diffusion models aims to steer each denoising step by sampling from the following unnormalized tempered, conditional distribution:

$$p^\gamma(\mathbf{z}_s | \mathbf{z}_t, y) \propto p(y | \mathbf{z}_s, \mathbf{z}_t)^\gamma \cdot p(\mathbf{z}_s | \mathbf{z}_t), \quad (10)$$

where  $\gamma$  represents a hyperparameter that controls how much we ‘focus’ on the conditioning. Applying Bayes’ rule to the first term on the right hand side of (10) eliminates the need for the classifier  $p(y | \mathbf{z}_s, \mathbf{z}_t)$ :

$$p^\gamma(\mathbf{z}_s | \mathbf{z}_t, y) \propto \left[ \frac{p(\mathbf{z}_s | \mathbf{z}_t, y)}{p(\mathbf{z}_s | \mathbf{z}_t)} \right]^\gamma \cdot p(\mathbf{z}_s | \mathbf{z}_t) = p(\mathbf{z}_s | \mathbf{z}_t, y)^\gamma \cdot p(\mathbf{z}_s | \mathbf{z}_t)^{1-\gamma}. \quad (11)$$

Note that in the first expression on the right of the proportionality symbol in (11), we absorbed the factor of  $p(y | \mathbf{z}_t)^\gamma$  into the proportionality constant. We can model both terms on the right hand side of (11) using denoising diffusion models. In practice, these distributions can be generated using the same model. This is achieved by randomly dropping out or masking out the conditioning during training.

Finally, we can also introduce the notion of negative conditioning, by replacing the unconditional model in (11) with one conditioned on a background or base value  $y_{bg}$ . In this way, we simultaneously steer the denoising distribution towards the condition of interest  $y_{target}$  and away from the base condition. Specifically, we sample from the following distribution:

$$p^\gamma(\mathbf{z}_s | \mathbf{z}_t, y_{target}, \neg y_{bg}) \propto p_\theta(\mathbf{z}_s | \mathbf{z}_t, y_{target})^\gamma \cdot p_\theta(\mathbf{z}_s | \mathbf{z}_t, y_{bg})^{1-\gamma}, \quad (12)$$

This framework allows CFG to flexibly steer generation under multiple constraints, including jointly specifying an activity level and a promoter specific context. For all experiments, we used an inverse temperature parameter  $\gamma = 2.0$  and 50 denoising steps.

For designing enhancers with graded activity with a fixed promoter, we input to the model the sequence vector containing the target promoter, and 249 masked tokens (nucleotides) at the enhancer position. In this experiment, the conditional distribution would be model’s output given desired activity level label. For the unconditional distribution, we provide the same input sequence, but with an masked label for conditions. After each denoising step, we take the output the model provided given the two conditions, and calculate the distribution given the CFG function, and feed the output as the input for both conditions of the next step.

For experiments that design enhancers with promoter-specific activity, we would provide our model with two different input vectors. One with the desired high-activity promoter, and the other with the undesired promoter. For each denoising step, the conditional distribution is calculated by inputting the desired promoter with highest activity level label, while the unconditional distribution with the undesired vector and its highest activity label to push away from. After each denoising step, we apply the CFG to make sure the 249 masked region would be the same while context remain different. Then the two vectors are again inputted back in, denoising for the next step.

#### 4.11.5. In-silico Evaluation

For each enhancer generation task, we identified sequences in the test set that satisfied the same functional conditions according to experimental ground truth. For activity prompted generation, we selected test sequences whose measured activity fell within the desired activity range for the corresponding promoter context. For promoter specific generation, we selected test enhancers that overlapped STARR-seq peaks in only one of the promoter experiments, ensuring that the matched set reflected promoter selective activity. These matched sets provided a direct basis for comparing generated sequences with real sequences under identical functional specifications. Before conducting *in-cellulo* experiments, we used the previously established oracle model DeepSTARR[5] to estimate enhancer activity. For every generated enhancer, we compared the DeepSTARR predicted activity of the proposed new sequence with the predictions for its matched test set.

We also performed sequence profile based in silico analyses, including comparisons of GC content, k mer composition, and motif enrichment between generated sequences and their matched test sets. These analyses demonstrate that NTv3 produces enhancer sequences with biologically realistic properties. In addition, we ran BLAST searches against the dm3 genome on all generated sequences to confirm that NTv3 proposes novel biological sequences rather than reproducing elements from the training data.

---

```

1
2 def linear_noise_schedule(t: float) -> float:
3     # Linear schedule: t=1 fully masked, t=0 fully unmasked
4     return t
5
6 def compute_mdml_posterior(tokens: Array, probs: Array, t: float, dt: float, mask_token_id: int) -> Array:
7     """
8     Compute MDLM posterior for one denoising step.
9
10    - Carry-Over Unmasking: If a latent token is unmasked, copy it over
11    - Masked positions: Interpolate between model prediction and staying masked.
12    - Assign probability mass to the predicted token proportional to the signal gain (signal_s - signal_t),
13      and to the mask token proportional to the remaining noise (1 - signal_s).
14    """
15    # Noise and signal levels
16    noise_t = linear_noise_schedule(t) # t
17    noise_s = linear_noise_schedule(max(t - dt, 0)) # t - dt
18    signal_t = 1 - noise_t # fraction unmasked now ( $\alpha_t$ )
19    signal_s = 1 - noise_s # fraction unmasked next ( $\alpha_s$ )
20
21    # Copy over unmasked tokens
22    unmasked_posterior = one_hot(tokens, vocab_size)
23
24    # For masked tokens, weighted mix of model prediction and mask
25    masked_posterior = probs * (signal_s - signal_t) / (1 - signal_t)
26    masked_posterior[..., mask_token_id] = (1 - signal_s) / (1 - signal_t)
27
28    # Combine based on mask
29    posterior = where(tokens != mask_token_id, unmasked_posterior, masked_posterior)
30    return posterior
31
32 def cfg_guided_logits(model: Model, tokens: Array, condition: Array, gamma: float) -> Array:
33     """
34     Classifier-free guidance logits.
35
36      $\log p_{\gamma}(z_s | z_t, y) \sim \gamma \log p(z_s | z_t, y) + (1-\gamma) \log p(z_s | z_t)$ 
37     """
38    # Unconditional forward pass
39    logits_uncond = model(tokens, condition=None)["logits"]
40    # Conditional forward pass
41    logits_cond = model(tokens, condition=condition)["logits"]
42    # Linear combination in log-space
43    return (1 - gamma) * logits_uncond + gamma * logits_cond
44
45 def generate_sequence_with_cfg(model: Model, init_tokens: Array, condition: Array,
46                               num_steps: int, gamma: float, mask_token_id: int) -> Array:
47     """
48     Generate sequence using MDLM with classifier-free guidance.
49     """
50    tokens = init_tokens
51    dt = 1.0 / num_steps
52
53    # Denoise from t=1 (fully masked) to t=0 (fully unmasked)
54    for t in linspace(1, 0, num_steps + 1):
55        # Compute CFG-guided logits
56        logits = cfg_guided_logits(model, tokens, condition, gamma)
57        probs = softmax(logits, axis=-1)

```

---

---

```

58
59     # Compute MDLM posterior and sample
60     posterior = compute_mdml_posterior(tokens, probs, t, dt, mask_token_id)
61     tokens = categorical_sample(posterior)
62
63     return tokens
64
65
66 def multi_context_cfg_logits(
67     model: Model,
68     tokens_target: Array,
69     tokens_background: Array,
70     target_region: tuple[int, int],
71     background_region: tuple[int, int],
72     gamma: float
73 ) -> Array:
74     """
75     Compute classifier-free guidance logits steering towards target context
76     and away from background context.
77     """
78     # Run model on full contexts
79     full_logits_target = model(tokens_target)["logits"]
80     full_logits_background = model(tokens_background)["logits"]
81
82     # Extract logits only for the generation region
83     logits_target = Extract(full_logits_target, target_region)
84     logits_background = Extract(full_logits_background, background_region)
85
86     # CFG combination: amplify target, suppress background
87     return gamma * logits_target + (1 - gamma) * logits_background
88
89
90 def generate_with_negative_conditioning(
91     model: Model,
92     init_tokens_target: Array,
93     init_tokens_background: Array,
94     target_region: tuple[int, int],
95     background_region: tuple[int, int],
96     num_steps: int = 50,
97     gamma: float = 2.0,
98     mask_token_id: int
99 ) -> tuple[Array, Array]:
100     """
101     Generate a sequence that is active with target context but inactive
102     with background context using MDLM and CFG.
103     """
104     # Ensure regions have the same length
105     assert (target_region[1] - target_region[0]) == (background_region[1] - background_region[0])
106
107     tokens_target = init_tokens_target
108     tokens_background = init_tokens_background
109     dt = 1.0 / num_steps
110
111     for t in linspace(1, 0, num_steps + 1):
112         # Compute CFG-guided logits
113         logits = multi_context_cfg_logits(model, tokens_target, tokens_background,
114                                           target_region, background_region, gamma)
115         probs = softmax(logits, axis=-1)
116
117         # Extract masked region from target
118         sub_tokens = Extract(tokens_target, target_region)
119         posterior = compute_mdml_posterior(sub_tokens, probs, t, dt, mask_token_id)
120         new_tokens = categorical_sample(posterior)
121
122         # Insert updated region into both contexts
123         tokens_target = Insert(tokens_target, target_region, new_tokens)
124         tokens_background = Insert(tokens_background, background_region, new_tokens)
125
126     return tokens_target, tokens_background
127

```

---

#### 4.11.6. Experimental Validation

**Selection of Candidate Generated Sequences.** The activity prediction head of NTv3 was used to filter all generated sequences prior to *in cellulo* evaluation. For experiments aimed at designing enhancers with specified activity levels, we selected the 60 sequences with the highest predicted probability for each target activity bin from 1,024 generated candidates per bin, yielding a total of 600 sequences across five bins and two promoter contexts. For experiments targeting promoter-specific enhancers, we selected 150 sequences that displayed

the largest predicted activity difference between the two promoters from 10,240 generated sequences. As a baseline for sequence design, we randomly generated 512,000 sequences by uniformly sampling over A, T, C, and G, and then used the DeepSTARR [5] oracle model to identify the 150 sequences with the highest predicted activity difference representing the baseline oracle guided selection performance. In total, 900 sequences were selected for experimental evaluation using STARR-seq.

**Generating STARR-seq Libraries.** 300-bp oligonucleotides containing 249-bp candidate sequences were custom synthesized by TWIST and amplified as inserts for oligonucleotide STARR-seq plasmid libraries in two consecutive PCR steps. The first PCR was performed in the linear range with the primer pair (forward: ACACTCTTTCCCTACACGACGCTCTTCCGATCT, reverse: GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT). In the second PCR amplification step, overhangs were added for Gibson cloning using the following primer pair (forward: TAGAGCATGCACCGGACACTCTTTCCCTACACGACGCTCTTCCGATCT, reverse: GGCCGAATTCGTCGAGTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT). Amplified fragments were cloned using Gibson assembly (New England BioLabs, catalog no. E2611S) into the previously published STARR-seq plasmid containing dscp or rps12 promoter (Addgene 71499[57] and Addgene 71504[60]). Oligonucleotide libraries were electroporated into MegaX DH10B electrocompetent bacteria (ThermoFisher C640003) and grown in 2L LB-Amp (Luria-Bertani medium plus ampicillin, 100 µg/ml) each and purified with a Qiagen Plasmid Plus Mega Kit (catalog no. 12981). All libraries were then tagged with a unique molecular identifier (UMI), amplified and sequenced to derive an input count value for the representation of sequences in the plasmid library, as previously described[108].

**Cell Culture.** *Drosophila* S2 cells (ThermoFisher R69007) were grown in Schneider’s Insect medium (Gibco 21720001) with 10% fetal bovine serum (FBS) (Sigma-Aldrich, heat inactivated) and 1% P/S (Gibco 15140122) at 27°C and 0.4% CO<sub>2</sub>. For electroporations, the regular growth of cells was monitored, and cells were seeded at a density of  $3 \times 10^6$ /mL the day before electroporation. For each biological replicate,  $50 \times 10^6$  cells were resuspended in 100 µL of a 1:1 dilution of HyClone MaxCyte electroporation buffer and serum-free Schneider’s and electroporated with 5 µg of the input libraries (see previous section) using the MaxCyte-STX system (“Optimization 1” protocol). Cells were then incubated in DNase I (Worthington Biochemical, 2000 U/mL) for 30 min and harvested for RNA extraction 24 h after addition of culture media for recovery and oligonucleotide library expression. RNA was extracted and processed following the previously described UMI-STARR-seq protocol [108].

**Illumina Sequencing.** Next-generation sequencing was performed at the VBCF NGS facility on a NovaSeq SP platform (150bp paired-end), following the manufacturer’s protocol, using standard Illumina i5 indexes as well as UMIs at the i7 index.

**STARR-seq Data Analysis.** A custom index containing the corresponding sequences was generated using the buildindex function from the Rsubread R package (Liao et al 2019 version 2.10.0). STARR-seq paired-end reads were then aligned using the align function from the same package, with the following parameters: type = “dna”, unique = TRUE, maxMismatches = 2. Then, UMI sequences were retrieved and collapsed as previously described [5].

## Data availability

All data used for NTv3 pre-training, post-training, and downstream fine-tuning were obtained from publicly available resources. The OpenGenome2 pre-training corpus is available via Hugging Face at <https://huggingface.co/datasets/arcinstitute/opengenome2>. Functional genomics tracks were obtained from the ENCODE consortium (<https://www.encodeproject.org/>) and from NCBI repositories (e.g., SRA/GEO; <https://www.ncbi.nlm.nih.gov/>) and detailed in Supplementary Table 1. Gene annotations were obtained from GENCODE (<https://www.gencodegenes.org/>) and Ensembl (<https://www.ensembl.org/>). Human regulatory element annotations were obtained from the ENCODE SCREEN database (<https://screen.wenglab.org/>). NTv3 BENCHMARK data is described in Supplementary Table C.3 and available at [https://huggingface.co/datasets/InstaDeepAI/NTv3\\_benchmark\\_dataset](https://huggingface.co/datasets/InstaDeepAI/NTv3_benchmark_dataset). Additional dataset-specific accession identifiers and processing details are provided in the Methods and Supplementary Notes and Tables.

## Code availability

We make code, model checkpoints, and interactive resources publicly available for research use as described below.

**Code repositories.** The JAX implementation of NTv3, including model definitions and inference code, is available on GitHub at <https://github.com/instadeepai/nucleotide-transformer>. PyTorch implementations supporting model loading, inference, fine-tuning, and downstream applications are distributed via Hugging Face and available from the NTv3 Space at <https://huggingface.co/spaces/InstaDeepAI/ntv3>.

**Interactive notebooks.** Executable tutorial and pipeline notebooks demonstrating inference, functional-track prediction, fine-tuning, interpretation, and sequence generation are provided through Hugging Face Spaces:

- Tutorial notebooks:  
[https://huggingface.co/spaces/InstaDeepAI/ntv3/tree/main/notebooks\\_tutorials](https://huggingface.co/spaces/InstaDeepAI/ntv3/tree/main/notebooks_tutorials)
- Pipeline notebooks:  
[https://huggingface.co/spaces/InstaDeepAI/ntv3/tree/main/notebooks\\_pipelines](https://huggingface.co/spaces/InstaDeepAI/ntv3/tree/main/notebooks_pipelines)

All NTv3 model checkpoints are released on Hugging Face via the [NTv3 collection](#). We provide both pre-trained and post-trained models at multiple scales, together with variants used for length-ablation and architectural studies.

*Primary released models* (used throughout the main paper):

- NTv3 8M (pre): [https://huggingface.co/InstaDeepAI/NTv3\\_8M\\_pre](https://huggingface.co/InstaDeepAI/NTv3_8M_pre)
- NTv3 100M (pre): [https://huggingface.co/InstaDeepAI/NTv3\\_100M\\_pre](https://huggingface.co/InstaDeepAI/NTv3_100M_pre)
- NTv3 650M (pre): [https://huggingface.co/InstaDeepAI/NTv3\\_650M\\_pre](https://huggingface.co/InstaDeepAI/NTv3_650M_pre)
- NTv3 100M (post): [https://huggingface.co/InstaDeepAI/NTv3\\_100M\\_post](https://huggingface.co/InstaDeepAI/NTv3_100M_post)
- NTv3 650M (post): [https://huggingface.co/InstaDeepAI/NTv3\\_650M\\_post](https://huggingface.co/InstaDeepAI/NTv3_650M_post)

*Length-specific checkpoints* (used for controlled sequence-length ablations):

- NTv3 650M (pre) (8 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_650M\\_pre\\_8kb](https://huggingface.co/InstaDeepAI/NTv3_650M_pre_8kb)
- NTv3 100M (pre) (8 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_100M\\_pre\\_8kb](https://huggingface.co/InstaDeepAI/NTv3_100M_pre_8kb)
- NTv3 8M (pre) (8 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_8M\\_pre\\_8kb](https://huggingface.co/InstaDeepAI/NTv3_8M_pre_8kb)
- NTv3 650M (post) (131 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_650M\\_post\\_131kb](https://huggingface.co/InstaDeepAI/NTv3_650M_post_131kb)
- NTv3 100M (post) (131 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_100M\\_post\\_131kb](https://huggingface.co/InstaDeepAI/NTv3_100M_post_131kb)

*Architectural ablation models* (five-downsampling variants):



- NTv3-5Downsample (650M, pre-trained): [https://huggingface.co/InstaDeepAI/NTv3\\_5downsample\\_pre](https://huggingface.co/InstaDeepAI/NTv3_5downsample_pre)
- NTv3-5Downsample (650M, pre-trained, 8 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_5downsample\\_pre\\_8kb](https://huggingface.co/InstaDeepAI/NTv3_5downsample_pre_8kb)
- NTv3-5Downsample (650M, post-trained): [https://huggingface.co/InstaDeepAI/NTv3\\_5downsample\\_post](https://huggingface.co/InstaDeepAI/NTv3_5downsample_post)
- NTv3-5Downsample (650M, post-trained, 131 kb context): [https://huggingface.co/InstaDeepAI/NTv3\\_5downsample\\_post\\_131kb](https://huggingface.co/InstaDeepAI/NTv3_5downsample_post_131kb)

**NTv3 Benchmark.** Benchmark leaderboard reporting model performance across tasks and species is available at: [https://huggingface.co/spaces/InstaDeepAI/ntv3\\_benchmark](https://huggingface.co/spaces/InstaDeepAI/ntv3_benchmark).

## Acknowledgments

This research was supported by Cloud TPUs from Google’s TPU Research Cloud (TRC). We also extend our gratitude to the researchers who deposited experimental data in public databases, those who maintain these databases, and those who make analytical and predictive methods accessible to the scientific community. F.K.L is supported by an ESPRIT fellowship of the Austrian Science Fund (FWF; 10.55776/ESP9305624 ). Y.S and V.K. work was supported by the National Science Foundation under award CAREER 2145577, and by the National Institutes of Health under award MIRA R35GM151243. Research at the Institute of Molecular Pathology (IMP) is supported by Boehringer Ingelheim GmbH and the Austrian Research Promotion Agency (FFG, FO999902549). Next-generation-sequencing (NGS) was performed by the Next Generation Sequencing Facility at Vienna BioCenter Core Facilities (VBCF), member of the Vienna BioCenter (VBC), Austria.

## Author contributions

B.P.d.A. and T.P conceived the research idea. S.B., B.E., Z.T., A.P., Y. A., C.R., T.K., S.S., D.E., J.M.-R., F.A.-A., E.S., Y.S., Y.B., A.H. and B.P.d.A. performed the computational analyses. F.L. performed the experimental work with advice from A.S. E.S. served as the scientific illustrator and led the figure synthesis. M.L., A.L., K.B., P.K., V.K. and A.S. provided advice on study design and analyses. B.P.d.A. and T.P. wrote the paper with contribution from all co-authors.

## Competing interests

S.B., B.E., Z.T., A.P., Y. A., C.R., T.K., S.S., D.E., J.M.-R., F.A.-A., E.S., Y.S., Y.B., A.H., M.L., A.L., K.B., B.P.d.A. and T.P. are employees of InstaDeep LTD. The remaining authors declare no competing interests.

## References

- [1] Gökçen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.
- [2] Gherman Novakovsky, Nick Dexter, Maxwell W. Libbrecht, Wyeth W. Wasserman, and Sara Mostafavi. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nature Reviews Genetics*, 24:125–137, 2023. doi: 10.1038/s41576-022-00532-2.
- [3] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931–934, 2015.
- [4] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature Biotechnology*, 33:831–838, 2015.
- [5] Bernardo P de Almeida, Franziska Reiter, Michaela Pagani, and Alexander Stark. Deepstarr predicts enhancer activity from dna sequence and enables the de novo design of synthetic enhancers. *Nature genetics*, 54(5):613–624, 2022.
- [6] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Sabrina Krueger, Amr Alexandari, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, and Julia Zeitlinger. Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nature Genetics*, 53(3):354–366, 2021. doi: 10.1038/s41588-021-00782-6.
- [7] Johannes Linder, Divyanshi Srivastava, Han Yuan, Vikram Agarwal, and David R. Kelley. Predicting RNA-seq coverage from DNA sequence as a unifying model of gene regulation. *Nature Genetics*, 57(4):949–961, 2025. doi: 10.1038/s41588-024-02053-6.
- [8] Žiga Avsec, Natasha Latysheva, Jun Cheng, Guido Novati, Kyle R. Taylor, Tom Ward, Clare Bycroft, Lauren Nicolaisen, Eirini Arvaniti, Joshua Pan, Raina Thomas, Vincent Dutordoir, Matteo Perino, Soham De, Alexander Karollus, Adam Gayoso, Toby Sargeant, Anne Mottram, Lai Hong Wong, Pavol Drotár, Adam Kosiorek, Andrew Senior, Richard Tanburn, Taylor Applebaum, Souradeep Basu, Demis Hassabis, and Pushmeet Kohli. Alphagenome: advancing regulatory variant effect prediction with a unified dna sequence model. *bioRxiv*, 2025. doi: 10.1101/2025.06.25.661532.
- [9] Kseniia Dudnyk, Donghong Cai, Chenlai Shi, Jian Xu, and Jian Zhou. Sequence basis of transcription initiation in the human genome. *Science*, 384:eadj0116, 2024.
- [10] Nicholas Bogard, Johannes Linder, Alexander B. Rosenberg, and Georg Seelig. A deep neural network for predicting and engineering alternative polyadenylation. *Cell*, 178:91–106, 2019.
- [11] Vikram Agarwal and David R. Kelley. The genetic and biochemical determinants of mrna degradation rates in mammals. *Genome Biology*, 23:245, 2022.
- [12] Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F. McRae, Siavash Fazel Darbandi, David Knowles, Yang I. Li, Jack A. Kosmicki, Juan Arbelaez, Wenwu Cui, Grace B. Schwartz, Eric D. Chow, Efsthios Kanterakis, Hong Gao, Amirali Kia, Serafim Batzoglou, Stephan J. Sanders, and Kyle Kai-How Farh. Predicting Splicing from Primary Sequence with Deep Learning. *Cell*, 176(3):535–548.e24, 2019. doi: 10.1016/j.cell.2018.12.015. Publisher: Elsevier.
- [13] Christof Angermueller, Heather J Lee, Wolf Reik, and Oliver Stegle. Deepcpvg: accurate prediction of single-cell dna methylation states using deep learning. *Genome Biology*, 18(67), 2017.
- [14] Jian Zhou. Sequence-based modeling of three-dimensional genome architecture from kilobase to chromosome scale. *Nature Genetics*, 54:725–734, 2022.
- [15] Kathleen M. Chen, Aaron K. Wong, Olga G. Troyanskaya, and Jian Zhou. A sequence-based global map of regulatory activity for deciphering human genetics. *Nature Genetics*, 54(7):940–949, 2022.

- 
- [16] Vikram Agarwal, Fumitaka Inoue, Max Schubach, Dmitry Penzar, Beth K. Martin, Pyaree Mohan Dash, Pia Keukeleire, Zicong Zhang, Ajuni Sohota, Jingjing Zhao, Ilias Georgakopoulos-Soares, William S. Noble, Galip Gürkan Yardımcı, Ivan V. Kulakovskiy, Martin Kircher, Jay Shendure, and Nadav Ahituv. Massively parallel characterization of transcriptional regulatory elements. *Nature*, 639:411–420, 2025.
  - [17] Jasper Janssens, Sara Aibar, Ibrahim Ihsan Taskiran, Joy N Ismail, Alicia Estacio Gomez, Gabriel Aughey, Katina I Spanier, Florian V De Rop, Carmen Bravo González-Blas, Marc Dionne, Krista Grimes, Xiao Jiang Quan, Dafni Papasokrati, Gert Hulselmans, Samira Makhzami, Maxime De Waegeneer, Valerie Christiaens, Tony Southall, and Stein Aerts. Decoding gene regulation in the fly brain. *Nature*, 601(7894):630–636, 2022.
  - [18] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203, 2021.
  - [19] 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.
  - [20] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.
  - [21] 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. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.
  - [22] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, Stefano Ermon, Christopher Ré, and Stephen Baccus. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 43177–43201. Curran Associates, Inc., 2023.
  - [23] Eric Nguyen, Michael Poli, Matthew G. Durrant, Brian Kang, Dhruva Katrekar, David B. Li, Liam J. Bartie, Armin W. Thomas, Samuel H. King, Garyk Brix, Jeremy Sullivan, Madelena Y. Ng, Ashley Lewis, Aaron Lou, Stefano Ermon, Stephen A. Baccus, Tina Hernandez-Boussard, Christopher Ré, Patrick D. Hsu, and Brian L. Hie. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.
  - [24] 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.
  - [25] Veniamin Fishman, Yuri Kuratov, Aleksei Shmelev, Maxim Petrov, Dmitry Penzar, Denis Shepelin, Nikolay Chekanov, Olga Kardymon, and Mikhail Burtsev. Gena-lm: a family of open-source foundational dna language models for long sequences. *Nucleic Acids Research*, 53(2):gkac1310, 01 2025.
  - [26] 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*, volume 235, pages 43632–43648. PMLR, 2024.
  - [27] Sam Boshar, Evan Trop, Bernardo P de Almeida, Liviu Copoiu, and Thomas Pierrot. Are genomic language models all you need? exploring genomic language models on protein downstream tasks. *Bioinformatics*, 40(9):btac529, 08 2024.
  - [28] Garyk Brix, Matthew G Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A Gonzalez, Samuel H King, David B Li, Aditi T Merchant, et al. Genome modeling and design across all domains of life with evo 2. *BioRxiv*, pages 2025–02, 2025.
-

- 
- [29] Frederikke Isa Marin, Felix Teufel, Marc Horrender, Dennis Madsen, Dennis Pultz, Ole Winther, and Wouter Boomsma. Bend: Benchmarking dna language models on biologically meaningful tasks. *arXiv preprint arXiv:2311.12570*, 2023.
  - [30] Ziqi Tang, Nirali Somia, Yiyang Yu, and Peter K. Koo. Evaluating the representational power of pre-trained dna language models for regulatory genomics. *Genome Biology*, 26(1):203, 2025. doi: 10.1186/s13059-025-03674-8.
  - [31] Aman Patel, Arpita Singhal, Austin Wang, Anusri Pampari, Maya Kasowski, and Anshul Kundaje. Dart-eval: A comprehensive dna language model evaluation benchmark on regulatory dna. *NeurIPS Datasets and Benchmarks*, 2024.
  - [32] 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)*, pages 4171–4186, 2019.
  - [33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
  - [34] Anusri Pampari, Anna Shcherbina, Evgeny Z. Kvon, Michael Kosicki, Surag Nair, Soumya Kundu, Arwa S. Kathiria, Viviana I. Risca, Kristiina Kuningas, Kaur Alasoo, William James Greenleaf, Len A. Pennacchio, and Anshul Kundaje. ChromBPNet: bias factorized, base-resolution deep learning models of chromatin accessibility reveal cis-regulatory sequence syntax, transcription factor footprints and regulatory variants. *bioRxiv: The Preprint Server for Biology*, 2025. doi: 10.1101/2024.12.25.630221.
  - [35] Mario Stanke and Stephan Waack. Gene prediction with a hidden markov model and a new intron submodel. *Bioinformatics*, 19:215–225, 2003.
  - [36] Mario Stanke, Oliver Keller, Irfan Gunduz, Alec Hayes, Stephan Waack, and Burkhard Morgenstern. Augustus: ab initio prediction of alternative transcripts. *Nucleic Acids Research*, 34:W435–W439, 2006.
  - [37] Bernardo P. de Almeida, Hugo Dalla-Torre, Guillaume Richard, Christopher Blum, Lorenz Hexemer, Maxence Gélard, Javier Mendoza-Revilla, Ziqi Tang, Frederikke I. Marin, David M. Emms, Priyanka Pandey, Stefan Laurent, Marie Lopez, Alexandre Laterre, Maren Lang, Uğur Şahin, Karim Beguir, and Thomas Pierrot. Annotating the genome at single-nucleotide resolution with DNA foundation models. *Nature Methods*, 22(11):2301–2315, 2025. doi: 10.1038/s41592-025-02881-2.
  - [38] Jianan Zhai, Andrew Gokaslan, Sheng-Kai Hsu, Shao-Ping Chen, Zhen-Yuan Liu, Eduardo Marroquin, Erin Czech, Brad Cannon, Alice Berthel, Miguel Cinta Romay, Matthew Pennell, Volodymyr Kuleshov, and Edward S. Buckler. PlantCAD2: A Long-Context DNA Language Model for Cross-Species Functional Annotation in Angiosperms. *bioRxiv*, 2025. doi: 10.1101/2025.08.27.672609.
  - [39] Javier Mendoza-Revilla, Evan Trop, Liam Gonzalez, Maša Roller, Hugo Dalla-Torre, Bernardo P de Almeida, Guillaume Richard, Jonathan Caton, Nicolas Lopez Carranza, Marcin Skwark, et al. A foundational large language model for edible plant genomes. *Communications Biology*, 7(1):835, 2024.
  - [40] Seungsoo Kim and Joanna Wysocka. Deciphering the multi-scale, quantitative cis-regulatory code. *Molecular Cell*, 83(3):373–392, 2023. doi: 10.1016/j.molcel.2022.12.032.
  - [41] François Spitz and Eileen EM Furlong. Transcription factors: from enhancer binding to developmental control. *Nature Reviews Genetics*, 13(9):613–626, 2012.
  - [42] Grant A. Jindal and Emma K. Farley. Enhancer grammar in development and disease. *Current Topics in Developmental Biology*, 144:1–29, 2021.
  - [43] Gergely Nagy and László Nagy. Motif grammar: The basis of the language of gene expression. *Computational and Structural Biotechnology Journal*, 18:2026–2032, 2020. doi: 10.1016/j.csbj.2020.07.007.
-

- 
- [44] Suhas S.P. Rao, Miriam H. Huntley, Neva C. Durand, Elena K. Stamenova, Ivan D. Bochkov, James T. Robinson, Adrian L. Sanborn, Ido Machol, Arina D. Omer, Eric S. Lander, and Erez Lieberman Aiden. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665–1680, 2014.
- [45] Eileen EM Furlong and Michael Levine. Developmental enhancers and chromosome topology. *Science*, 361(6409):1341–1345, 2018.
- [46] Michael R. O'Dwyer, Meir Azagury, Katharine Furlong, Amani Alsheikh, Elisa Hall-Ponsele, Hugo Pinto, Dmitry V. Fyodorov, Mohammad Jaber, Eleni Papachristoforou, Hana Benchetrit, James Ashmore, Kirill Makedonski, Moran Rahamim, Marta Hanzevacki, Hazar Yassen, Samuel Skoda, Adi Levy, Steven M. Pollard, Arthur I. Skoultschi, Yosef Buganim, and Abdenour Soufi. Nucleosome fibre topology guides transcription factor binding to enhancers. *Nature*, 638:251–260, 2025. doi: 10.1038/s41586-025-08492-z.
- [47] Vikram Agarwal, Fumitaka Inoue, Max Schubach, Dmitry Penzar, Beth K Martin, Pyaree Mohan Dash, Pia Keukeleire, Zicong Zhang, Ajuni Sohota, Jingjing Zhao, et al. Massively parallel characterization of transcriptional regulatory elements. *Nature*, 639(8054):411–420, 2025.
- [48] Qiliang Li, Kenneth R Peterson, Xiangdong Fang, and George Stamatoyannopoulos. Locus control regions. *Blood, The Journal of the American Society of Hematology*, 100(9):3077–3086, 2002.
- [49] Gabriella E. Martyn, Michael T. Montgomery, Hank Jones, Katherine Guo, Benjamin R. Doughty, Johannes Linder, Deepa Bisht, Fan Xia, Xiangmeng S. Cai, Ziwei Chen, Kelly Cochran, Kathryn A. Lawrence, Glen Munson, Anusri Pampari, Charles P. Fulco, Nidhi Sahni, David R. Kelley, Eric S. Lander, Anshul Kundaje, and Jesse M. Engreitz. Rewriting regulatory DNA to dissect and reprogram gene expression. *Cell*, 188(12):3349–3366.e23, 2025. doi: 10.1016/j.cell.2025.03.034.
- [50] Xilin Shen, Yayan Hou, Xueer Wang, Chunyong Zhang, Jilei Liu, Hongru Shen, Wei Wang, Yichen Yang, Meng Yang, Yang Li, Jin Zhang, Yan Sun, Kexin Chen, Lei Shi, and Xiangchun Li. A deep learning model for characterizing protein-rna interactions from sequences at single-base resolution. *Patterns*, 6(1):101150, 2025. doi: 10.1016/j.patter.2024.101150.
- [51] Evan E. Seitz, David M. McCandlish, Justin B. Kinney, and Peter K. Koo. Uncovering the mechanistic landscape of regulatory DNA with deep learning. *bioRxiv*, 2025. doi: 10.1101/2025.10.07.681052. URL <https://www.biorxiv.org/content/10.1101/2025.10.07.681052>.
- [52] Qingbo S. Wang, David R. Kelley, Jacob Ulirsch, Masahiro Kanai, Shuvom Sadhuka, Ran Cui, Carlos Albors, Nathan Cheng, Yukinori Okada, Koichi Matsuda, Yuji Yamanashi, Yoichi Furukawa, Takayuki Morisaki, Yoshinori Murakami, Yoichiro Kamatani, Kaori Muto, Akiko Nagai, Wataru Obara, Ken Yamaji, Kazuhisa Takahashi, Satoshi Asai, Yasuo Takahashi, Takao Suzuki, Nobuaki Sinozaki, Hiroki Yamaguchi, Shiro Minami, Shigeo Murayama, Kozo Yoshimori, Satoshi Nagayama, Daisuke Obata, Masahiko Higashiyama, Akihide Masumoto, Yukihiro Koretsune, Francois Aguet, Kristin G. Ardlie, Daniel G. MacArthur, Hilary K. Finucane, and The Biobank Japan Project. Leveraging supervised learning for functionally informed fine-mapping of cis-eQTLs identifies an additional 20,913 putative causal eQTLs. *Nature Communications*, 12(1):3394, 2021. doi: 10.1038/s41467-021-23134-8.
- [53] Houshang Monajemi, Ruud D. Fontijn, Hans Pannekoek, and Anton J.G. Horrevoets. The apolipoprotein l gene cluster has emerged recently in evolution and is expressed in human vascular tissue. *Genomics*, 79(4):539–546, 2002. ISSN 0888-7543. doi: <https://doi.org/10.1006/geno.2002.6729>. URL <https://www.sciencedirect.com/science/article/pii/S0888754302967297>.
- [54] William McLaren, Laurent Gil, Sarah E Hunt, Harpreet Singh Riat, Graham R S Ritchie, Anja Thormann, Paul Flicek, and Fiona Cunningham. The Ensembl Variant Effect Predictor. *Genome Biology*, 17(1):122, 2016. doi: 10.1186/s13059-016-0974-4. URL <https://doi.org/10.1186/s13059-016-0974-4>.
- [55] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.
-



- 
- [56] Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 1, 2025.
  - [57] Cosmas D Arnold, Daniel Gerlach, Christoph Stelzer, Łukasz M Boryń, Martina Rath, and Alexander Stark. Genome-wide quantitative enhancer activity maps identified by starr-seq. *Science*, 339(6123): 1074–1077, 2013.
  - [58] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.
  - [59] Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models, 2024.
  - [60] Muhammad A Zabidi, Cosmas D Arnold, Katharina Schernhuber, Michaela Pagani, Martina Rath, Olga Frank, and Alexander Stark. Enhancer–core-promoter specificity separates developmental and housekeeping gene regulation. *Nature*, 518(7540):556–559, 2015.
  - [61] Hu Zhao, Zhuo Tu, Yinmeng Liu, Zhanxiang Zong, Jiacheng Li, Hao Liu, Feng Xiong, Jinling Zhan, Xuehai Hu, and Weibo Xie. Plantdeepsea, a deep learning-based web service to predict the regulatory effects of genomic variants in plants. *Nucleic Acids Research*, 49(W1):W523–W529, 05 2021.
  - [62] Guanqing Liu, Long Chen, Yuechao Wu, Yangshuo Han, Yu Bao, and Tao Zhang. Pdllms: A group of tailored dna large language models for analyzing plant genomes. *Molecular Plant*, 18(2):175–178, 2025. ISSN 1674-2052. doi: <https://doi.org/10.1016/j.molp.2024.12.006>. URL <https://www.sciencedirect.com/science/article/pii/S1674205224003903>. A Focus Issue on Crop2035.
  - [63] C. Huang et al. Personal transcriptome variation is poorly explained by current genomic deep learning models. *Nature Genetics*, 55:2056–2059, December 2023.
  - [64] A. Sasse et al. Benchmarking of deep neural networks for predicting personal gene expression from dna sequence highlights shortcomings. *Nature Genetics*, 55:2060–2064, December 2023.
  - [65] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.
  - [66] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
  - [67] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL <http://github.com/google/flax>.
  - [68] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
  - [69] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
  - [70] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Eric Xing, and Zhiqiang Hu. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*, 2020.
  - [71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
  - [72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
-

- 
- [73] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL <https://www.sciencedirect.com/science/article/pii/S0925231223011864>.
  - [74] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941. PMLR, 2017.
  - [75] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
  - [76] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11671. URL <https://ojs.aaai.org/index.php/AAAI/article/view/11671>.
  - [77] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.
  - [78] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
  - [79] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
  - [80] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
  - [81] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, 2020. ISSN 1471-2164. doi: 10.1186/s12864-019-6413-7.
  - [82] Diego Calderon, Ronnie Blecher-Gonen, Xingfan Huang, Stefano Secchia, James Kentro, Riza M. Daza, Beth Martin, Alessandro Dulja, Christoph Schaub, Cole Trapnell, Erica Larschan, Kate M. O’Connor-Giles, Eileen E. M. Furlong, and Jay Shendure. The continuum of *Drosophila* embryonic development at single-cell resolution. *Science*, 377(6606):eabn5800, 2022. doi: 10.1126/science.abn5800.
  - [83] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature Methods*, 9(4):357–359, 2012.
  - [84] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009. doi: 10.1093/bioinformatics/btp352.
  - [85] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.
  - [86] Felix Krueger. Trim galore!: A wrapper around cutadapt and fastqc to consistently apply adapter and quality trimming to fastq files, with extra functionality for rrbs data. *Babraham Institute*, 2015.
  - [87] Fidel Ramírez, Devon P Ryan, Björn Grüning, Vivek Bhardwaj, Fabian Kilpert, Andreas S Richter, Steffen Heyne, Friederike Dünder, and Thomas Manke. deeptools2: a next generation web server for deep-sequencing data analysis. *Nucleic acids research*, 44(Web Server issue):W160, 2016.
  - [88] Simon Andrews, Felix Krueger, Anne Segonds-Pichon, Laura Biggins, Christel Krueger, and Steven Wingett. FastQC. January 2012.
  - [89] Picard toolkit. <https://broadinstitute.github.io/picard/>, 2019.
-

- 
- [90] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, 2008. doi: 10.1137/040608635. URL <https://doi.org/10.1137/040608635>.
  - [91] Fragkiskos D. Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *arXiv preprint arXiv:1308.0971*, 2013.
  - [92] Zhangyuan Pan, Ying Wang, Mingshan Wang, Yuzhe Wang, Xiaoning Zhu, Shenwen Gu, Conghao Zhong, Liqi An, Mingzhu Shan, Joana Damas, et al. An atlas of regulatory elements in chicken: A resource for chicken genetics and genomics. *Science Advances*, 9(18):eade1204, 2023.
  - [93] Ying-Juan Cheng, Yu Zhang, Jiao Yu, Xiang-Ru Meng, Zhong-Yuan Chang, Fang-Ling Jiang, Zhen Wu, Ying-Ying Zhang, Jia-Yu Xue, and Tian-Qi Zhang. Chromatin accessibility analysis reveals functional cis-regulatory regions related to fruit development and domestication in tomato. *The Plant Journal*, 123(3):e70416, 2025.
  - [94] Daehwan Kim, Joseph M Paggi, Chanhee Park, Christopher Bennett, and Steven L Salzberg. Graph-based genome alignment and genotyping with hisat2 and hisat-genotype. *Nature biotechnology*, 37(8): 907–915, 2019.
  - [95] Peter St. John, Dejun Lin, Polina Binder, Malcolm Greaves, Vega Shah, John St. John, Adrian Lange, Patrick Hsu, Rajesh Illango, Arvind Ramanathan, Anima Anandkumar, David H Brookes, Akosua Busia, Abhishaike Mahajan, Stephen Malina, Neha Prasad, Sam Sinai, Lindsay Edwards, Thomas Gaudet, Cristian Regep, Martin Steinegger, Burkhard Rost, Alexander Brace, Kyle Hippe, Luca Naef, Keisuke Kamata, George Armstrong, Kevin Boyd, Zhonglin Cao, Han-Yi Chou, Simon Chu, Allan dos Santos Costa, Sajad Darabi, Eric Dawson, Kieran Didi, Cong Fu, Mario Geiger, Michelle Gill, Darren J Hsu, Gagan Kaushik, Maria Korshunova, Steven Kothen-Hill, Youhan Lee, Meng Liu, Micha Livne, Zachary McClure, Jonathan Mitchell, Alireza Moradzadeh, Ohad Mosafi, Youssef Nashed, Saeed Paliwal, Yuxing Peng, Sara Rabhi, Farhad Ramezanghorbani, Danny Reidenbach, Camir Ricketts, Brian C Roland, Kushal Shah, Tyler Shimko, Hassan Sirelkhatim, Savitha Srinivasan, Abraham C Stern, Dorota Toczydlowska, Srimukh Prasad Veccham, Niccolò Alberto Elia Venanzi, Anton Vorontsov, Jared Wilber, Isabel Wilkinson, Wei Jing Wong, Eva Xue, Cory Ye, Xin Yu, Yang Zhang, Guoqing Zhou, Becca Zandstein, Alejandro Chacon, Prashant Sohani, Maximilian Stadler, Christian Hundt, Feiwen Zhu, Christian Dallago, Bruno Trentini, Emine Kucukbenli, Saeed Paliwal, Timur Rvachov, Eddie Calleja, Johnny Israeli, Harry Clifford, Risto Haukioja, Nicholas Haemel, Kyle Tretina, Neha Tadimet, and Anthony B Costa. Bionemo framework: a modular, high-performance library for ai model development in drug discovery, 2025. URL <https://arxiv.org/abs/2411.10548>.
  - [96] Mihaela Perte, Geo M Perte, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. *Nature biotechnology*, 33(3):290–295, 2015.
  - [97] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at the International Conference on Learning Representations (ICLR)*, 2014. URL <https://arxiv.org/abs/1312.6034>. arXiv preprint arXiv:1312.6034.
  - [98] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2022.
  - [99] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.
  - [100] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert’s attention. *arXiv preprint arXiv:1906.04341*, 2019.
  - [101] Ammar Tareen and Justin B Kinney. Logomaker: beautiful sequence logos in python. *Bioinformatics*, 36(7):2272–2274, 2020.
-

- [102] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.
- [103] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [104] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.
- [105] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.
- [106] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.
- [107] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.
- [108] Christoph Neumayr, Michaela Pagani, Alexander Stark, and Cosmas D Arnold. Starr-seq and umi-starr-seq: assessing enhancer activities for genome-wide-, high-, and low-complexity candidate libraries. *Current protocols in molecular biology*, 128(1):e105, 2019.
- [109] Emma E Laing, Carla S Möller-Levet, Norman Poh, Nayantara Santhi, Simon N Archer, and Derk-Jan Dijk. Blood transcriptome based biomarkers for human circadian phase. *eLife*, 6:e20214, feb 2017. ISSN 2050-084X. doi: 10.7554/eLife.20214. URL <https://doi.org/10.7554/eLife.20214>.
- [110] Chao Wang and Chun Liang. Dysregulation of locus-specific repetitive elements in tcga pan-cancers. *Genes*, 16(5), 2025. ISSN 2073-4425. doi: 10.3390/genes16050528. URL <https://www.mdpi.com/2073-4425/16/5/528>.
- [111] Shu Lou, Guirong Zhu, Changyue Xing, Shushu Hao, Junyan Lin, Jiayi Xu, Dandan Li, Yifei Du, Congbo Mi, Lian Sun, Lin Wang, Meilin Wang, Mulong Du, and Yongchu Pan. Transcriptome-wide association identifies KLC1 as a regulator of mitophagy in non-syndromic cleft lip with or without palate. *iMeta*, 3(6):e262, 2024. doi: 10.1002/imt2.262.
- [112] Jeongha Lee, Eun Young Jeon, Liyang Yu, Hye-Yeong Jo, Sang Cheol Kim, Woong-Yang Park, Hyun-Young Park, Siming Zhao, and Murim Choi. Integrated analysis of covid-19 multi-omics data for eqtls reveals genetic mechanisms underlying disease severity. *bioRxiv*, 2025. doi: 10.1101/2024.12.18.629144. URL <https://www.biorxiv.org/content/early/2025/05/29/2024.12.18.629144>.

# A foundational model for joint sequence-function multi-species modeling at scale for long-range genomic prediction

## *Supplementary Material*

---

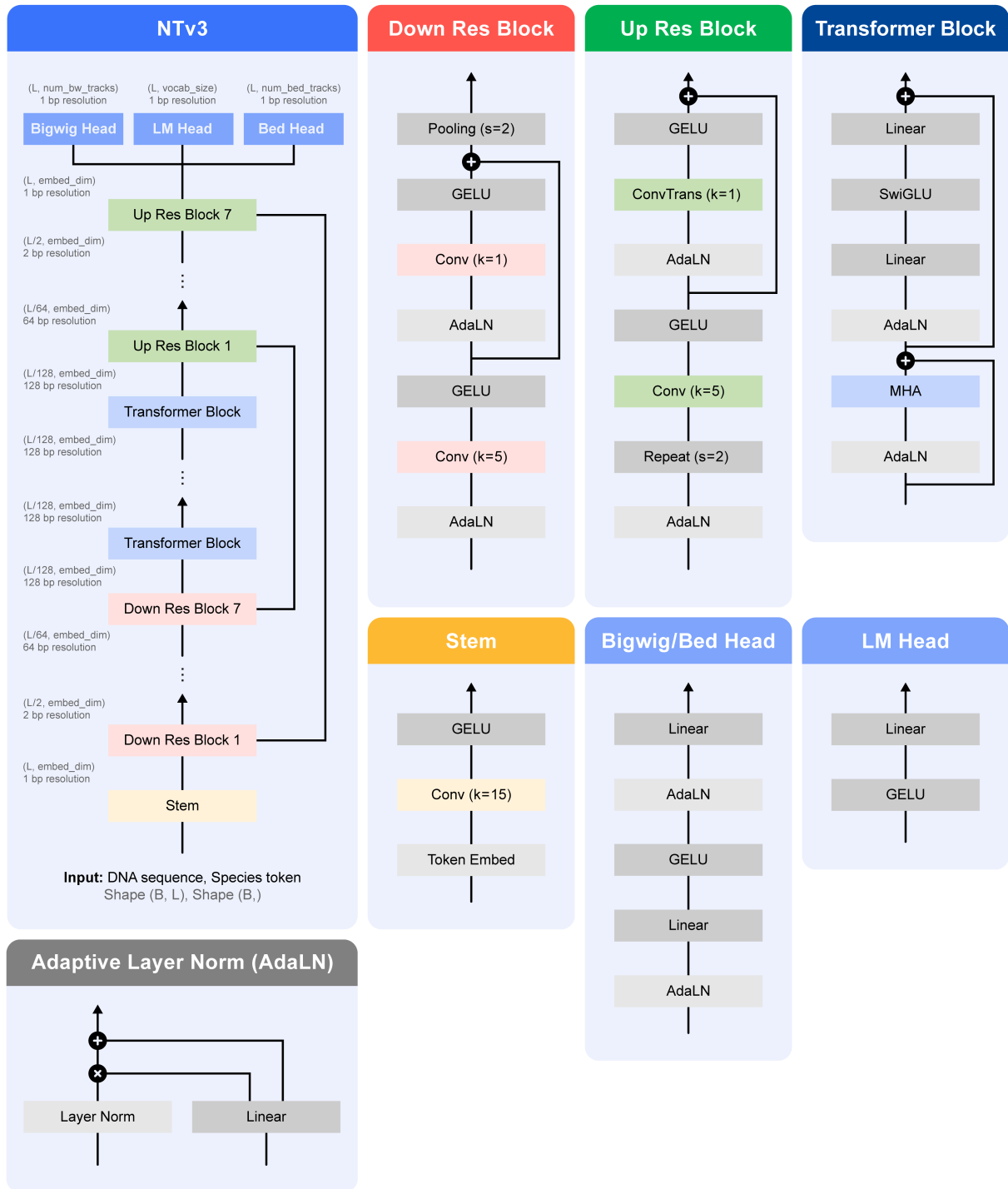
<b>A. Supplementary Figures</b>	<b>70</b>
A.1 NTv3 model architecture. . . . .	71
A.2 NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.	72
A.3 NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.	73
A.4 NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.	74
A.5 Post-training dynamics and impact of pre-training. . . . .	75
A.6 Screenshot of NTv3 predictions for an Arabidopsis genomic window. . . . .	76
A.7 Multispecies genome-annotation data and performance. . . . .	77
A.8 Effect of sequence length and length-extension training on genome-annotation performance. .	78
A.9 NTv3 performance on functional-track prediction. . . . .	79
A.10 NTv3 compared to Borzoi on human and mouse track prediction at 32bp resolution. . . . .	80
A.11 Detailed results on human NTv3 BENCHMARK functional tracks. . . . .	81
A.12 Impact of pre-training on NTv3 BENCHMARK performance. . . . .	82
A.13 In-silico and in-cellulo evaluation of NTv3 enhancers for the DSCP promoter. . . . .	83
<b>B. Supplementary Notes</b>	<b>84</b>
B.1 Model Optimization . . . . .	84
B.2 Model Scaling . . . . .	84
B.2.1 Pre-training Behavior Across Downsampling Depths . . . . .	85
B.2.2 Pre-training of Reduced-Downsampling Models . . . . .	85
B.2.3 Reduced-Downsampling in Models of Smaller Sizes . . . . .	85
B.2.4 Post-training of Reduced-Downsampling Models . . . . .	87
B.2.5 Evaluation on Downstream Tasks . . . . .	87
B.2.6 Conclusions . . . . .	87
B.3 Sequence length mixing . . . . .	89
B.3.1 Pre-training . . . . .	89
B.3.2 Post-training . . . . .	95



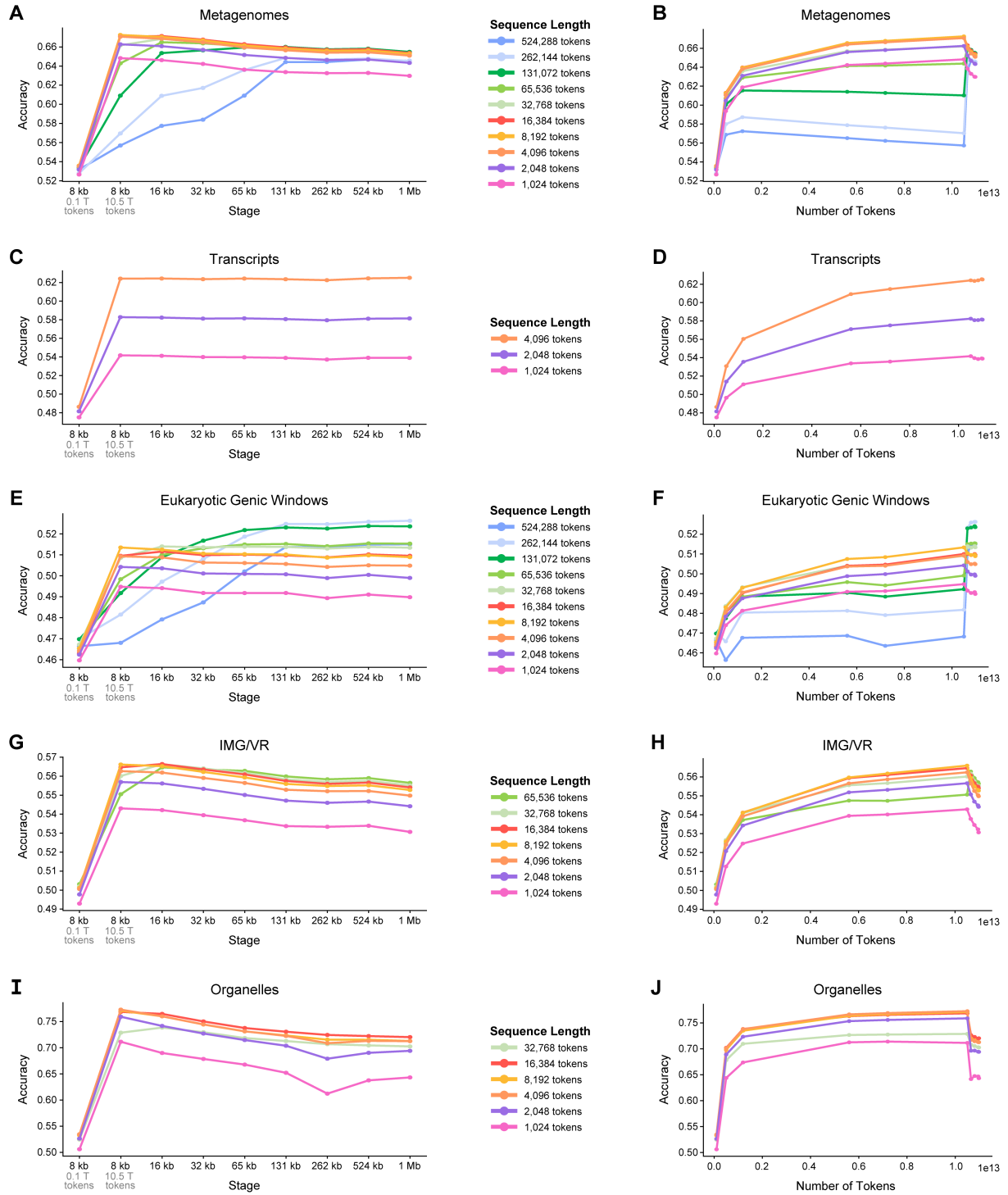
---

B.3.3	Implications . . . . .	95
B.4	Pre-Training Sequence Length Extension . . . . .	95
B.4.1	Evaluation . . . . .	97
B.4.2	7 Trillion Token Length Extension: Performance Across Sequence Lengths and Distributions	97
B.4.3	Choice of Length Extension Phase . . . . .	97
B.5	Multi-species Functional Tracks Head Implementation . . . . .	99
B.6	NTv3 benchmark . . . . .	99
B.6.1	Motivation . . . . .	99
B.6.2	Design choices . . . . .	100
B.6.3	Main conclusions . . . . .	101
B.7	Model interpretation . . . . .	101
B.7.1	Future Directions for Model Interpretability and Development . . . . .	102
B.7.2	Methodological Considerations for Interpretable Variant Prioritization . . . . .	102
B.7.3	Biological Interpretation of Representative Variant Case Studies . . . . .	104
B.8	Fine-tuning NTv3 for sequence generation . . . . .	105
<b>C.</b>	<b>Supplementary Table Legends</b>	<b>107</b>

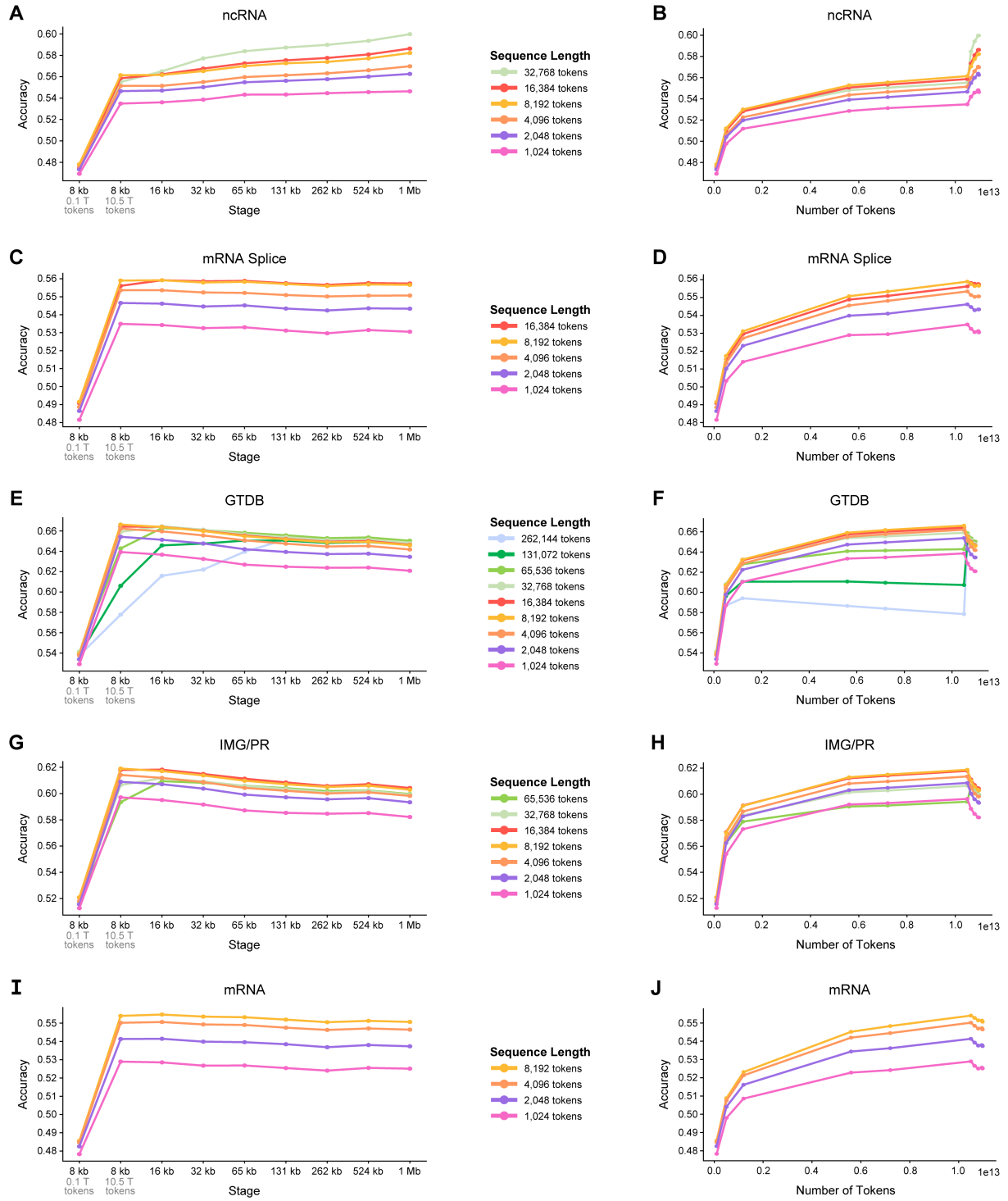
## A. Supplementary Figures



**Supplementary Figure A.1 | NTv3 model architecture.** NTv3 follows a U-Net-style hierarchy for single-nucleotide modeling: a stem produces 1-bp embeddings, a stack of seven downsampling residual blocks reduces resolution by  $2\times$  per stage to a 128-bp bottleneck, a Transformer tower performs long-range mixing at the bottleneck, and symmetric upsampling residual blocks reconstruct base-resolution features via skip connections. Task heads operate at 1-bp resolution: a regression head for functional tracks, a segmentation head for genome annotations, and an LM head for masked language modeling and variant-effect probabilities. The right panels summarize the building blocks used throughout: convolutional residual down/up blocks (with pooling or upsampling), a pre-norm Transformer block, lightweight MLP prediction heads, and an adaptive LayerNorm for conditional normalization (e.g., species-specific modulation) in multispecies training.

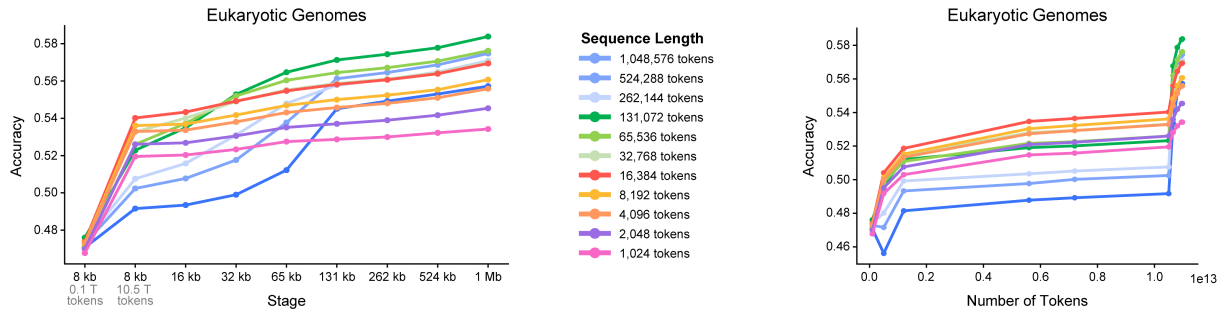


**Supplementary Figure A.2 | NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.** Line plots shows masked-token accuracy at each sequence length, evaluated on a fixed set of held-out tokens. (A,C,E,G,I) show accuracy at the end of every pre-training stage. (B,D,F,H,J) show accuracy in function of pre-training tokens.

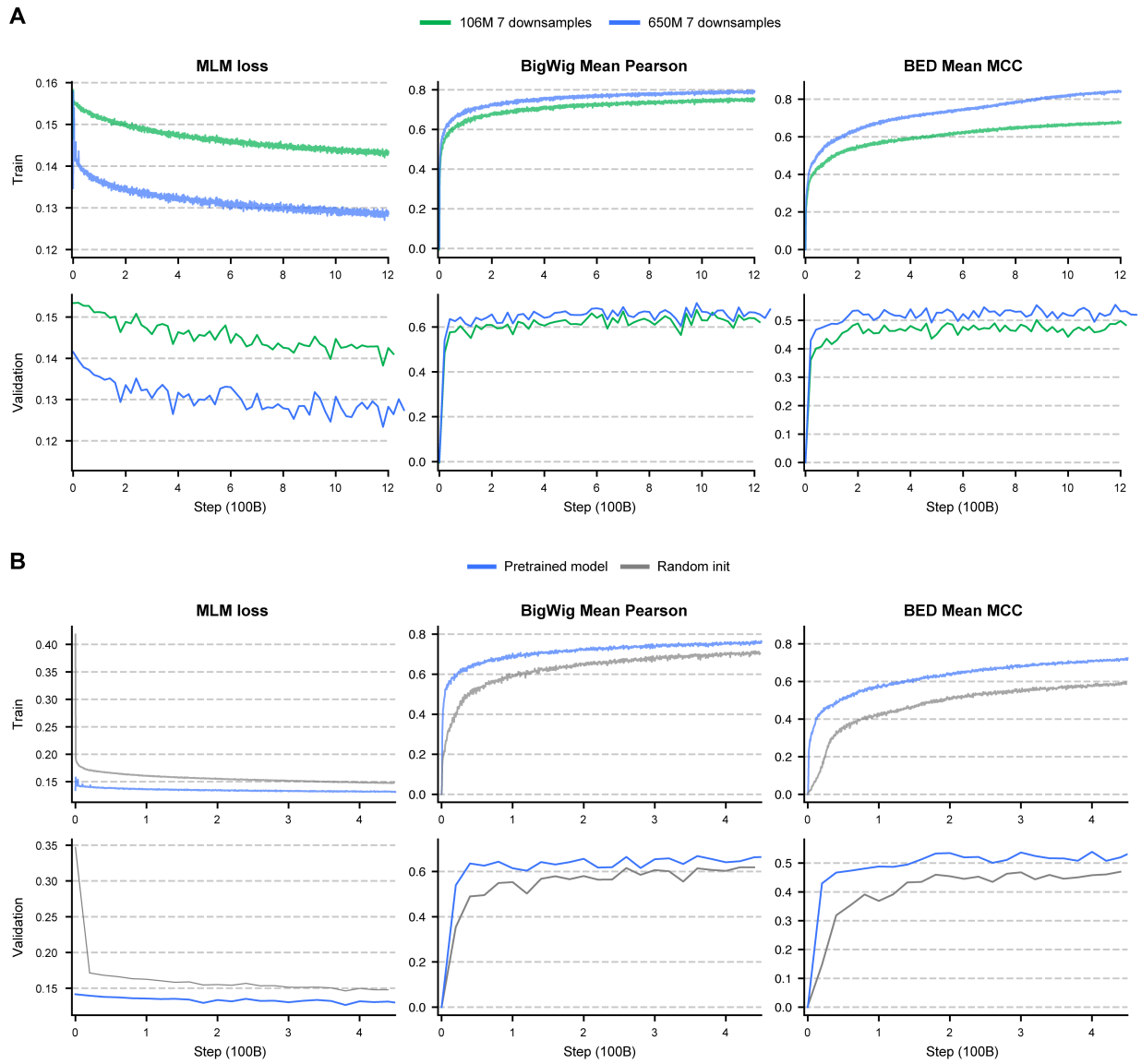


**Supplementary Figure A.3 | NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.** Line plots shows masked-token accuracy at each sequence length, evaluated on a fixed set of held-out tokens. (A,C,E,G,I) show accuracy at the end of every pre-training stage. (B,D,F,H,J) show accuracy in function of pre-training tokens.

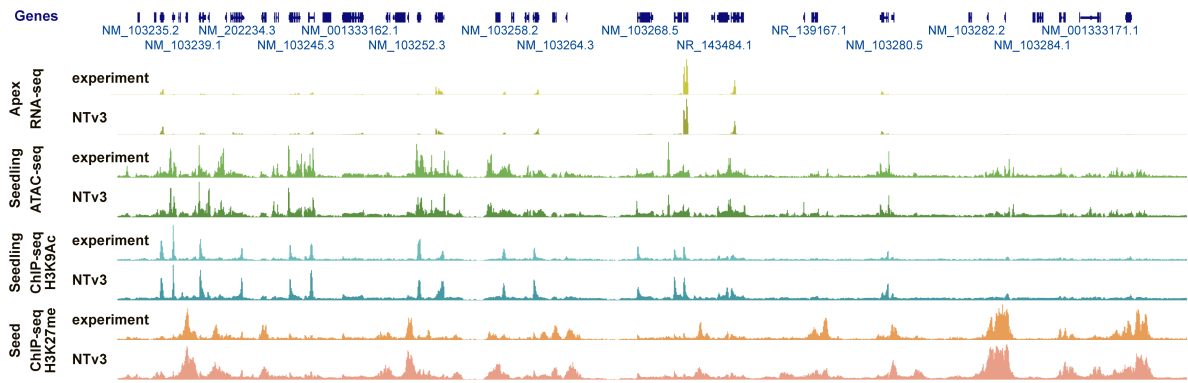




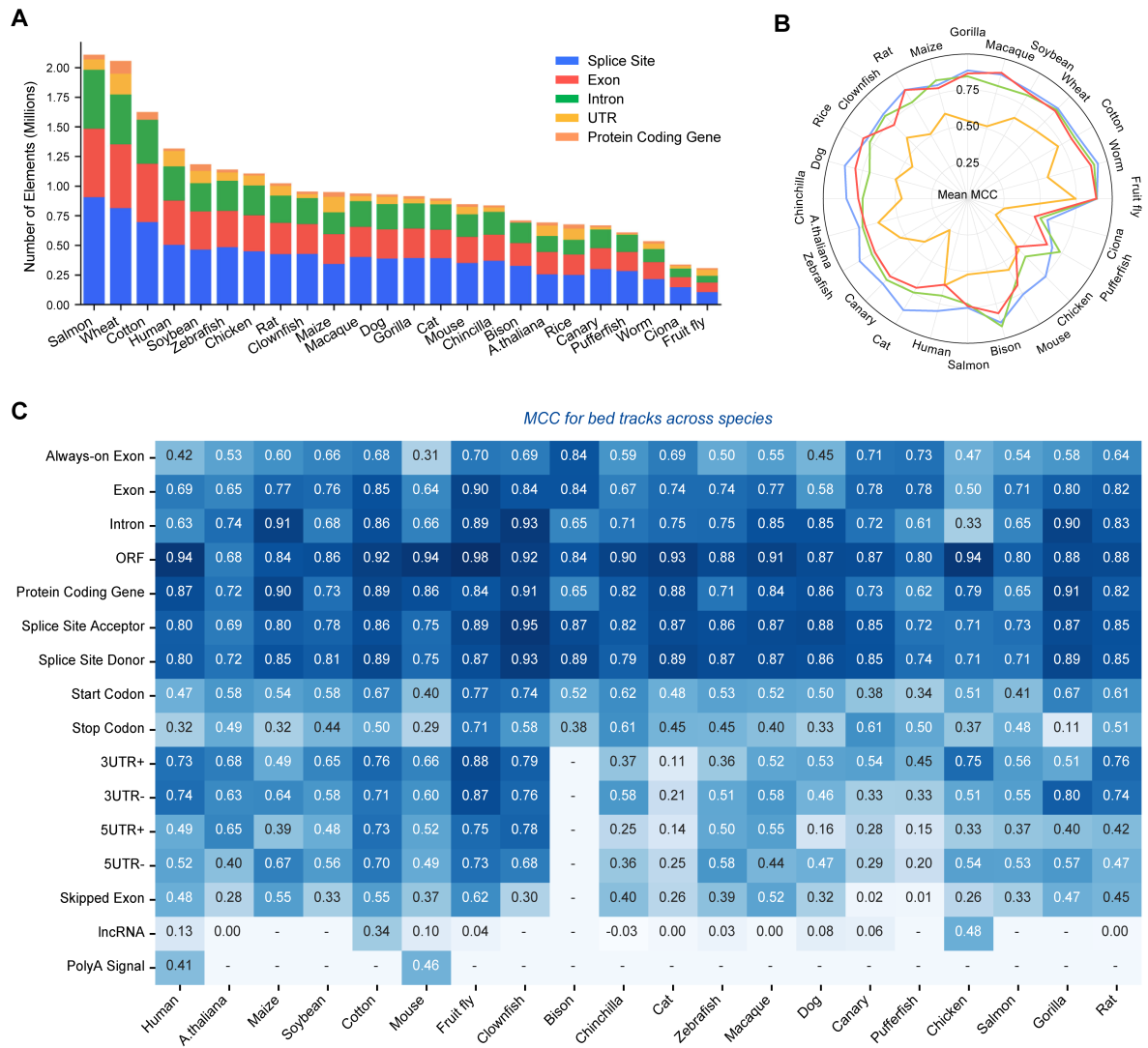
**Supplementary Figure A.4 | NTv3 performance across different sequence lengths on different OpenGenome2 sub-datasets.** Line plots shows masked-token accuracy at each sequence length, evaluated on a fixed set of held-out tokens. (A) shows accuracy at the end of every pre-training stage. (B) shows accuracy in function of pre-training tokens.



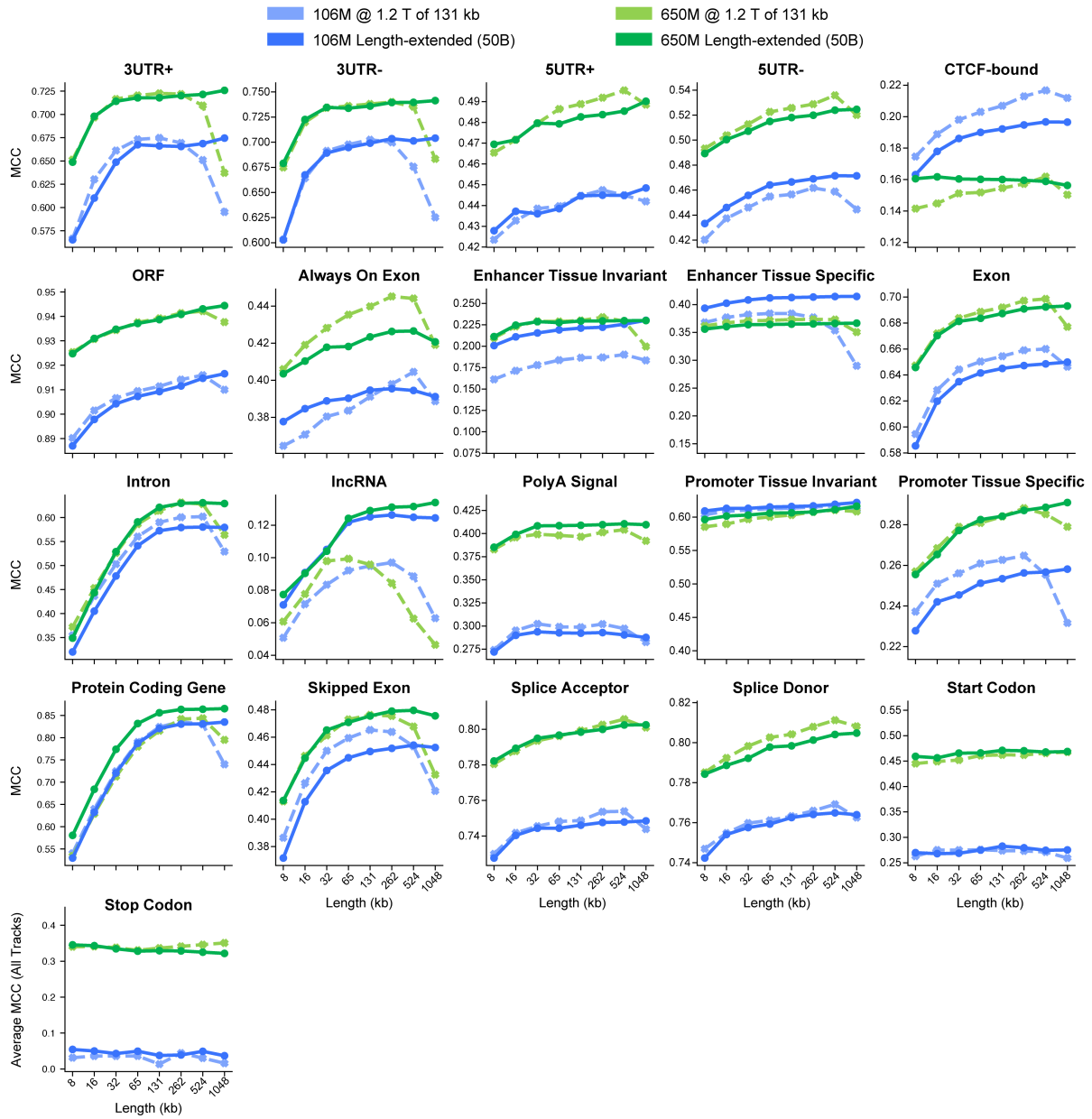
**Supplementary Figure A.5 | Post-training dynamics and impact of pre-training.** A) Post-training learning curves for NTv3 100M (post) 106M (green) and NTv3 650M (post) 650M (blue) under the joint objective, reporting masked language modeling (MLM) loss (left), mean Pearson correlation on functional tracks (middle), and mean MCC on genome-annotation labels (right). Curves are shown for both training (top) and validation (bottom) as a function of post-training steps (in units of 100B tokens). B) Effect of initialization on post-training. NTv3 650M (post) model initialized from the pretrained checkpoint (blue) is compared to the same architecture trained from random initialization (gray). Pretraining improves optimization and transfer, yielding lower MLM loss (left) and higher validation performance on both functional-track prediction (middle) and genome annotation (right) throughout training.



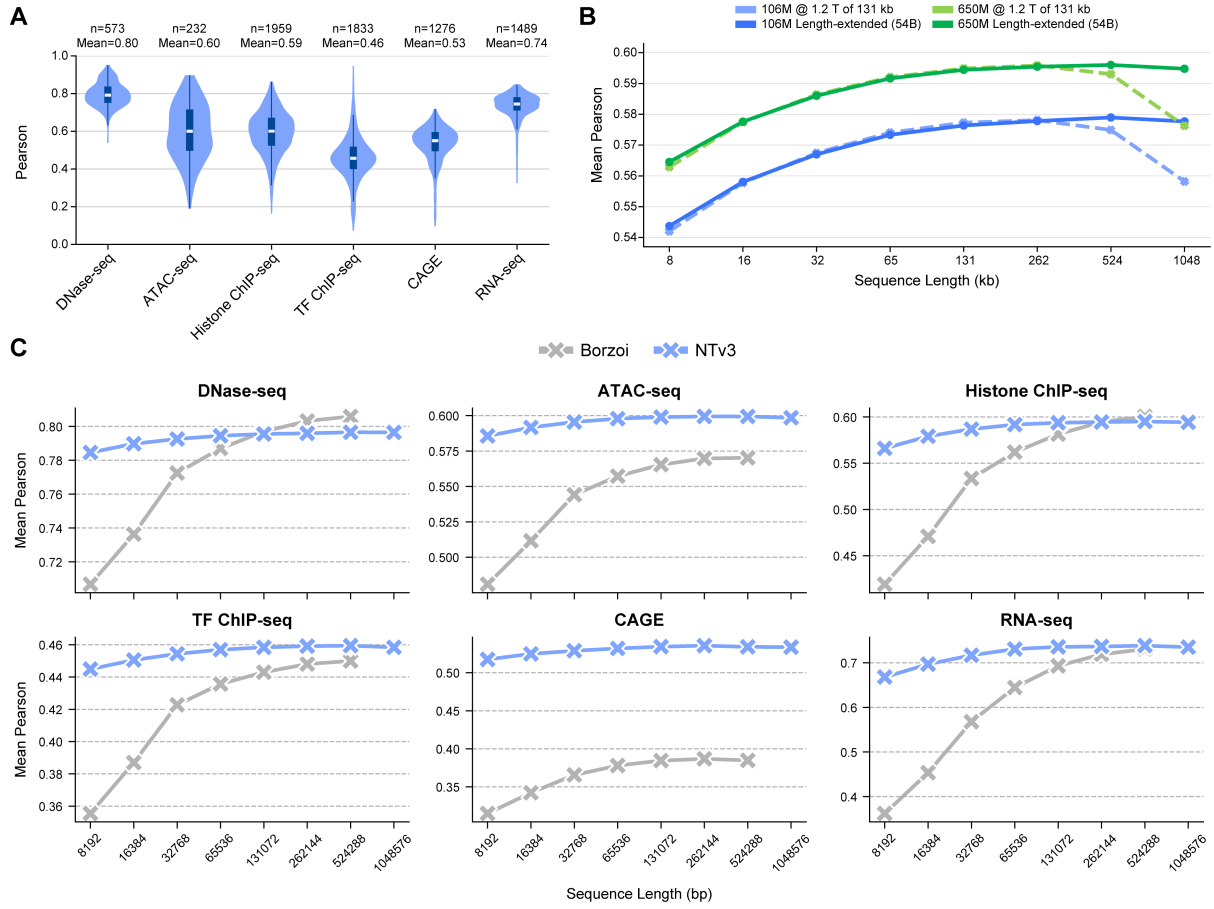
Supplementary Figure A.6 | Screenshot of NTv3 predictions for an Arabidopsis genomic window.



**Supplementary Figure A.7 | Multispecies genome-annotation data and performance.** **A)** Label statistics for the genome-annotation corpus across species, showing the number of annotated elements (in millions) per genome, broken down by element type (splice sites, exons, introns, UTRs, and protein-coding genes). **B)** Radar plot displaying cross-species genome-annotation accuracy for NTv3, summarized as mean Matthews correlation coefficient (MCC) per species and label family, highlighting consistent performance across a broad phylogenetic range. **C)** Heat map with detailed MCC for individual annotation labels across species. Each cell reports the per-label MCC on the test set; dashes indicate labels that are not available or not evaluated for a given species.

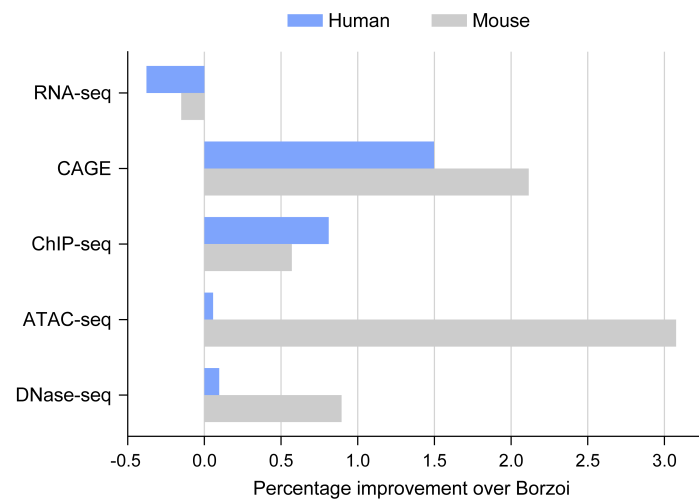


**Supplementary Figure A.8 | Effect of sequence length and length-extension training on genome-annotation performance.** Per-label MCC as a function of input sequence length (8 kb to 1 Mb) for NTv3 100M (post) and NTv3 models. Dashed curves denote checkpoints trained up to 131 kb context, while solid curves denote checkpoints that underwent additional length-extension training at 1Mb context. Each panel reports one annotation label and the final panel summarizes the average MCC across all labels.

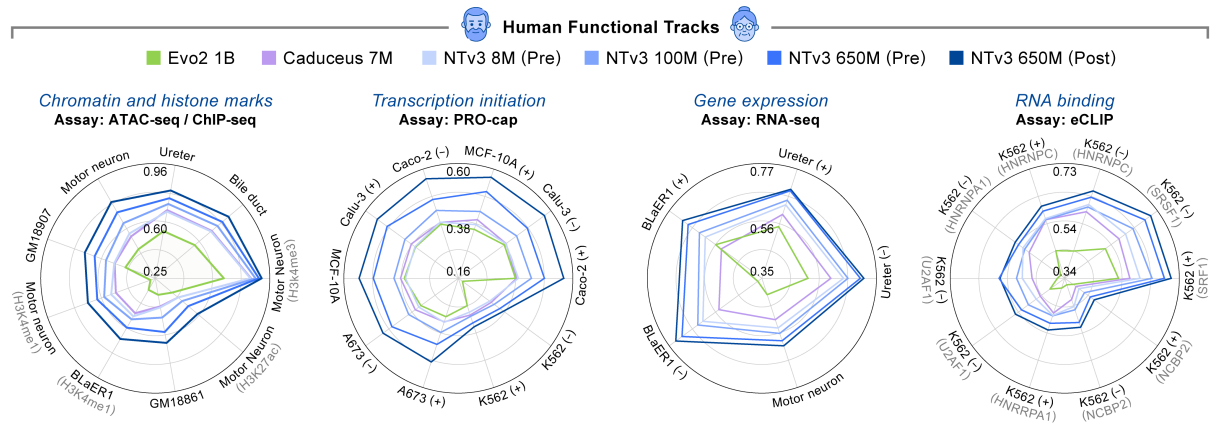


**Supplementary Figure A.9 | NTv3 performance on functional-track prediction.** **A)** Test-set Pearson correlation across human assay families (DNase-seq, ATAC-seq, histone ChIP-seq, TF ChIP-seq, CAGE, and RNA-seq); distributions summarize performance across tracks within each assay type (numbers indicate track counts and mean correlation). **B)** Mean Pearson correlation as a function of input sequence length for NTv3 100M and NTv3 models, comparing checkpoints trained up to 131 kb context to checkpoints further length-extended at long context; longer context yields consistent gains, with additional benefit from length extension at the largest windows. **C)** Per-assay comparison of NTv3 versus Borzoi across sequence lengths, reporting mean Pearson correlation over shared test regions; NTv3 matches Borzoi on DNase-seq and ChIP-seq and improves notably on ATAC-seq, CAGE, and RNA-seq, while maintaining stable performance across a wide range of input lengths.

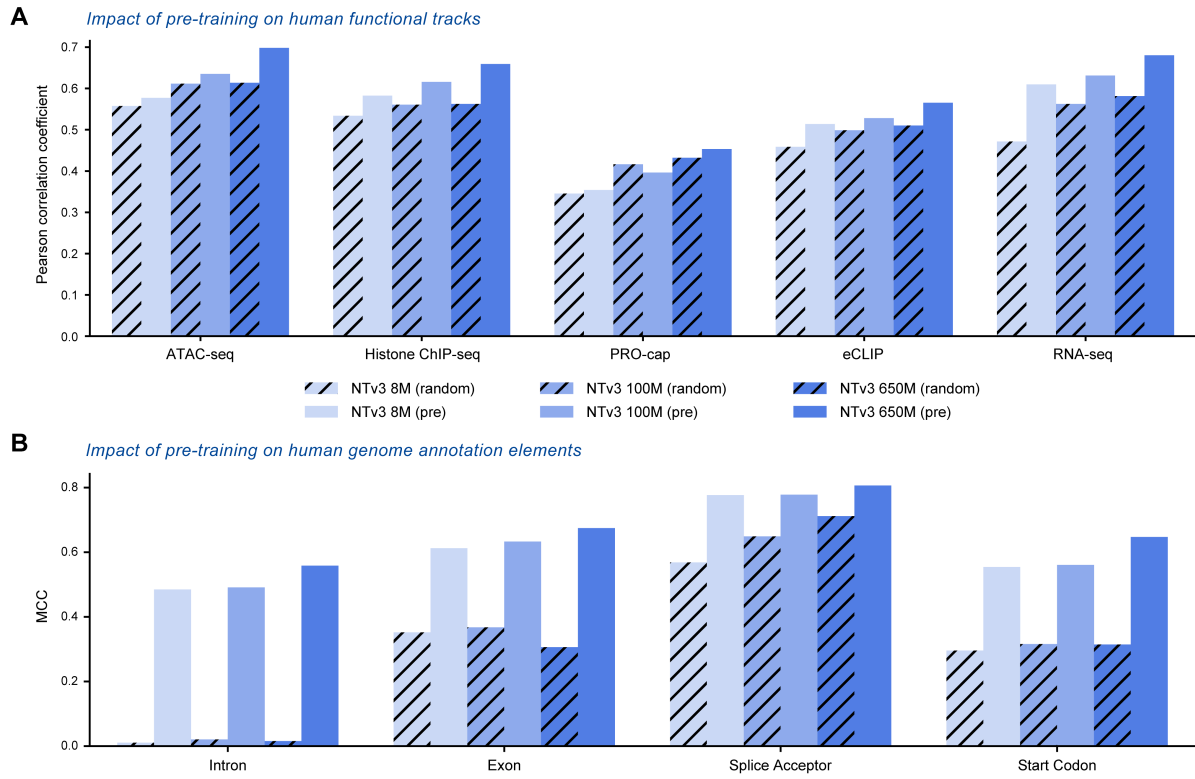




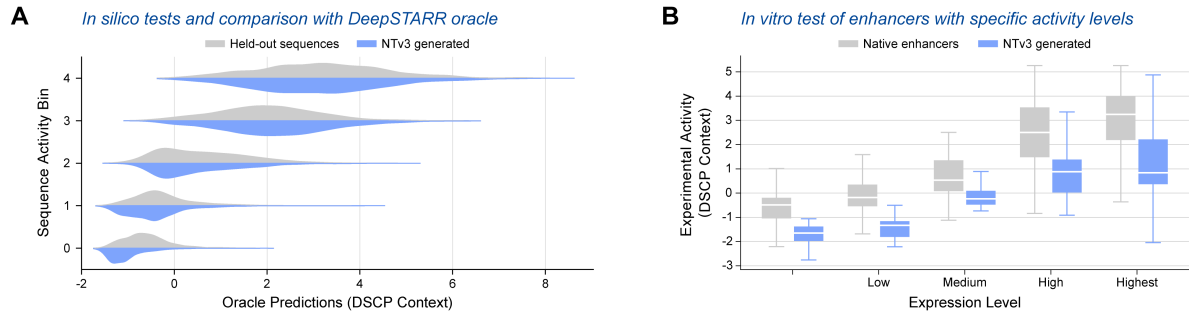
**Supplementary Figure A.10 | NTV3 compared to Borzoi on human and mouse track prediction at 32bp resolution.**  
Bar plot showing the percentage relative improvement over Borzoi for human and mouse tracks averaged per assay.



**Supplementary Figure A.11 | Detailed results on human NTv3 BENCHMARK functional tracks.** Radar plots displaying Pearson correlation across human functional tracks of different types.



**Supplementary Figure A.12 | Impact of pre-training on NTv3 BENCHMARK performance.** **A)** Functional-track prediction. Mean test-set Pearson correlation across assay groups in the NTv3 BENCHMARK for NTv3 models trained from random initialization versus initialized from genome MLM pre-training, shown for three model sizes (8M, 100M, and 650M). Pre-training improves performance across all assay families, with larger gains for more challenging modalities (e.g., PRO-cap, eCLIP, RNA-seq). **B)** Genome-annotation prediction. Test-set MCC for universal annotation labels (intron, exon, splice acceptor, start codon) under the same comparison. Pre-training yields substantial improvements across all elements and model sizes, with the largest gains for boundary-sensitive labels (splice acceptor and start codon).



**Supplementary Figure A.13 | In-silico and in-cellulo evaluation of NTv3 enhancers for the DSCP promoter.** **A)** *In silico* activity control evaluated with a DeepSTARR oracle [5]. Oracle-predicted activities of generated enhancers recapitulate the intended activity bins and closely match the distributions of held-out sequences. **B)** *In cellulo* STARR-seq measurements validate that generated enhancers express at the intended activity levels. NTv3 designs reproduce the expected stratification across activity bins when compared with native enhancers and also extend into a broader activity range, supporting the design of particularly strong regulatory elements.

## B. Supplementary Notes

This section provides additional methodological and analytical details that complement the main text and Supplementary Figures. We expand on key implementation choices for training and scaling NTv3, including data processing and optimization considerations, and provide supporting derivations, ablations, and clarifications that would otherwise interrupt the narrative flow of the paper. Together, these notes are intended to improve reproducibility and to serve as a practical cookbook for developing, scaling, and deploying new long-context genomic models beyond NTv3, spanning pre-training, multispecies post-training, fine-tuning, interpretation, and sequence-design workflows.

### B.1. Model Optimization

After selecting the base NTv3 architecture, we performed a targeted optimization study to improve training and inference throughput while preserving optimization stability and comparable training loss. Because the dominant bottlenecks differ between short and long contexts, we evaluated each change at two representative sequence lengths: 8 kb (short-context regime) and 262 kb (long-context regime). We report tokens processed per second and cumulative speedups relative to the unoptimized baseline.

Mixed-precision computation proved essential for increasing throughput. Executing convolution operations in BF16, rather than F32, yielded a large speedup and also reduced memory pressure, enabling larger effective batch sizes and higher token throughput. For the decoder, we replaced transposed convolutions with an explicit upsampling operation followed by a standard convolution, a design choice that has been shown to be efficient in related genomics architectures [8]; this change produced the largest gains in the long-context regime. Finally, we extended mixed precision to the Transformer dot-product operations (MLP projections and attention  $Q$ ,  $K$ ,  $V$ , and output projections), while keeping numerically sensitive components (e.g., layer normalization and other reductions) in higher precision to maintain training stability. Overall, these optimizations provide substantial cumulative throughput improvements at both short and long sequence lengths (Table 10).

Optimization	8 kb		262 kb	
	Tokens/s	Speedup	Tokens/s	Speedup
Baseline	537,492	1.00	311,996	1.00
BF16 convolutions	1,219,839	2.27	418,427	1.34
Upsample + conv	1,314,225	2.45	1,171,971	3.76
BF16 Transformer dot-products	1,428,066	2.66	1,289,580	4.13

**Table 10** | Cumulative throughput improvements from successive optimization steps evaluated at two context lengths. Tokens/s denotes tokens processed per second.

### B.2. Model Scaling

Model	4 Downsamp.	5 Downsamp.	6 Downsamp.	7 Downsamp.
NTv3 8M (pre)	5.32M*	6.11M	6.90M*	7.69M
NTv3 100M (pre)	85.20M*	92.29M	99.38M*	106.46M
NTv3 650M (pre)	566.84M	595.17M	623.50M	651.83M

**Table 11** | Parameter counts for each NTv3 model across different numbers of downsampling layers. \* indicates model configurations which were not explored in the paper but which were included in the table for completeness.

While foundation models are typically scaled by increasing parameter count (depth and width), U-Net architectures offer an orthogonal scaling axis: the resolution of the latent representation processed by the central Transformer. Following previous work in genomic foundation modeling [7, 8, 18], the standard NTv3 configuration employs  $N = 7$  down-sampling blocks, compressing the input sequence by a factor of 128 ( $2^7$ ) before it reaches the Transformer. This compression makes long-sequence training tractable but lowers the effective resolution of Transformer tokens, potentially limiting the ability to learn fine-grained nucleotide interactions. As we aim to develop models capable of accurately processing genomic sequences across all scales,

from short regulatory elements to megabase-context windows, higher-resolution latent representations become an important consideration for shorter sequences.

Reducing the number of downsampling blocks shrinks the total parameter count but increases both the length of the sequence seen by the Transformer and the overall FLOPs required per training step. Here, we present what is, to our knowledge, the first systematic study of this resolution–compute tradeoff in genomic language models.

To explore the compute-optimality trade-offs of this paradigm, we conducted the 8kb-training of the 652M NTv3 650M (pre) model with varying downsampling depths  $N \in 4, 5, 6, 7$ . These configurations correspond to latent resolutions of 16 bp, 32 bp, 64 bp, and 128 bp, respectively, while keeping all other model components constant (Fig. 2). Across these experiments, we find that five downsampling blocks provide the most FLOP-efficient training. Guided by this result, we train a 5-downsample configuration for each model size and characterize the resulting behavior.

### ***B.2.1. Pre-training Behavior Across Downsampling Depths***

We begin by evaluating how the number of downsampling blocks affects performance and compute cost, holding all other architectural and training settings fixed. This sweep, conducted on the largest NTv3 650M (pre) model to ensure relevance to current genomic foundation models of comparable scale, spans  $N \in 4, 5, 6, 7$ . The resulting insights motivate our use of the 5-downsample configuration for smaller model sizes.

In this study, we considered model pre-training performance through two complementary lenses: loss per token (sample efficiency) and loss per FLOP (compute efficiency) (Fig. 2). Sample efficiency improves consistently with higher resolution (smaller  $N$ ), with the  $N = 4$  model achieving the lowest loss per token, suggesting that higher-resolution latent representations capture more information per nucleotide (Fig. 2).

Interestingly, compute efficiency exhibits diminishing returns beyond  $N = 5$ . Although  $N = 4$  is sample-efficient, the substantial increase in FLOPs slows convergence relative to compute time. In contrast,  $N = 7$  achieves the fastest per-step throughput but reaches a higher loss floor. The  $N = 5$  configuration provides an attractive trade-off, converging as quickly relative to FLOPs as  $N = 4$  while requiring substantially less memory. This trend persists up to  $\sim 10^{21}$  FLOPs and 2 trillion tokens, indicating that the standard 7-downsample configuration may not be fully compute-optimal under a fixed FLOP budget (Fig. 2D).

### ***B.2.2. Pre-training of Reduced-Downsampling Models***

Motivated by these observations, we pre-trained a 5-downsample variant of the largest NTv3 650M (pre) to completion, following the methodology outlined in Section 4.2.

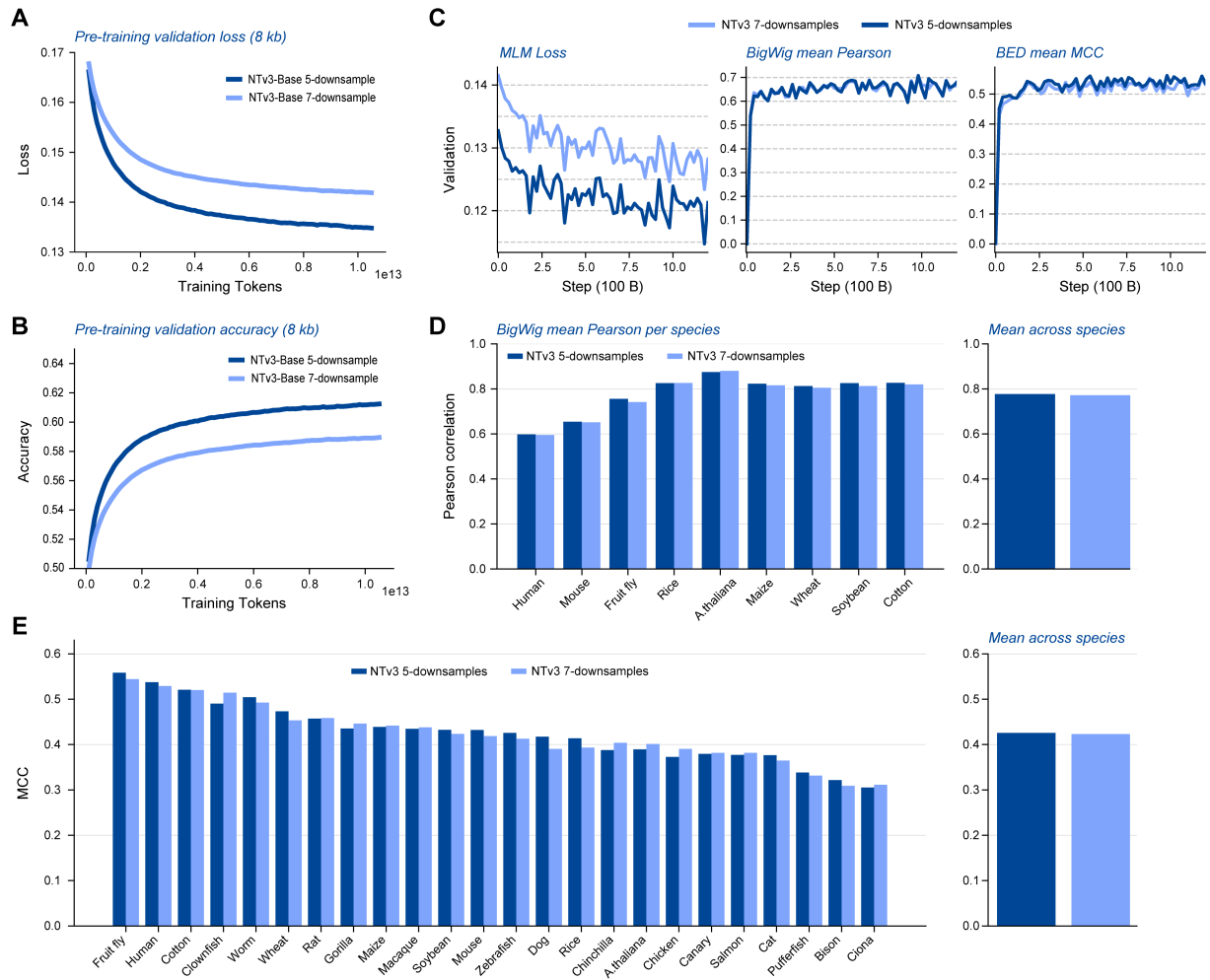
During pre-training, the 5-downsample variant showed substantial improvements: by the end of 10.5 trillion tokens in 8kb-training, it achieved a  $\sim 5\%$  reduction in training loss (0.1402 vs. 0.1314) and a  $\sim 2\%$  increase in training accuracy (Fig. 2E). The same also held for validation metrics at each of the four sequence lengths seen in 8kb-training. This corresponded to a decrease in perplexity from 1.151 to 1.140. Similar improvements were observed during sequence length extension.

### ***B.2.3. Reduced-Downsampling in Models of Smaller Sizes***

The success of the 5-downsample variant in the largest NTv3 650M (pre) model motivated us to explore the same configuration for smaller model sizes. However, we found that decreasing the number of downsamples did not yield comparable improvements for these smaller models. In fact, for the NTv3 8M (pre) model, pre-training performance was slightly worse than the standard 7-downsample configuration (Fig. 2E).

We hypothesize that this is due to the disproportionate reduction in parameter count caused by decreasing the number of downsamples, which limits model expressivity. For instance, NTv3 100M (pre) decreased from 106M to 92M parameters, retaining only 87% of its original capacity, while NTv3 8M (pre) dropped from 7.7M to 6.1M parameters, keeping just 79%. In contrast, the largest NTv3 650M (pre) retained  $\approx 91\%$  of its parameters with a similar reduction in downsampling (Supplementary Table 11 and Supplementary Fig. B.1A,B). For the smallest model, this substantial loss of capacity likely limited its representational power, accounting for the absence of performance gains.





**Supplementary Figure B.1 | Model Scaling on Pre- and Post-training** A) Validation loss during pre-training the 5 and 7 downsample models. B) Validation accuracy during pre-training the 5 and 7 downsample models. C) Validation MLM loss, mean functional track Pearson and mean genome annotation MCC during post-training the 5 and 7 downsample models. D) Mean functional track Pearson correlation on the test set for each species after post-training. E) Mean genome annotation MCC on the test set for each species after post-training.

These findings suggest that simply reducing downsampling is not always beneficial for smaller models. They motivate future work exploring approaches that increase latent resolution while maintaining or matching overall parameter count. We expect that particularly for the smallest models, where memory constraints are less severe, performance could benefit from reallocating computation from convolutional layers to the Transformer, effectively increasing FLOPs per parameter.

#### **B.2.4. Post-training of Reduced-Downsampling Models**

Based on the observation that reducing the number of downsampling layers did not improve performance in smaller models, we concentrated subsequent post-training experiments on only the largest NTv3 650M (pre) model. We continued post-training under conditions identical to NTv3 650M (post), allowing a direct comparison with standard 7-downsample models.

Post-training, the 5-downsample variant continued to show a substantial improvement in masked language model loss, consistent with the gains observed during pre-training (Supplementary Figure B.1C). In contrast, functional track performance did not exhibit a significant difference (Supplementary Figure B.1D). One possible explanation is that, for this post-training experiment in particular, the dataset contains an enormous amount of supervised functional track data, reducing the environment in which pre-training improvements have the largest impact. Annotation track performance showed modest gains, consistent with previous results (Supplementary Figure B.1E), where pre-training disproportionately benefits annotation tracks compared to functional tracks, likely because there is comparatively less information available in annotation tracks. During training, we did not observe any substantial difference in the rate of convergence for functional or annotation tracks relative to the 7 downsample variant (Supplementary Figure B.1C).

Overall, the 5-downsample model matches or exceeds the performance of the 7-downsample variant in post-training, while achieving comparable results on functional tracks. We next investigate whether these improvements in pre- and post-training translate into measurable gains on downstream tasks.

#### **B.2.5. Evaluation on Downstream Tasks**

To determine whether these pre-training improvements translate to downstream performance, we evaluated both pre- and post-trained 5 down-sample models on a suite of tasks, including functional track prediction, genome annotation, and zero-shot variant effect prediction. Despite improved pre-training metrics, the 5 down-sample variant exhibited comparable, or modest improvements in performance to the 7-down-sample model across most tasks (Supplementary Figure B.2). This indicates a saturation effect, where additional pre-training improvements do not meaningfully translate to increases in downstream performance.

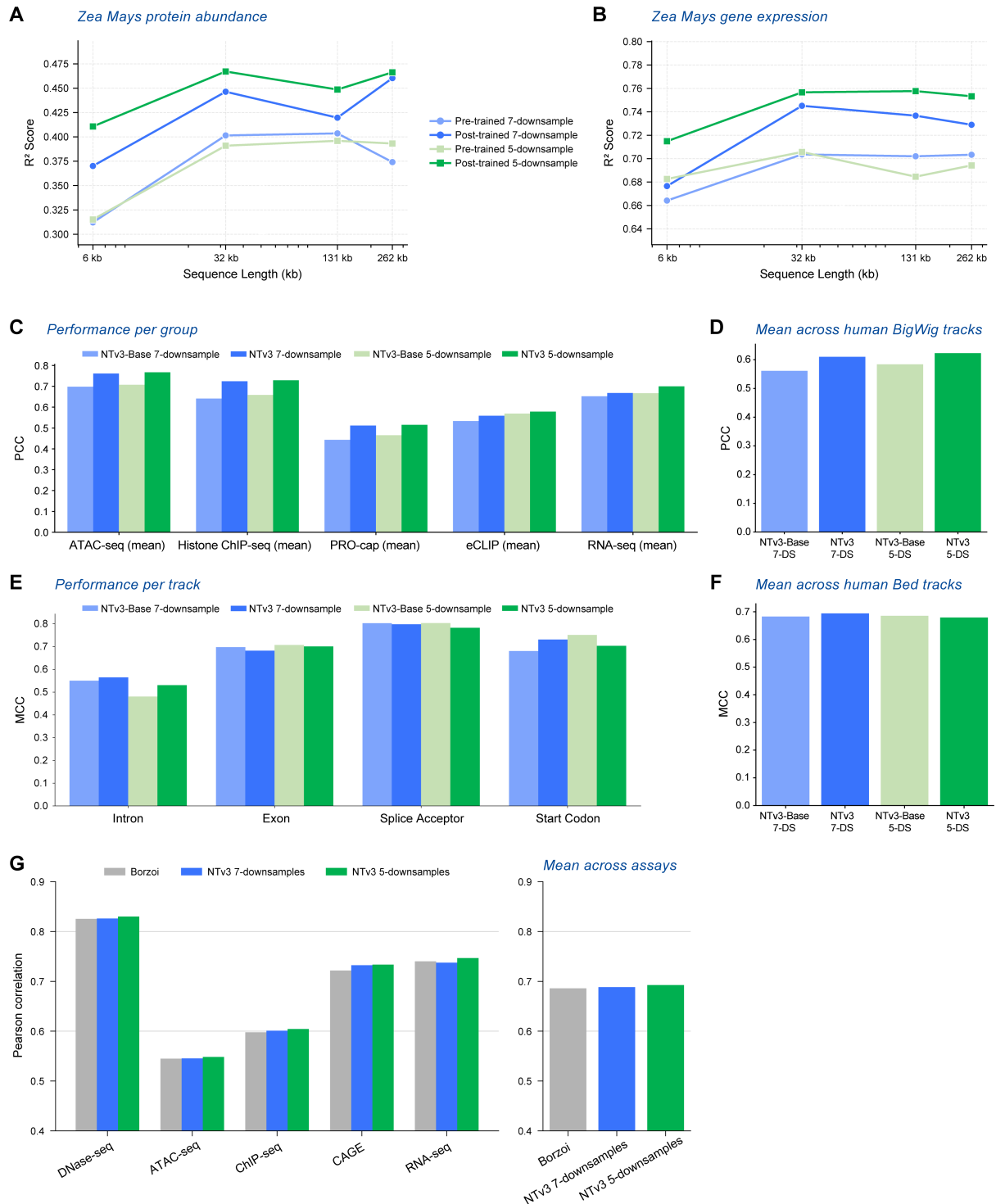
We hypothesize three possible explanations: (1) the pre-training objective may be less relevant when supervised fine-tuning data is abundant, (2) the 128 bp resolution ( $N = 7$ ) is sufficient for capturing biologically relevant long-range dependencies in current benchmarks, and (3)

#### **B.2.6. Conclusions**

Reducing the down-sampling factor  $N$  induces a quadratic increase in attention cost, quickly pushing against practical memory limits. In practice, this makes low down-sampling variants substantially more difficult to train and deploy, especially when they require sequence parallelism even for inference, limiting accessibility for the broader community.

As the  $N = 7$  configuration delivers comparable downstream performance while offering significantly higher training and inference throughput, we adopt it as the default setting for the main NTv3 models. Achieving meaningfully higher performance with lower  $N$  architectures would require improvements large enough to justify their added engineering and computational burden.

However, we note that the 5-downsample variant converges more quickly relative to computational cost (FLOPs). While our analysis focused on the final checkpoints, it is possible that intermediate checkpoints of the 5-downsample model would have reached equivalent downstream performance earlier in training than the 7-downsample model, before eventual saturation. We hypothesize that reduced-downsampling models could offer a more compute-efficient training trajectory, even if the final performance is similar. Exploring this possibility, including whether early convergence translates to downstream gains, represents an interesting



**Supplementary Figure B.2 | Model Scaling on Downstream Tasks** Comparison of post-trained 5- and 7-downsample models across downstream tasks. Panels A–F also include pre-trained counterparts unless noted. **A)** Protein abundance prediction in *Zea mays*. Post-training markedly improves performance across all sequence lengths, downsample variant perform similarly. **B)** Gene expression prediction in *Zea mays*. Post-trained models outperform pre-trained baselines; the post-trained 5-downsample model shows a modest improvement. **C)** Mean Pearson correlation across functional tracks (ATAC-seq, histone ChIP-seq, PRO-cap, eCLIP, RNA-seq). Post-trained models consistently score higher, with similar results for 5- and 7-downsample variants. **D)** Overall mean Pearson correlation across tracks. Post-training improves performance, and downsample (DS) variants perform comparably. **E)** Mean annotation track MCC for intron, exon, splice acceptor, and start codon classes. Performance is similar across downsample variants. **F)** Mean MCC across annotation tracks is comparable for all models. **G)** Fine-tuning on the *Borzoi* dataset yields comparable performance for the 5- and 7-downsample post-trained models.

OpenGenome Subdataset	Number of Tokens	Max Sequence Length in Evaluation Set ( $L_{\max}$ )
Eukaryotic Genic Windows	10,485,760	524,288
GTDB v220	18,874,368	262,144
IMG/PR	11,534,336	65,536
IMG/VR	17,825,792	65,536
Metagenomes	22,020,096	524,288
mRNA	29,360,128	8,192
mRNA Splice	13,631,488	16,384
NCBI Eukaryotic Genomes	79,691,776	1,048,576
ncRNA	13,631,488	32,768
Organelles	10,485,760	32,768
Transcripts	23,068,672	4,096

**Table 12** | Details on the OpenGenome2 Evaluation sets for each sub-dataset. For each sub-dataset we evaluated NTv3 on a held-out set, on all sequence lengths up to the reported maximum sequence length. The maximum evaluated sequence length is distinct for each sub-dataset as the distribution of sequence length within each sub-dataset is variable.

direction for future work.

Such efficiency considerations further highlight the broader trade-off between latent resolution and computational cost in model design. The trade-off between latent resolution and compute efficiency remains a promising avenue for scaling genomic language models, motivating further studies to systematically probe how pre-training loss, latent resolution, convergence dynamics, and downstream performance interact.

### B.3. Sequence length mixing

Recent advances in genomic language modeling have shown that extending input context to the megabase scale improves predictive performance by capturing distal regulatory interactions. However, models trained exclusively on long sequences often underperform on tasks requiring short-range context. Recently reported work [8] shows that on many functional tracks, models trained solely on long sequences exhibit degraded performance on shorter inputs, relative to models specialized for these shorter lengths. In testing, we found the same to hold true for other foundational models trained at long sequence lengths (Fig. 3G). This indicates a trade-off between capturing global genome-scale structure and retaining robustness on local, short-range tasks.

To investigate this phenomenon, we systematically evaluated how NTv3 performs across a range of sequence lengths and sub-dataset distributions, and how pre-training on sequences of different lengths affects short-context retention.

#### B.3.1. Pre-training

Dataset	1 kb	2 kb	4 kb	8 kb	16 kb	32 kb	64 kb	128 kb	256 kb	512 kb	1 mb	Row Total
Eukaryotic Genic Windows	2027	4899	11943	112336	110471	93764	51755	16855	5420	2697	1976	414149
GTDB v220	1516	12000	26433	38471	44825	46349	43627	38000	29377	19230	54903	354736
IMGPR	0	27	352	723	938	1075	964	815	570	232	51	5751
IMGVR	124	520	7911	7544	8102	9309	3023	1374	355	24	14	38305
Metagenomes	132013	184436	169243	125499	81595	56654	36972	22750	12169	5463	3015	829814
mRNA	12747	17940	13256	4931	1325	718	319	0	0	0	0	51240
mRNA Splice	9410	38018	29753	10856	2247	815	357	0	0	0	0	91459
NCBI Eukaryotic Genomes	0	0	0	0	0	616643	552967	456878	415052	448382	3740181	6230106
ncRNA	3057	2397	1682	887	722	1204	1198	494	421	41	0	12109
Organelles	0	0	0	107	153	34	141	2114	103	99	30	2784
Transcripts	603	2146	3563	2007	221	31	6	0	0	0	0	8580
<b>TOTAL</b>	161501	262388	264140	303367	250603	826601	691334	539284	463470	476172	3800172	8039036

**Table 13** | Raw token count per sub-dataset and sequence length, including row totals.

During pre-training, we observed that training predominantly on longer sequences, without continued exposure to shorter sequences, led to catastrophic short-context forgetting. In an initial experiment, an early version of NTv3 650M (pre), NTv3 650M (old pre), was trained in three sequential phases with increasing, but fixed, sequence lengths: 8 kb (3.8T tokens), 32 kb (6.7T tokens), and 1 mb (7T tokens). Each phase iterated once over all sequences at or above the target length, except for the 8 kb phase, which did not include any whole

genome sequences. Notably, sequences of 1 mb were present in both the 32 kb and 1 mb phases. Despite strong performance on long sequences and significant early training on shorter sequences, final checkpoints of NTv3 650M (old pre) exhibited poor performance on short sequences, highlighting short-context forgetting.

**Evaluation Procedure.** While final checkpoints performed well on the longest sequences, performance on shorter sequences degraded markedly. We reason that this degradation can come from two sources (1) short sequences come from a markedly different distribution than long sequences and model has forgotten this distribution, or (2) the model has lost the ability to process short contexts, regardless of distribution.

To investigate this phenomenon, we analyzed model performance across sequence lengths and sub-dataset distributions. Specifically, we evaluated the 650M-parameter NTv3 model on each OpenGenome2 sub-dataset using sequence lengths of the form  $L = 2^i$  for integers  $i \in \{10, 11, \dots, 20\}$ . Evaluation followed the same 15% masking procedure used during training, and we report masked-token prediction accuracy. To isolate the impact of sequence length, we evaluated the model on an identical held-out set of tokens for each sub-dataset, which we segmented into chunks corresponding to the sequence length under consideration.

To ensure statistically robust estimates, we imposed a minimum of 10 million validation tokens per evaluation, corresponding to approximately 1.5 million masked tokens on average. This requirement sets a maximum allowable sequence length for each sub-dataset: for a given sub-dataset, in order to evaluate the model on  $L = 2^i$ , there must be at least 10 million tokens in sequences of length  $\geq 2^i$ . More explicitly, this can be expressed as

$$L_{\max} = \max \{2^i \mid \text{number of tokens in sequences longer than } 2^i \geq 10^7\}.$$

Because the distribution of sequence lengths varies across sub-datasets, the number of usable tokens and the maximum evaluated sequence length differ for each. These values are summarized in Table 12.

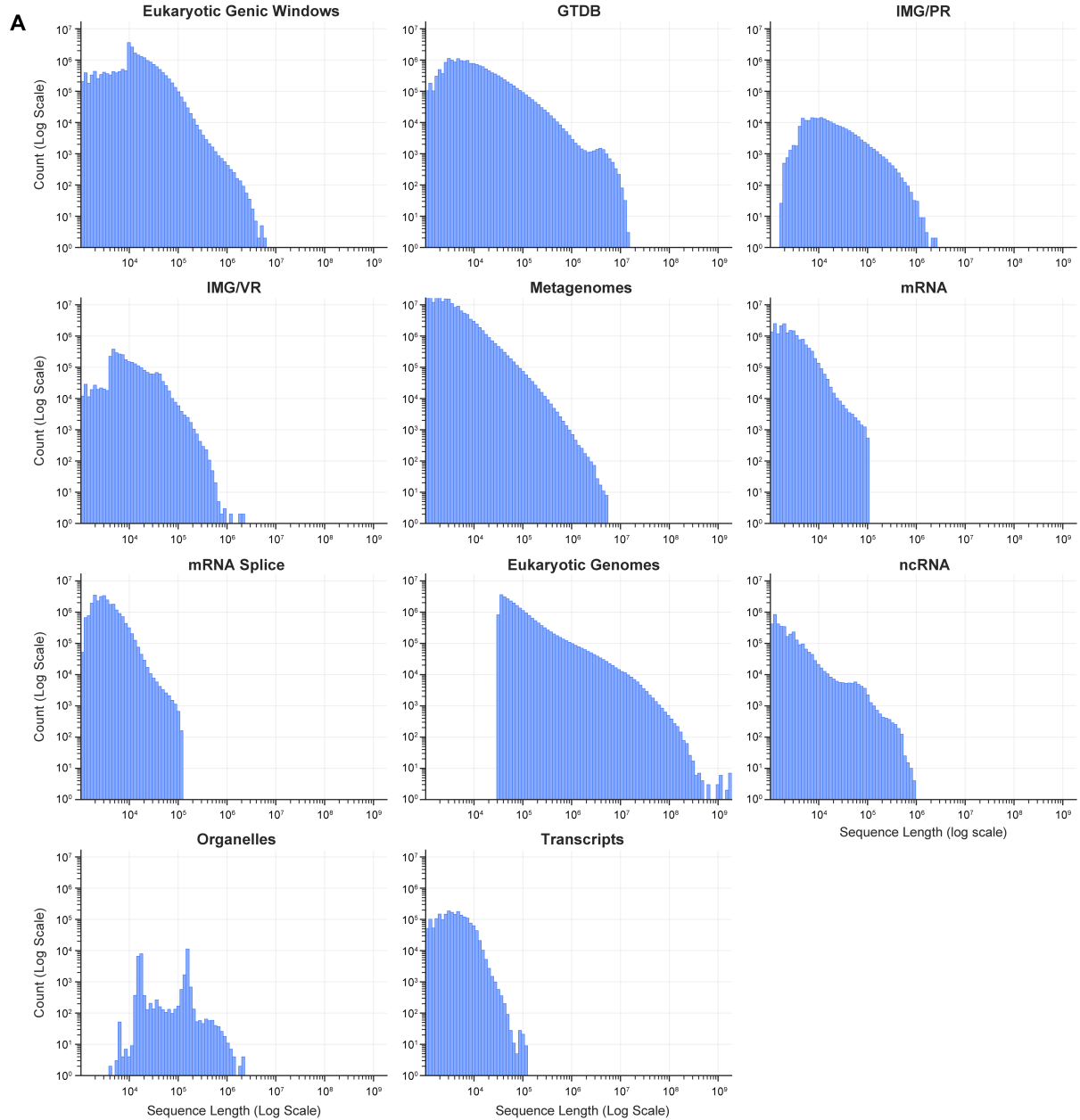
**Data Distribution Shift.** OpenGenome2 is composed of multiple sub-datasets, each with a distinct sequence-length distribution (Table 13 and Supplementary Figure B.3). Sub-datasets such as mRNA, ncRNA, organelles, and transcripts are heavily underrepresented at longer sequence lengths, whereas sub-datasets like whole genomes are overrepresented. Consequently, the data distribution encountered during the 8 kb phase, which excludes whole genomes, differs substantially from later phases dominated by whole-genome sequences (Panel Figure 2C, Supplementary Figure B.3).

We found that extended training on longer sequences caused NTv3 650M (old pre) to specialize for the long-sequence distribution, leading to forgetting of shorter-sequence sub-datasets. This effect is evident in the evaluation results. We find performance on short-sequence sub-datasets (mRNA, ncRNA, organelles, and transcripts) declines in later training stages, while performance on long-sequence sub-datasets (whole eukaryotic genomes) continues to improve (Supplementary Figure B.4). These observations underscore the critical role of data distribution in genomic language model pretraining and highlight the need to account for sequence-length composition when designing training curricula.

**Short-Context Forgetting.** Beyond distributional shifts, we investigated whether additional mechanisms may contribute to the degradation of performance on short sequences. Specifically, we asked whether poor short-sequence performance could arise even in the absence of distributional under-representation. Our analyses indicate that distributional effects alone do not fully explain the observed degradation.

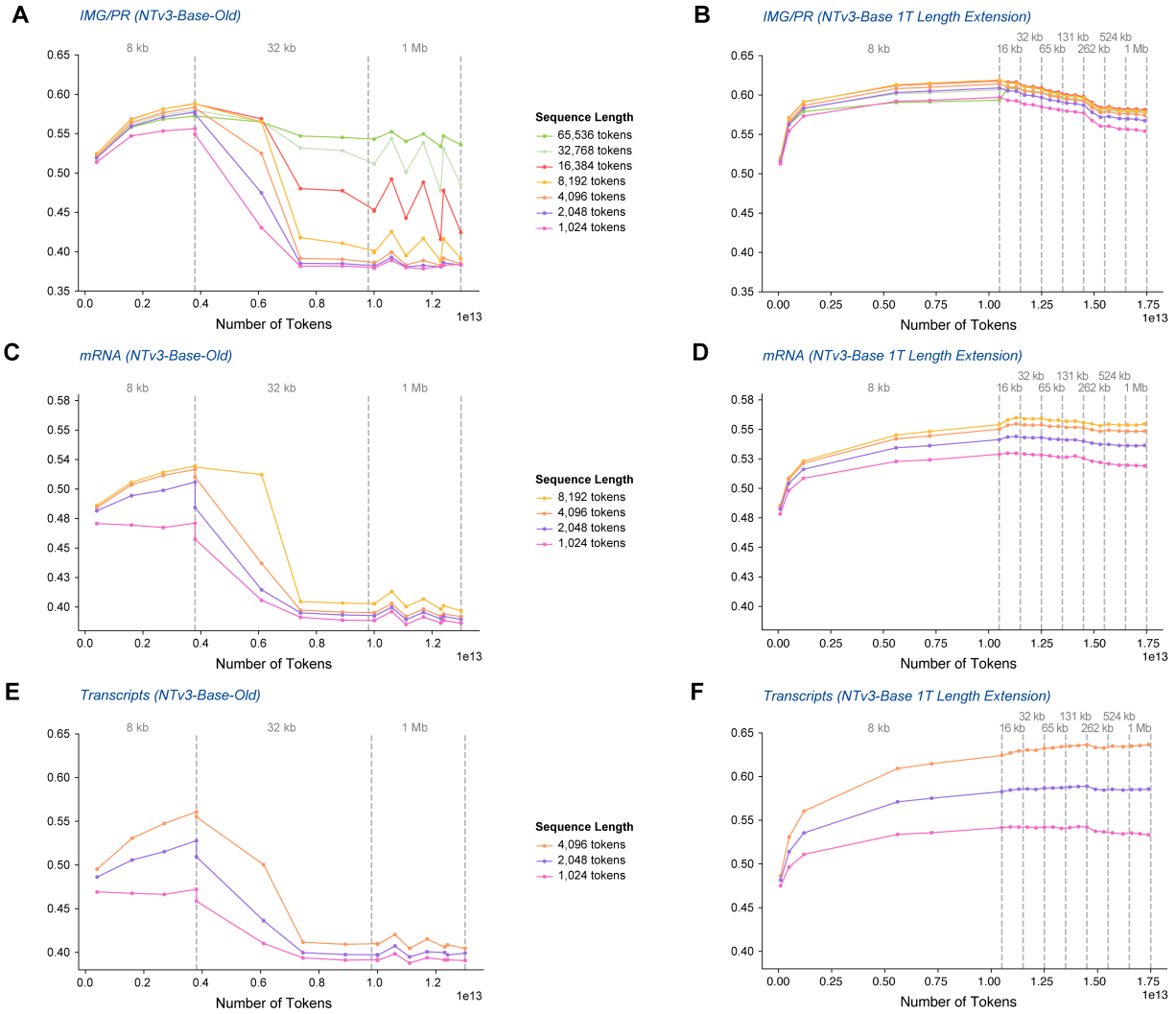
Even for sub-datasets that are well-represented and dominate the long-sequence distribution, particularly whole genomes, performance on short sequences deteriorated in the later training phases (Supplementary Figure B.5). This effect cannot be explained by a change in distribution as, in the case of whole genomes, this distribution was only introduced in the final stages of the pre-training of NTv3 650M (old pre). That is, we see performance degradation on short sequences in the absence of any changes in training data distribution.

This effect is most pronounced in evaluations of whole-genome sequences. In fact, we even observe that checkpoints from the 8 kb phase consistently outperform later-phase checkpoints when evaluated on short windows of whole genomes. This is especially surprising because there are no whole genome sequences in the 8 kb phase, whereas later phases are dominated by whole genome sequences (Supplementary Figure B.4A). The most likely explanation is that following the 8 kb phase, NTv3 650M (old pre) has lost the ability to effectively process short contexts. In total, these results suggest that prolonged training on long sequences may inherently compromise the model’s ability to retain information from shorter contexts. This phenomenon may be especially strong given our architecture choice (U-Net) and the masked language modeling objective. By contrast, autoregressive models naturally train on all contiguous subsequences up to the current sequence

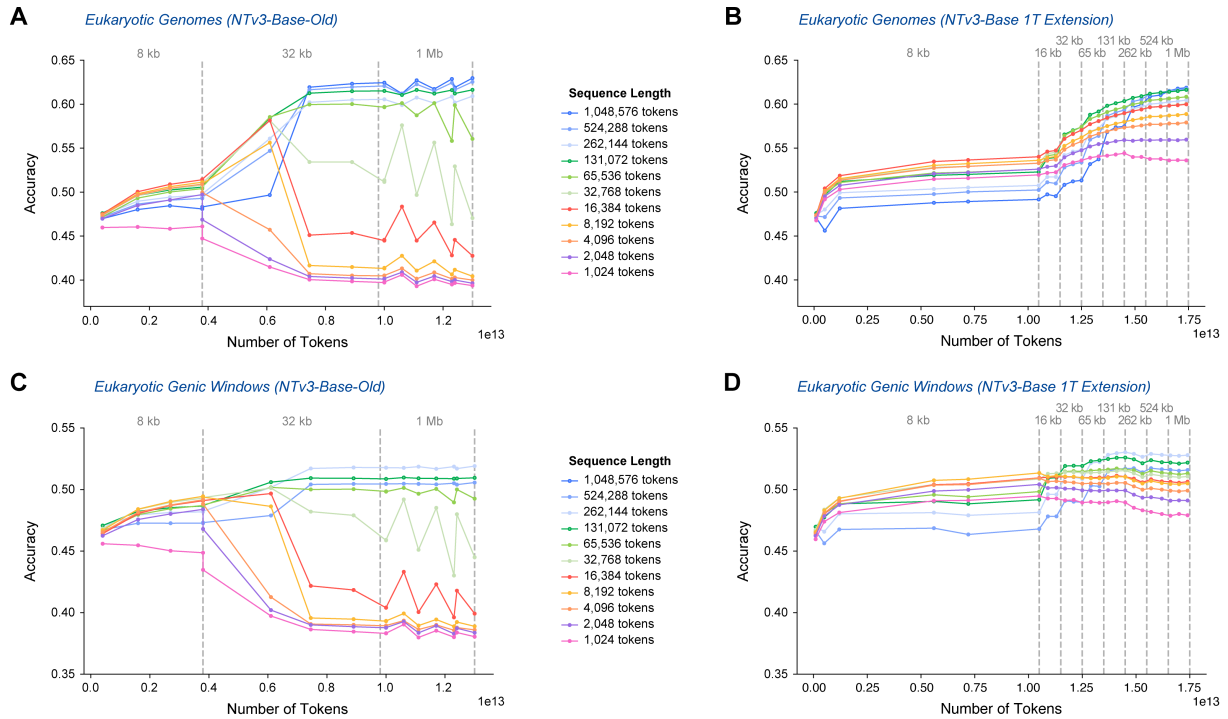


**Supplementary Figure B.3 | OpenGenome2 Sequence Length Distribution by Sub-Dataset** Histogram plots for each of the 11 OpenGenome2 sub-datasets in the pre-training corpus of NTv3. Plots are shown on a log-log scale to capture the orders-of-magnitude variation in both sequence lengths and counts. All sequences were filtered to have a minimum length of 1,024 bp (hence the x-axis beginning at  $10^3$ ), and eukaryotic genomes were further filtered to sequences of at least 32 kb, as reflected in the distribution.

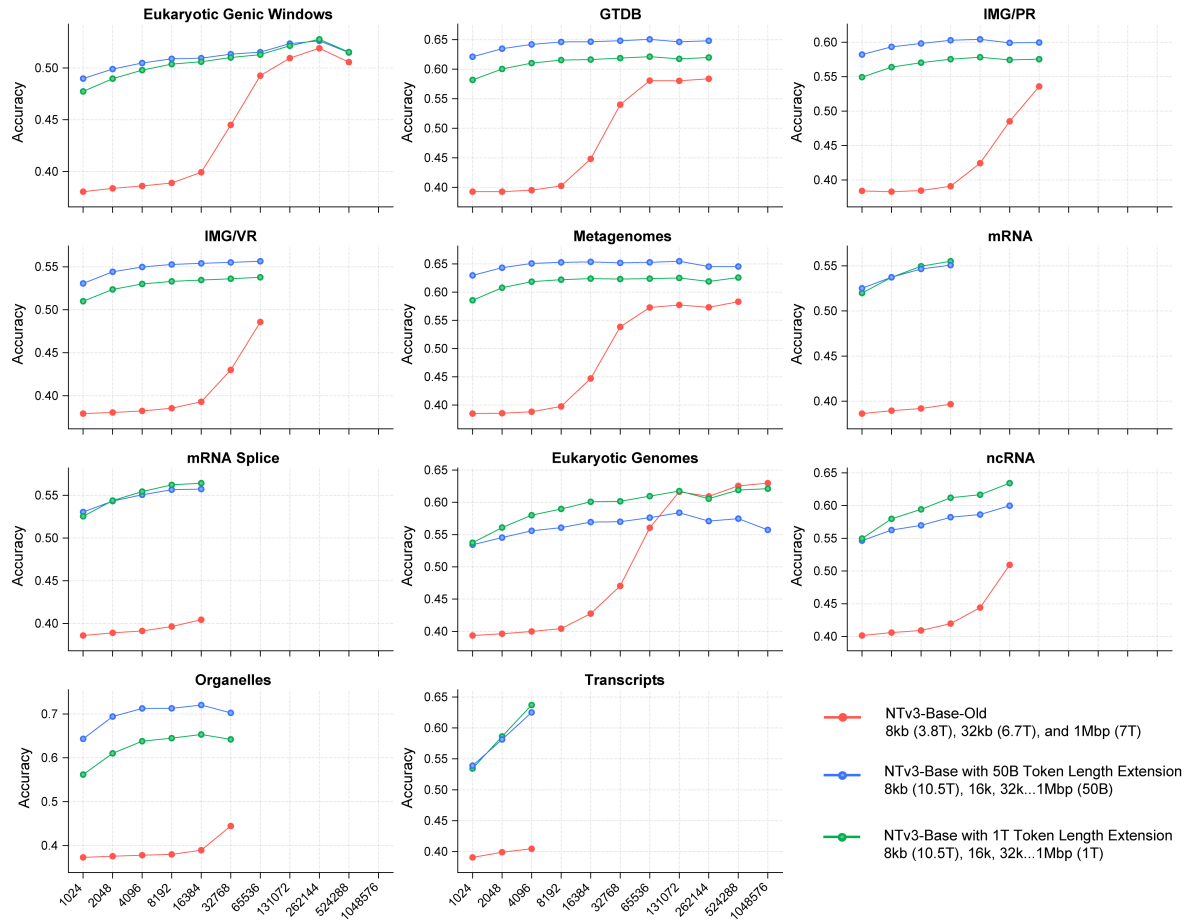




**Supplementary Figure B.4 | Effect of Sequence Length Mixing on Long-Context Generalization** Models trained without sequence-length mixing (NTv3 650M (old pre)) show noticeable degradation when evaluated on sub-datasets whose distribution are no present at long contexts, whereas models trained with 1T-token sequence-length mixing (NTv3 650M (pre)) mitigate the impacts of distribuion shift. **A)** NTv3 650M (old pre) on IMG/PR shows degradation after long-context training. **B)** NTv3 650M (pre) with 1T-token sequence-length mixing on IMG/PR shows minimal degradation. **C)** NTv3 650M (old pre) on mRNA also shows degradation. **D)** NTv3 650M (pre) with sequence-length mixing on mRNA remains stable. **E)** NTv3 650M (old pre) on transcripts also shows degradation. **F)** NTv3 650M (pre) with sequence-length mixing mitigates performance degradation on transcripts.



**Supplementary Figure B.5 | Short Context Forgetting** Models trained without sequence-length mixing (NTv3 650M (old pre)) degrade on shorter contexts, even on distribution that make up the majority of the later stages of training (Eukaryotic Genomes and Eukaryotic Genomic Windows). Models trained with 1T-token sequence-length mixing (NTv3 650M (pre)) maintain performance. **A)** NTv3 650M (old pre) shows degradation on short sequences of Eukaryotic Genomes. **B)** NTv3 650M (pre) with sequence length mixing retains performance on short Eukaryotic Genomes sequences. **C)** NTv3 650M (old pre) shows degradation on short sequences of Eukaryotic Genomic Windows **D)** NTv3 650M (pre) with sequence length mixing maintains performance on short Eukaryotic Genomic Windows sequences.



**Supplementary Figure B.6 | Sequence-Length Extension** A) Sequence-length mixing mitigates or completely eliminates short-context forgetting observed in NTv3 650M (old pre). Even after seven stages of 1T-token length extension, NTv3 650M (pre) exhibits very little degradation. Models with 50B-token stages of length extension show less degradation on sub-datasets primarily composed of short sequences, but under-perform the longer length extension models on sub-datasets primarily composed of long sequences (Eukaryotic Genomes).

length, which could explain their improved ability to generalize to varying sequence lengths.

**Mixed-Length Pretraining Mitigates Short-Context Forgetting.** To mitigate the effects of short-context forgetting and distributional shifts, we implemented mixed-length pretraining, in which sequences of varying lengths are continuously sampled throughout training, including in the sequence length extension phase as the model is processing the longest context lengths.. The exact procedure is detailed in Sections 4.2.1 and 4.2.2. This approach ensures that the model maintains exposure to shorter sequences while simultaneously learning from long-range contexts.

To assess whether mixed-length pretraining mitigates short-context forgetting, we examined the performance on short-sequence windows of whole genomes and eukaryotic genic sub-datasets, which were previously found to degrade under sequential-phase training. We find that training on a continuous mixture of sequence lengths preserves short-context performance while still enabling efficient learning of long-range dependencies. We compare the original sequential approach of NTv3 650M (old pre) to a sequence length ablation where we spend over 7 Trillion tokens on sequence length extension, thus giving substantial time for short sequence length degradation were it to occur. The experiment, and its motivation, is described in detail in Supplementary Section B.4. Despite such a significant time spent on a long sequence, we find these sub-datasets exhibit little to no degradation on shorter sequences, in contrast to the original training approach (Supplementary Figure B.5B and D).

Mixed-length pretraining also mitigates the effects of distributional shift caused by the unequal representation of sub-datasets across sequence lengths. Sub-datasets that are underrepresented at long sequence lengths, such as mRNA, mRNA Splice, ncRNA, organelles, and transcripts, maintain substantially higher masked-token accuracy under the mixed-length training scheme. Even when we extend the length extension to 7 trillion tokens, under-represented sub-datasets perform well despite comprising a small fraction of the training data. Accuracy on these short-sequence sub-datasets increased from as low as 40% under sequential-phase training to 50-60% under the sequence length mixing (Supplementary Figure B.4B, D and F).

Short sequences remain a small percentage of the long-sequence-dominated data distribution (Table 2), so mixed-length pretraining does not fully eliminate distributional effects. Nevertheless, by continuously sampling sequences of varying lengths, the model retains exposure to underrepresented sub-datasets, reducing performance degradation relative to sequential-phase training. Future work could further improve this balance through adaptive sampling strategies or more carefully designed curricula that explicitly account for the full spectrum of sequence lengths.

### B.3.2. Post-training

A similar phenomenon was observed during post-training on functional tracks and genome annotations. In a controlled experiment, we compared models trained either at a fixed sequence length of 131 kb or with a mixed-length schedule up to 131 kb. The mixed-length schedule sampled sequences of length 16 kb, 32 kb, 65 kb, and 131 kb, with weights of 10% assigned to each of the three shorter lengths and 70% to 131 kb. All models were initialized from the final stage of length extension described in Section 4.2.2 and trained for 300 B tokens under identical conditions to the post-training setup described in Section 4.3. Evaluation was performed on a fixed held-out set, partitioned into chunks matching the sequence length under consideration. Metrics were computed over the central 37.5% of each sequence, using a 62.5% overlap to ensure every nucleotide contributed to evaluation.

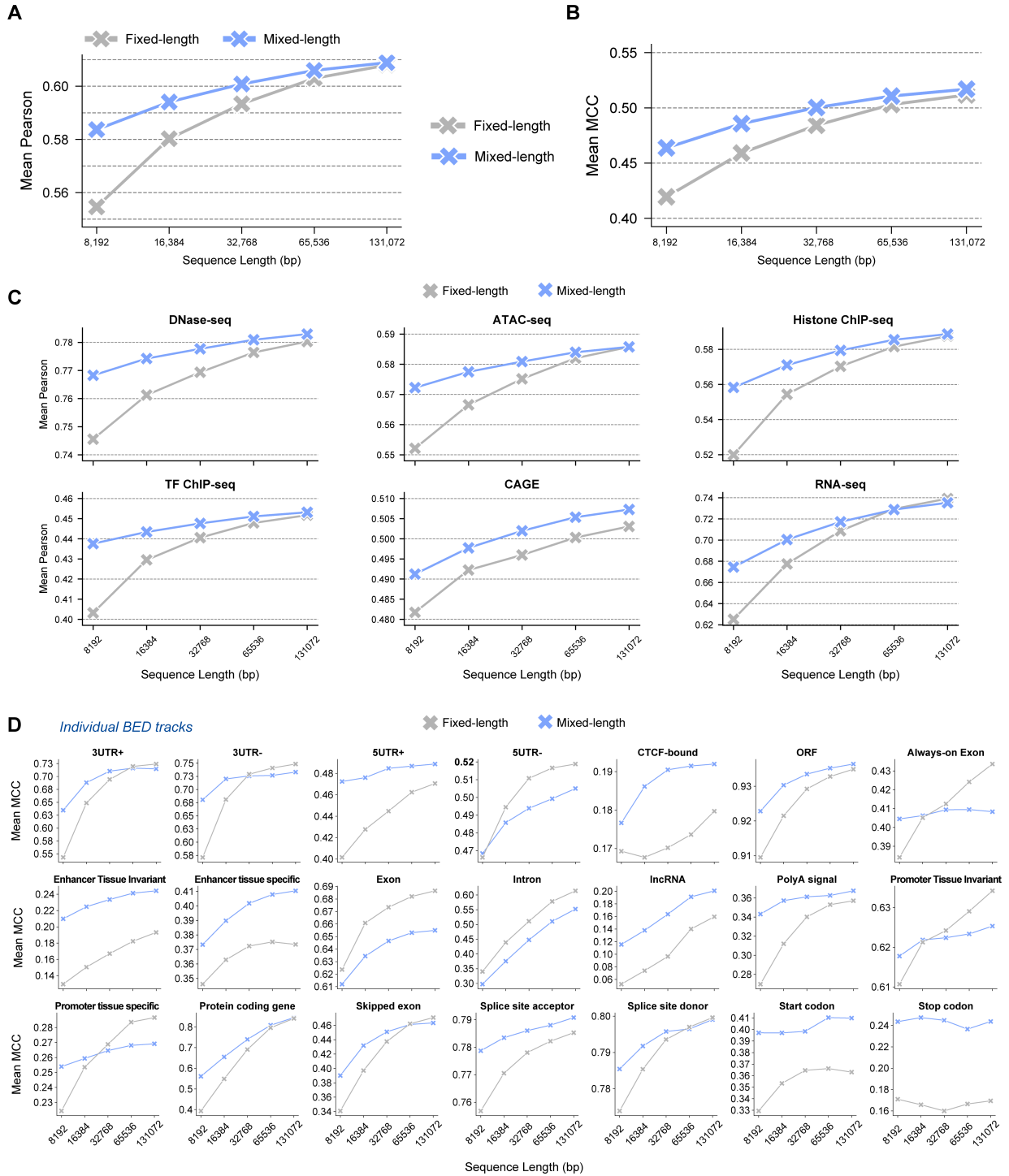
Mixed-length post-training substantially preserved supervised performance on short sequences for both functional and annotation tracks, without reducing performance at longer lengths. For functional tracks, we report the mean Pearson correlation across DNase-seq, ATAC-seq, histone ChIP-seq, TF ChIP-seq, CAGE, and RNA-seq. The mixed-length protocol yielded consistent improvements at short contexts across all assays, with the largest gains in RNA-seq and ChIP-seq (Supplementary Figure B.7A,C).

We observed similar trends for annotation tracks. For most of the 21 evaluated tracks, the mixed-length strategy markedly outperformed fixed-length training at short contexts and matched performance at longer contexts. Tracks that benefited from mixed-length post-training, including Enhancer Tissue-Invariant/Specific, Start/Stop Codon, Long Non-Coding RNA, ORF, Splice Site, and 5UTR+, showed improvements across all sequence contexts, including the longest lengths, despite receiving fewer training tokens at those lengths. A small subset of tracks exhibited slightly lower performance under mixed-length training, most notably Intron/Exon and 5UTR- (Supplementary Figure B.7B and D).

Taken together, these results demonstrate that sequence-length mixing is an effective strategy for preventing short-context forgetting during downstream supervised fine-tuning.

### B.3.3. Implications

Our experiments indicate that catastrophic short-context forgetting can occur when models are trained primarily on long sequences, and that distribution shifts may exacerbate this effect. This phenomenon may be influenced by characteristics of the model architecture, such as the UNET-based design, or by the masked language modeling training objective, which, unlike auto-regressive objectives, does not explicitly expose the model to all shorter subsequences. We find continuous exposure to shorter sequences, achieved through mixed-length training, mitigates short-context forgetting and ensures the model remains robust across the full range of sequence lengths. Understanding the relative contributions of architectural and training factors which contribute to short context forgetting remains an important direction for future research.



**Supplementary Figure B.7 | Impact of Mixed Sequence Length on Post-Training** All figures compare models trained for 400B tokens either using sequence length mixing (as described in [subsubsection 4.3.1](#)) or using a fixed length of 131 kb. **A)** The mean Pearson over all human functional tracks at varying sequence lengths. **B)** The mean MCC over all human annotation tracks at varying sequence lengths. **C)** Mean Pearson correlation per assay for human functional tracks. **D)** Human genome annotation MCC for each element.

## B.4. Pre-Training Sequence Length Extension

Training on extended sequence lengths is prohibitively expensive for standard transformer-based approaches. NTv3, however, enables efficient training at long contexts, providing the opportunity to capture long-range interactions and leverage the abundance of whole-genome data that prior models could not fully exploit.

Motivated by this capability and the success of shorter 50B-token pretraining stages, we scaled up the length extension phase significantly. In this ablation, we conducted a length-extension phase on a version of the NTv3 650M (pre) model taken after 8kb-training. We do a length extension using 1 trillion tokens per stage for a total of 7 trillion tokens. This brought the cumulative training data to over 17 trillion tokens.

### B.4.1. Evaluation

We evaluated the model on the OpenGenome2 dataset, measuring performance across multiple sequence lengths and each sub-dataset. Checkpoints throughout the training process were assessed on a fixed held-out set to track performance evolution during both pretraining and length extension. Details on the construction of the held-out set follow the mixed-length pretraining experiments, and are described in detail in Section B.3. The complete composition of the OpenGenome2 sub-datasets during training is shown in Figure F of Panel 2, and we also provide a table with number of tokens present at each sequence length for each of the OpenGenome2 sub-datasets (Table 13).

### B.4.2. 7 Trillion Token Length Extension: Performance Across Sequence Lengths and Distributions

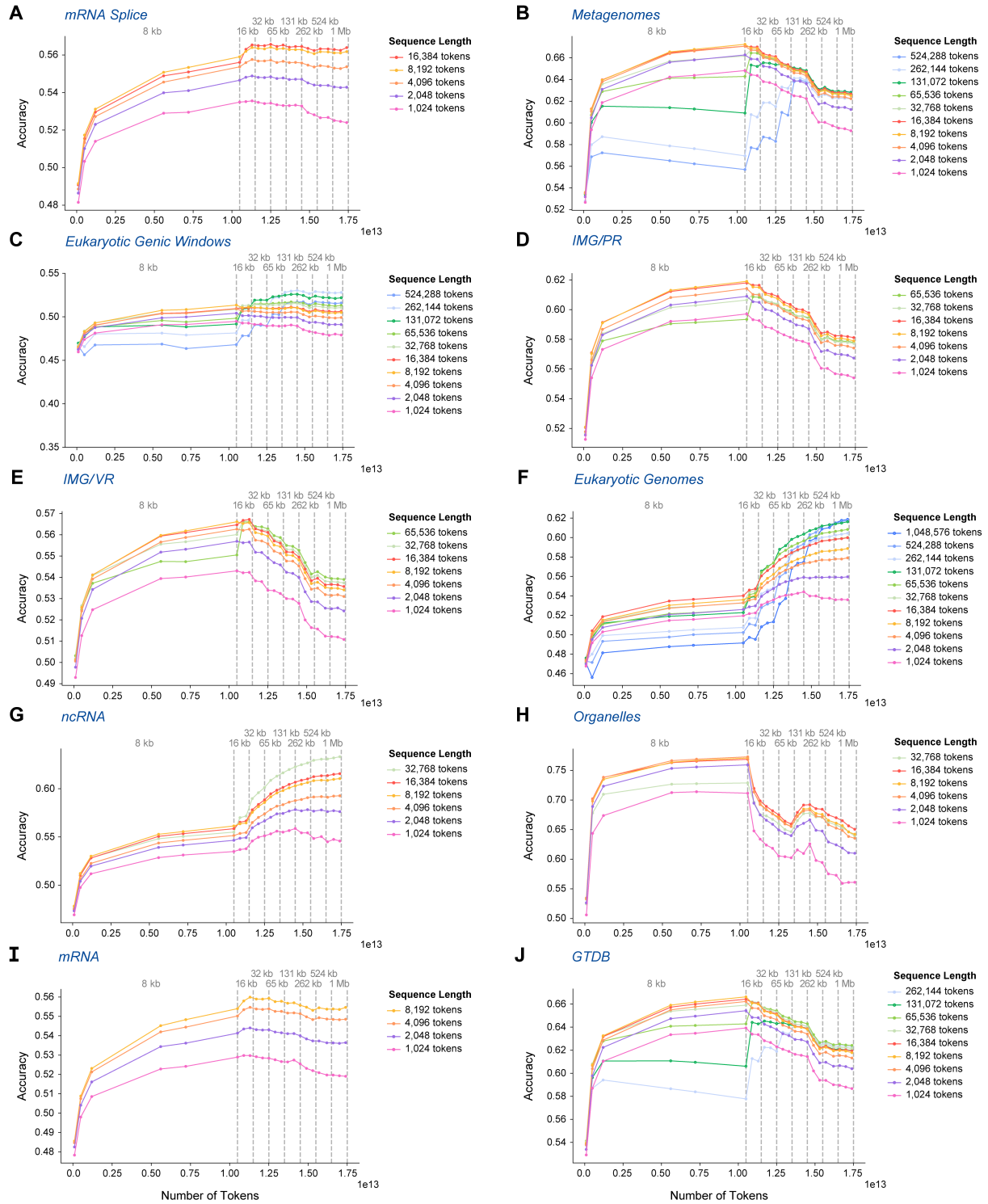
Consistent with observations from the shorter 50B-token stages, in the 1 trillion sequence length extension phase, the model adapts rapidly to newly introduced sequence lengths, quickly achieving performance comparable to shorter sequence. For sub-datasets containing abundant long sequences, such as whole genomes, extended training led to continued improvements, with peak performance observed at the longest sequence lengths and later stages of training (Supplementary Figure B.8F). In contrast, sub-datasets lacking long sequences, including ncRNA, mRNA, transcripts, and others, exhibited increased performance degradation (Supplementary Figure B.8). This degradation occurred despite sequence-length mixing, which heavily favored the longest sequences: in the later stages of the length-extension phase, over 50% of tokens were sampled from the two longest sequence lengths (2). Most long sequences are drawn from whole genomes (Table 13), so extended training disproportionately emphasizes this distribution, reducing performance on underrepresented sub-datasets that primarily appear in shorter sequences.

Our choice of weighting schedule was driven by the assumption that the model should primarily observe the newly introduced longest sequence lengths to rapidly adapt to them. However, the combination of quick adaptation to new sequence lengths and the observed degradation of short-sequence sub-datasets suggests that a smaller fraction of tokens could be allocated to the longest sequences. Future work could define sequence sampling primarily based on the data distributions deemed most critical rather than allocating a large fraction of tokens to the longest sequences. For architectures that efficiently handle long-range contexts, this approach could enable prolonged mixed-length training without the need for distinct length-extension phases, simply by specifying the desired sampling probabilities for each distribution and sequence length.

### B.4.3. Choice of Length Extension Phase

Although performance on whole genomes improved, the combined effect of degradation on short-sequence sub-datasets and the high computational cost of extended training motivated our decision not to apply the full 1T-token length extension. Instead, we relied on the shorter 50B-token stages which showed less degradation on the OpenGenome2 sub-datasets which are underrepresented at long contexts (Supplementary Figures A.2, A.3 and A.4). These findings highlight a key consideration in long-sequence training: realizing gains from extended context lengths depends critically on the composition of the training data. Over-representation of certain regions, such as whole genomes, can introduce distributional shifts that degrade performance on underrepresented sub-datasets, while under-representation of functionally important regions may limit overall model utility. As demonstrated previously [28], careful curation and weighting of the pretraining dataset, focusing on functionally enriched regions, can enhance downstream task performance even when long sequences are available.





**Supplementary Figure B.8 | 1 Trillion Token Stage Sequence Length Extension** A–J) Accuracy of NTv3 650M (pre) over the course of pre-training and seven stages of 1T-token length extension on each OpenGenome2 sub-dataset. Models were evaluated at multiple sequence lengths on a fixed held-out set of each sub-dataset.

For architectures capable of efficiently processing long-context whole-genome sequences, additional gains may be achieved through extended training, provided that sequence length schedules and dataset distributions are carefully balanced. These results underscore the importance of both model architecture and data engineering in realizing the benefits of long-context pretraining for DNA sequences.

### B.5. Multi-species Functional Tracks Head Implementation

Post-training uses a linear head for each species to predict the number of functional tracks for that species. During the first phase of post-training, 8 samples of 131 kb are simultaneously processed using data parallel (i.e. one sample per device). Since the samples are randomly picked from the different species, the batch normally contains data from multiple species.

Requiring the model to use different heads for different samples in the batch is not possible with JAX’s distributed jit compilation. JAX’s distributed jit compilation follows the single-program-multiple-data (SPMD) paradigm, meaning that every device must perform exactly the same computation, irrespective of the data involved.

Each species has a tracks-head that predicts the number of tracks for that species. Our solution is to pad the output logits from each head to the maximum number of tracks in any species, in our case, this is human with 7326 tracks. The multi-species head initialises a logit array with the padded size of the logits for each species. Then the head loops through each species and computes the logits. A mask is used to determine if current head correlates to the samples species. Then the head logits are multiplied by the mask and added to the output logit array.

This multi-species head ensures that exactly the same computation is performed on all devices irrespective of the species. This results in the code being suitable for the SPMD paradigm. A limitation of this approach is the inefficiency in calculating the logits for every single species. However, it is not possible to avoid this when using the SPMD framework.

It is not possible to compute the logits for all tracks in a single pass because of memory limitations when using a large number of tracks (NTv3 used around 16k).

---

```

1
2 def multi_species_tracks_head(x: Array, species_idx: Array,
3                               num_tracks_per_species: list[int]) -> Array:
4     # Determine maximum number of tracks across species for padding
5     max_tracks = max(num_tracks_per_species)
6     # Initialize output logits, padded to max_tracks
7     logits = zeros(x.shape[:-1] + (max_tracks,))
8     for i, num_tracks in enumerate(num_tracks_per_species):
9         # Compute species-specific predictions
10        species_logits = tracks_head(x, num_tracks)
11        # Pad to maximum tracks
12        species_logits = pad(species_logits, (0, max_tracks - num_tracks))
13        # Mask to select predictions for current species
14        mask = (species_idx == i)[:, None, None]
15        logits = where(mask, species_logits, logits)
16    return logits
17

```

---

### B.6. NTv3 benchmark

In this section, we outline the NTv3 BENCHMARK rationale, including the methodological choices adopted to ensure fair, efficient, and biologically meaningful evaluation across species. We describe the implementation details across JAX, DeepSpeed, and BioNeMo pipelines. We then summarize the main conclusions drawn from benchmarking the different models on downstream tasks.

#### B.6.1. Motivation

The NTv3 BENCHMARK serves as a standardized evaluation framework to compare SOTA models and assess the transferability of post-trained genomic representations to downstream prediction tasks. By spanning seven species across both animal and plant kingdoms, the benchmark enables evaluation of model performance across

diverse phylogenetic contexts and varying genome sizes. For species encountered during post-training, we exclusively use regulatory tracks absent from post-training supervision, thereby isolating the contribution of learned representations from direct signal reuse. For species not seen during post-training, the benchmark provides a direct test of cross-species generalization capabilities. This dual-strategy approach ensures that performance differences reflect genuine improvements in representation quality rather than artifacts from supervision overlap, while simultaneously offering a comprehensive assessment across both familiar and novel genomic contexts.

### B.6.2. Design choices

**Stopping criterion.** To ensure fair comparison across species with varying genome sizes, we define an epoch as a complete pass over the training portion of each species’ genome, and train models for 10 epochs. This approach normalizes the training budget relative to genome size, ensuring that models are not disadvantaged by the availability of species-specific data. Analysis of training and validation curves across multiple species and tasks revealed that 10 epochs provide sufficient training for models to reach convergence. The choice of 10 epochs balances computational efficiency with ensuring that all models have adequate opportunity to learn task-specific patterns. Table 14 summarizes the training token budgets for each species.

Species	Genome Split Size (Nucleotides)			Training Tokens (10 epochs)	Tokens per Validation (Half epoch)
	train	val	test		
Human	2.09B	345.58M	345.49M	20.9B	1.05B
Chicken	947.28M	52.61M	52.60M	9.47B	473.64M
Cattle	2.64B	55.27M	55.57M	26.4B	1.32B
Arabidopsis	106.15M	6.00M	7.00M	1.06B	53.01M
Maize	2.01B	59.11M	59.22M	20.1B	1.01B
Rice	333.21M	20.02M	20.02M	3.33B	166.61M
Tomato	799.82M	8.46M	8.46M	8.0B	399.9M

**Table 14 | Training token budgets per species.** Training tokens are calculated as  $10 \times$  the genome (nucleotides) in the training split. Validation and checkpointing occur every half-epoch, corresponding to 5% of the total training budget. Genome split sizes show the number of unique nucleotides in each split (train, validation, test) for each species.

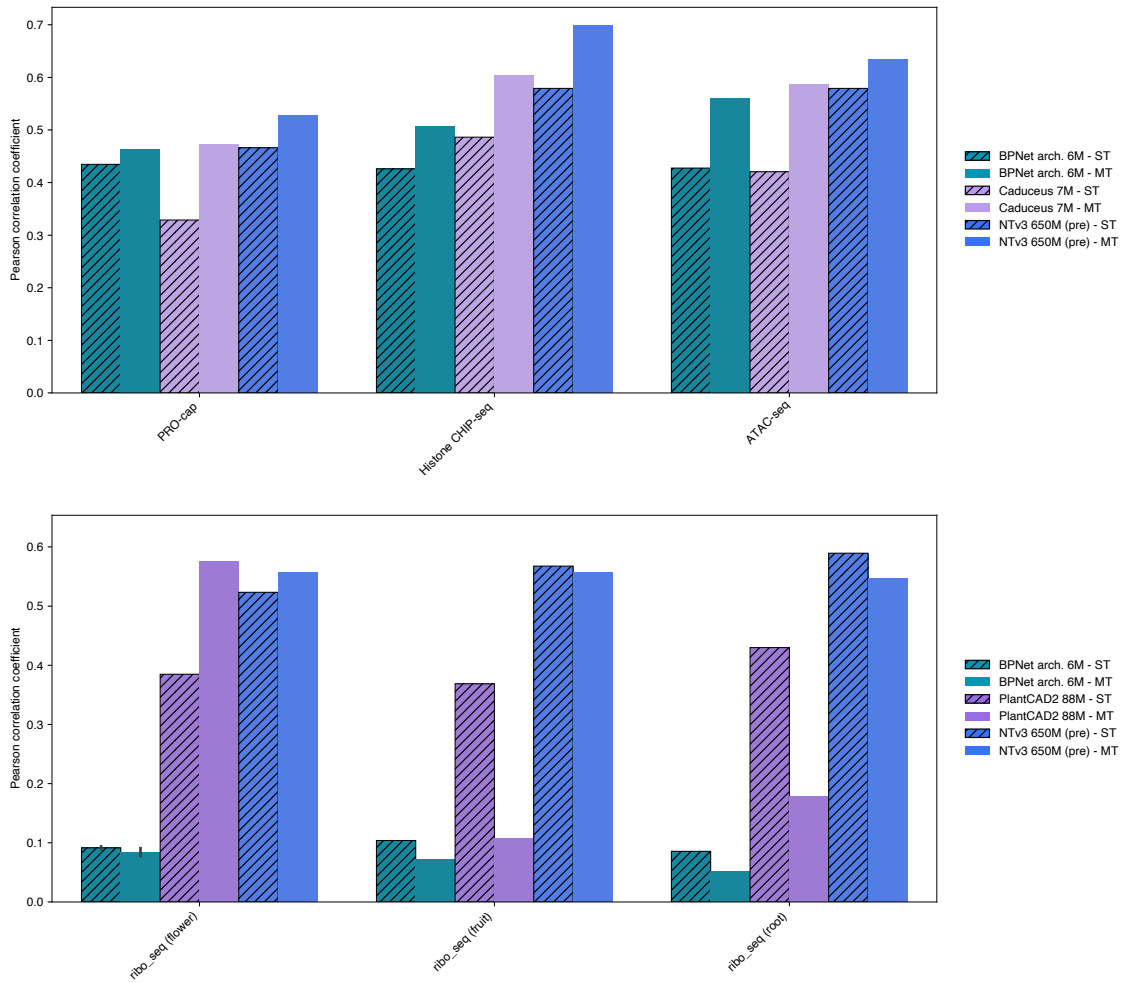
**Multi-tasking for functional tracks prediction.** The NTV3 BENCHMARK evaluate models with different strategies for functional and genome annotation tracks. Genome annotation tracks are evaluated individually, whereas functional tracks are evaluated jointly in a multi-task learning setup. This design choice for functional tracks is motivated by computational efficiency and the biological observation that different regulatory signals (e.g., ATAC-seq, RNA-seq, ChIP-seq) are often correlated and can benefit from joint learning.

To validate that multi-task functional tracks training achieves comparable performance to single-task training, while being more efficient, we conducted preliminary experiments comparing models trained on several tracks simultaneously (multi-task) versus models trained on individual tracks separately (single-task) on human and arabidopsis functional tracks. Test set evaluations demonstrate that multi-task training achieves similar or better performance compared to training separate models for each track, while requiring significantly less computational resources (Supplementary Fig. B.9). Based on these findings, we adopt multi-task training as the standard approach for functional track prediction in the benchmark, ensuring computational efficiency while maintaining evaluation quality.

**Multi-framework implementation details.** The JAX pipeline was used to fine-tune the NTV3 suite of models and CNNs baselines. In order to evaluate competing models on NTV3 BENCHMARK, we decided to implement a torch-based pipeline compatible with various pretrained models available on Hugging Face, using DeepSpeed as a framework to accelerate training.

However, to our knowledge, no Evo2 fine-tuning pipeline has been implemented. Following authors’ suggestion on the public inference repository (<https://github.com/NVIDIA/bionemo-framework/issues/1170>), we implemented the fine-tuning pipeline using BioNeMo framework.

All pipelines maintain identical core components: (1) data pre-processing using the same sequence extraction and track processing functions, (2) loss computation using the same Poisson-multinomial loss for functional tracks and weighted focal loss for genome annotation tracks, and (3) backend-specific metrics definitions.



**Supplementary Figure B.9 | Comparison of multi-task vs single-task learning on functional tracks. (A)** Pearson correlation on 3 human tracks. **(B)** Pearson correlation on 4 Arabidopsis tracks. **MT** refers to multi-task models while **ST** refers to single-task models.

Furthermore, we maintained a similar optimization scheme across pipelines, ensuring comparable training dynamics.

### B.6.3. Main conclusions

The NTV3 BENCHMARK provides a comprehensive evaluation framework that reveals several key insights about genomic foundation models. First, post-training significantly improves downstream task performance across all model architectures, demonstrating the value of representation learning for genomic applications. Second, larger models generally achieve better performance, though the relationship between model size and performance varies across species and track types. Third, models show strong performance on species seen during post-training, with more variable results on unseen species, highlighting both the benefits of species-specific training and the challenges of cross-species generalization. Finally, the benchmark demonstrates that multi-task learning is an effective and efficient approach for genomic track prediction, achieving comparable performance to single-task training while reducing computational costs.

## B.7. Model interpretation

### B.7.1. Future Directions for Model Interpretability and Development

Our analysis highlights several avenues for enhancing the interpretability and utility of deep sequence models. First, incorporating explicit conditioning tokens (e.g., cell-type or species embeddings) at the input stage could improve the resolution of attention mechanisms. Currently, as shown in Fig. 6B, our attention maps and predicted functional annotations aggregate signals across all cell types, which—while accurately aligning with the ENCODE cCRE annotations across cell types—can obscure cell-type-specific regulatory logic. Conditioning tokens would allow the model to dynamically mask irrelevant interactions and more cleanly reveal the regulatory grammar active in a single cellular context. Relatedly, although we use the mean attention map from the final multi-head attention layer for interpretability, a richer analysis should examine how regulatory information is propagated across layers and distributed among heads. Such a decomposition—tracking which heads specialize in motif-level, chromatin-scale, or long-range enhancer–promoter interactions, and how these roles evolve through depth—would offer a more mechanistic view of how the model integrates regulatory cues.

A second interpretability challenge arises in the predicted genome annotation probabilities for exon and intron states, which occasionally appear simultaneously elevated in the same genomic region. In most cases where this non-exclusive behavior occurs, the corresponding GENCODE annotations show substantial isoform heterogeneity—regions annotated as exonic in some transcripts but intronic in others. Because our post-trained model was optimized under a one-to-one mapping between each input sequence and an annotation track, it effectively learns a collapsed, isoform-averaged representation of these ambiguous regions. As a result, the model may express uncertainty by activating multiple mutually exclusive states, reflecting the latent structural diversity it was not explicitly trained to disentangle. Future extensions could address this limitation by training on transcript-resolved annotations, using multi-label or soft-target supervision to represent isoform mixtures, or incorporating probabilistic decoders that explicitly model transcript-specific structure rather than forcing a single deterministic label per position.

A final direction for enhancing interpretability lies in expanding the diversity and granularity of supervised genomic annotations used during post-training. For instance, a model trained on a richer set of transcription factor (TF) occupancy labels—spanning distinct TF families, cell types, and species—would allow attribution-based motif discoveries to be corroborated directly through explicit prediction heads. Such annotations could serve as a form of grounded interpretability, where motifs illuminated by backpropagation are immediately validated by corresponding TF-specific outputs, reducing the burden on deeper mechanistic probing and increasing confidence in causal inferences. This would also mitigate the methodological ambiguity introduced by the wide range of available attribution techniques and the many ways they can be summarized (e.g., profile-summed, enhancer-restricted, or window-specific), which currently create variability in mechanistic interpretation. Relatedly, it would reduce dependence on downstream motif-mining tools whose outputs can vary substantially depending on hyperparameters and method-specific limitations. In practice, however, the availability of high-quality TF binding data remains highly uneven: ChIP-seq and CUT&RUN datasets are sparsely distributed across cell types, often limited to a few dozen canonical regulators, and far less comprehensive in non-human species. As broader, multi-context TF profiling becomes increasingly feasible, integrating these annotations into training pipelines may yield models whose internal regulatory abstractions align more directly with experimentally defined transcriptional grammars, thereby reducing the reliance on post hoc interpretability methods and making the models themselves more intrinsically interpretable.

### B.7.2. Methodological Considerations for Interpretable Variant Prioritization

To leverage NTv3's expansive cross-species and multi-modal training for variant prioritization, we anchored our analysis on ground-truth expression quantitative trait loci (eQTLs) from the GTEx project (Human Whole Blood) and developed a differential framework (Fig. 6E). While our global analysis demonstrated that pathogenic variants yield significantly larger attribution differences than benign controls (Wilcoxon  $p < 0.001$ ; Fig. 6F), the distributions also indicated that the model is most expressive for a specific subset of pathogenic candidates. To maximize the model's utility, we focused on mechanistically characterizing these "putative true positives"—variants that drive substantial rewiring in the model's internal representations—while distinguishing them from cases with negligible attribution shifts. We therefore developed a ranking framework to curate the strongest candidates for detailed mechanistic interpretation.

**(i) Quantitative Signal Prioritization.** Given that the quantitative attribution-shift metric used in our global Wilcoxon analysis already separates high-impact pathogenic variants from benign controls, we begin our ranking framework by formalizing the rationale behind this metric and extending it into a broader quantitative layer for variant prioritization. The key observation is that although the genomic windows processed by NTv3 are large (up to 1 Mb), the reference and alternative alleles remain structurally aligned, sharing an identical regulatory context everywhere except at the mutation site. Because NTv3 internalizes this shared *cis*-regulatory background—chromatin context, nucleosome positioning, and neighboring TFBSs—as stable features across alleles, the variant-induced mechanistic change is decoupled from the background. This decoupling enables global distance metrics to detect focal disruptions without being overwhelmed by high-dimensional background variation, consistent with recent counterfactual attribution studies (e.g., SEAM [51]) showing that gradient-based saliency maps respond selectively to the causal nucleotide substitution while leaving the surrounding background attribution signals largely unchanged.

Ranking variants directly by the distance between their reference and alternative attribution maps can be computed using different distance metrics—including Euclidean distance ( $L_2$ ), cosine similarity, and Kullback–Leibler (KL) divergence—and each metric may optionally be evaluated over attribution windows of varying length centered on the SNV. Interestingly, we found that these distance metrics stratified variants differently as a function of window length. Euclidean distance yielded the highest AUROC for distinguishing pathogenic from benign variants at smaller windows (approximately  $\pm 50$  bp), consistent with its sensitivity to the magnitude of localized attribution changes. In contrast, cosine similarity achieved the highest AUROC when computed over the full attribution window, reflecting its scale invariance and its sensitivity to coordinated, directionally consistent shifts across broader regions. These complementary behaviors indicate that the metrics capture distinct scales of biological perturbation. In the case of Euclidean distances, the ranking provides an effective first-pass stratification of variants with sharp motif perturbations, such as the “motif ablation” and “motif creation” classes illustrated in Fig. 6G,H. The top-ranked candidates also illuminate a distinct class of “distributed” regulatory rewiring (Fig. 6I), in which pathogenic variants perturb regulatory logic across broader genomic regions rather than producing a single, localized burst—motivating the extension of this quantitative prioritization to spatially aware differential metrics.

To capture these spatially distributed attribution signals, we propose a secondary suite of ranking strategies centered on fold-change analysis. For attribution maps, we quantify the spatial diffusivity of the signal by computing positive and negative  $\log_2$  fold-change profiles between alternative and reference alleles (denoted as  $\sum \log_2 \text{FC}$  in Fig. 6J). Applying a signal-detection threshold to these differential profiles yields a “maximum spatial extent” metric—defined as the furthest genomic position from the central SNV retaining significant positive or negative signal—which provides an orthogonal ranking axis to the global distance measures. This differential approach extends to NTv3’s other latent and output modalities, enabling parallel rankings based on structural and functional rewiring. For attention, we compute the  $\log_2$  fold-change between reference and alternative attention matrices based on the last layer or optionally the layer with the largest absolute  $\log_2$  fold-change. For model outputs, variants are ranked by the  $\log_2$  fold-change in predicted functional tracks or by the fold-change in predicted genome annotations. Additional orthogonal metrics may also be explored—for example, computing the entropy of the attention maps to prioritize variants that induce highly coordinated or cooperative interaction patterns. Collectively, these metrics constitute the quantitative signal prioritization layer of our framework, isolating high-signal candidates for subsequent biological validation.

**(ii) Genomic Context and Functional Annotation.** Following quantitative prioritization, we integrated biological annotations to ground the model-derived signals in known genomic features. Using the Ensembl Variant Effect Predictor (VEP) [54] with GENCODE annotations, we cataloged for each prioritized variant the affected gene targets, variant consequence (e.g., splice donor region variant, intron variant, 3’UTR variant), and predicted impact scores (high, medium, low, modifier). We also captured transcript-level complexity by enumerating associated isoforms—from single transcripts to dozens of alternatively spliced variants—and the full set of associated genes, including bidirectional promoters and other multi-gene regulatory domains. These annotations filter the high-signal candidates to those with clear biological relevance, ensuring that the mechanistic changes inferred by NTv3 correspond to interpretable genomic elements rather than modeling artifacts. By cross-referencing the annotations with the quantitative rankings, we isolate variants where the model’s mechanistic prediction—such as the loss of a splice motif—aligns with the structural disruptions annotated by VEP, thereby increasing confidence in the model’s interpretation.

**(iii) Mechanistic Synthesis and Literature Validation.** In the final stage, we contextualized prioritized



variants within the broader scientific literature. We ranked candidates by their citation frequency in the NCBI database to distinguish well-studied variants from more novel discoveries. To ensure relevance to the biological system under study, we advise filtering retrieved citations to those containing tissue- or process-specific keywords—for example, in whole-blood models, terms such as “erythroid”, “hematopoietic”, “immune”, “leukocyte”, “myeloid”, “lymphoid”, “blood”, or phrases indicating multi-tissue or broadly acting regulatory effects. This filtering helps ensure that literature evidence is functionally aligned with the regulatory context modeled by NTv3. To bridge the gap between NTv3’s internal representations and established biology, we also cross-referenced the TFBS motifs identified as “ablated” or “created” in the attribution maps with the Tomtom database, thereby linking the model’s learned regulatory grammar to specific transcription factors and their known biological functions.

To streamline this process at scale, we envision AI agents capable of automatically retrieving and synthesizing relevant papers across a broader scope than traditional databases. Such agents could mitigate the limitations of structured indexing; for instance, we observed that supplemental manual searches (e.g., via direct web searches for rsIDs) often identified relevant functional studies or preprints not yet captured by NCBI. By connecting the molecular mechanisms proposed by NTv3 (e.g., “ablation of an ELK1 site”) with disparate evidence of a factor’s role in the relevant tissues or disease context, these agents could yield coherent, automated mechanistic reconstructions similar to our manual analysis for Fig. 6.

**(iv) Comprehensive Visualization and Reproducibility.** We implemented this ranking framework in a class-based Python toolkit tailored to NTv3. The toolkit provides modular interfaces for computing and visualizing the mechanistic representations described in this note—from attribution and attention matrices to differential fold-change profiles and predicted signal tracks. We distribute this toolkit via a Colab notebook to enable users to inspect, rank, and visualize the impact of variants at their loci of interest using NTv3’s learned regulatory representations.

### B.7.3. Biological Interpretation of Representative Variant Case Studies

By contrasting reference–alternative attribution maps, NTv3 resolves pathogenic SNVs into distinct mechanistic classes. Here we provide detailed biological interpretation for representative variants in each class.

**Motif ablation.** In the motif-ablation class, NTv3 signals that a pathogenic variant destroys TFBS functionality by collapsing a strong attribution-based motif in the reference allele to near zero in the alternative. For example, NTv3 reveals a pathogenic SNV (Fig. 6G) that ablates the binding site for TEF—a transcriptional driver in the human hematopoietic compartment [109]—within TMEM252, a recently identified pan-cancer tumor suppressor gene [110].

**Motif creation.** In the complementary motif-creation class, NTv3 signals that a pathogenic variant introduces new regulatory grammar by generating an attribution-based motif that is absent in the reference. For example, upon introduction of SNV rs12884809 in the KLC1 locus (Fig. 6H), NTv3 reconstructs a *de novo* ELK1 motif. Aligning with this reconstruction, independent GWAS/TWAS analyses report enhanced ELK1 binding at the KLC1 promoter and elevated KLC1 expression for the same pathogenic allele [111].

**Motif synergy.** In a third, less commonly recognized class, NTv3 signals that a pathogenic variant perturbs regulatory logic in a distributed rather than local manner. Whereas standard attribution frameworks emphasize base-resolution, motif-centric effects, they do not typically account for variant-induced architectural remodeling that unfolds across tens to thousands of base pairs. In contrast, NTv3 reveals coordinated, multi-motif rewiring: for example, NTv3 attributes the noncoding transcript exon variant [54] rs12216891 to the enhancement of a binding site for the mucosal regulator SPDEF, triggering a cascade of motif gains and losses within the ABO locus including a repressive NFIC site 150 bp away (Fig. 6I). Interpretation of NTv3’s gradients provides a putative mechanistic basis for the variant’s validated link to severe COVID-19 [112], suggesting it upregulates ABO glycosyltransferase activity, thereby increasing the density of surface antigens in the respiratory lining that facilitate viral entry.

**Multi-scale variant reconstruction.** The mechanistic classes described above are not mutually exclusive; distributed regulatory rewiring often encompasses local motif-level changes as subcategories of larger architectural effects. The APOL4 splice variant rs9610445 (Fig. 6J) exemplifies this multi-scale behavior. At the central variant position, NTv3 identifies splice-donor motif ablation—characteristic of the motif-ablation class—while simultaneously revealing distributed, long-range architectural reweighting in attention maps extending nearly

Pre-training	Sequence Context	Activity (KL ↓)	K-mer spectrum (JS ↓)	Motif occurrence (PR ↑)
True	True	0.72	0.045	0.967
True	False	0.81	0.045	0.967
False	True	1.08	0.044	0.966
False	False	0.97	0.045	0.967

**Table 15** | Ablation results on NTv3 generative results. Arrows indicate the preferred direction for each metric (lower ↓ or higher ↑).

32 kb from the variant site. Notably, AlphaGenome [8] independently recovers the same local splice-donor disruption through *in silico* mutagenesis in colon tissue, providing orthogonal support for NTv3’s mechanistic reconstruction, although that analysis was limited to motif-level changes and did not examine additional regulatory modalities. This example illustrates how NTv3’s multimodal outputs can disentangle hierarchical regulatory effects that span from nucleotide-level motif disruption to kilobase-scale interaction remodeling.

### B.8. Fine-tuning NTv3 for sequence generation

During the development of NTv3 for sequence generation, we investigated two key design questions. First, we asked whether the model truly leverages sequence interaction information provided by the promoter reporter context, rather than treating the two promoter constructs as simple categorical prompts during in context training. Second, we aimed to assess the extent to which the generative model benefits from the pretrained NTv3 650M (pre) initialization. To examine these factors, we constructed a simplified experimental setup using the same model architecture under the MDLM training scheme. We trained a model on STARR seq peak sequences without any conditioning signals. These peaks represent enhancer regions with measurable activity under either DSCP or RpS12. To ablate promoter specific context, we replaced all DSCP promoter constructs with a poly T sequence and all RpS12 constructs with a poly A sequence. Input length remained 4096 bp and Enhancers were still inserted in their usual positions. The model was trained using the same hyperparameters as the full NTv3 setup. And to test the effect of pre-training, we trained identical model architectures from random initialization under the same training procedure and data setup. All models were trained to convergence.

Because these ablation models cannot be evaluated *in cellulo*, we performed more stringent *in silico* assessments. For each model, generated sequences were compared with held out test sequences using both oracle predicted activity and sequence level similarity measures. To assess functional agreement, we computed the KL divergence between the oracle predicted activity distributions of generated and held out sequences. To assess compositional similarity, we performed K mer spectrum analysis for 1 through 4 mers and computed the Jensen Shannon divergence between generated and held out profiles. To quantify regulatory signature similarity, we scanned all sequences using motifs from the JASPAR database and computed the Pearson correlation between motif enrichment profiles. These metrics follow the assumption that sequences serving comparable regulatory roles should share similar compositional and motif level structure.

The resulting metrics are summarized in Table 15. From these results, we observe that pretraining consistently improves generation quality across conditions. Interestingly, including sequence context improves performance only when the model is initialized from NTv3 650M (pre), but slightly degrades performance when training from scratch. This suggests that without prior knowledge of genomic sequence structure, additional context makes the task more difficult, whereas pretrained models are able to integrate contextual information in a beneficial way. This supports the conclusion that NTv3 does in fact leverage sequence interaction information when such information is available.

A second observation is that sequence profile metrics such as K mer spectrum and motif occurrence show limited sensitivity in this setting. Despite clear functional differences reflected in the oracle predicted activity, these structural metrics exhibit minimal variation across conditions. As a result, in our main experiments we place greater emphasis on oracle based activity metrics when assessing generation quality. These metrics were used to select key generation hyperparameters, including the guidance scale  $\gamma$  and the number of denoising steps. The  $\gamma$  value used in the final model was chosen to minimize the KL divergence between generated and held out activity distributions, and the number of generation steps was set at the threshold beyond which additional steps did not further reduce KL divergence. With these hyperparameters established through *in silico* evaluation, our final assessment of generated sequences relies on the gold standard of direct *in cellulo*

validation.

## C. Supplementary Table Legends

**Supplementary Table C.1** | Details of functional tracks used in post-training per species.

**Supplementary Table C.2** | Details of genome annotation elements used in post-training per species.

**Supplementary Table C.3** | Details of NTV3 BENCHMARK tasks.

**Supplementary Table C.4** | Details of gene-level downstream tasks in Agronomics.