The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics

Hugo Dalla-Torre[1], Liam Gonzalez[1], Javier Mendoza-Revilla[1], Nicolas Lopez Carranza[1], Adam Henryk Grzywaczewski[2], Francesco Oteri[1], Christian Dallago[2 3], Evan Trop[1], Bernardo P. de Almeida[1], Hassan Sirelkhatim[2], Guillaume Richard[1], Marcin Skwark[1], Karim Beguir[1], Marie Lopez[*† 1], Thomas Pierrot[*† 1]

[1]**InstaDeep** [2]**Nvidia** [3]**TUM**

Abstract

Closing the gap between measurable genetic information and observable traits is a longstanding challenge in genomics. Yet, the prediction of molecular phenotypes from DNA sequences alone remains limited and inaccurate, often driven by the scarcity of annotated data and the inability to transfer learnings between prediction tasks. Here, we present an extensive study of foundation models pre-trained on DNA sequences, named the Nucleotide Transformer, ranging from 50M up to 2.5B parameters and integrating information from 3,202 diverse human genomes, as well as 850 genomes selected across diverse phyla, including both model and non-model organisms. These transformer models yield transferable, context-specific representations of nucleotide sequences, which allow for accurate molecular phenotype prediction even in low-data settings. We show that the developed models can be fine-tuned at low cost and despite low available data regime to solve a variety of genomics applications. Despite no supervision, the transformer models learned to focus attention on key genomic elements, including those that regulate gene expression, such as enhancers. Lastly, we demonstrate that utilizing model representations can improve the prioritization of functional genetic variants. The training and application of foundational models in genomics explored in this study provide a widely applicable stepping stone to bridge the gap of accurate molecular phenotype prediction from DNA sequence. Code and weights available at: https://github.com/instadeepai/nucleotide-transformer in Jax and https://huggingface.co/InstaDeepAI in Pytorch. Example notebooks to apply these models to any downstream task are available on HuggingFace.

Introduction

Foundation models in artificial intelligence (AI) are characterized by their large-scale nature, incorporating millions of parameters trained on extensive datasets. These models can be adapted for a wide range of subsequent predictive tasks and have profoundly transformed the AI field. Notable examples in natural language processing (NLP) include the so-called language models (LMs) BERT [1] and GPT [2]. LMs have gained significant popularity in recent years owing to their ability to be trained on unlabeled data, creating general-purpose representations capable of solving downstream tasks. One way they achieve a comprehensive understanding of language is by solving billions of cloze tests, in which they predict the correct word to fill in the blank in a given sentence. This approach is known as masked language modeling [1]. Early instances of foundation models applying this objective to biology involved training LMs on protein sequences, where they were tasked with predicting masked amino acids in large protein sequence datasets [3, 4, 5]. These protein LMs, when applied to downstream tasks using transfer learning, demonstrated the ability to compete with and even outperform previous methods for tasks such as predicting protein structure [3, 4] and function [6, 7], even in data scarce regiments [8].

[*]**Equal Supervision**
[†]**Corresponding authors**: t.pierrot@instadeep.com & m.lopez@instadeep.com

Beyond protein sequences, the dependency patterns encoded in DNA sequences play a fundamental role in understanding genomic processes, from characterizing regulatory regions to assessing the impact of individual variants within their haplotypic context. In this context, specialized deep learning (DL) models have been trained to uncover meaningful patterns of DNA [9, 10, 11, 12]. For example, DL models have been used to predict gene expression from DNA sequences [13, 14, 15, 16, 17, 18], with recent advancements combining convolutional neural networks (CNN) and transformer architectures enabling the encoding of regulatory elements located up to 100 kilobases (kb) upstream [19]. The abundance of data generated by modern genomics research presents both an opportunity and a challenge. On one hand, intricate patterns of natural variability across species and populations are readily available; on the other hand, powerful deep learning methods capable of handling large-scale data are necessary for accurate signal extraction from unlabeled datasets. Large foundation models trained on nucleotides sequences appear to be a natural choice for addressing this challenge [20, 21, 22, 23, 24, 25].

With the Nucleotide Transformer we present a systematic study and benchmark on how to construct and evaluate robust foundation models to encode genomic sequences. We iniatiated our study by building four distinct language models of varying sizes, ranging from 500M up to 2.5B parameters. These models were pre-trained them on three different datasets including the human reference genome, a collection of 3,202 diverse human genomes, and 850 genomes from various species. After training, we leveraged the representations (i.e. embeddings) from each of these models to simultaneously train them on a diverse set of 18 genomic curated prediction tasks. To decipher the sequence features learned during pre-training, we explored the models' attention maps, perplexities, and conducted data dimensionality reduction on their embeddings. Additionally, we evaluated the embeddings' capacity to model the impact of functionally important genetic variants in humans, through zero-shot-based scores. Expanding upon the findings from the initial set of experiments, we developed a second set of four language models, with decreasing sizes from 500M to 50M parameters, to investigate the scaling laws of such models. We successfully constructed a model that matches the performance of the previous best model while achieving this level of performance with only one-tenth the number of parameters and doubling the perception field size. We utilized our introduced benchmark and standardized robust evaluation methodology to systematically compare our eight models to other foundational models, consistently demonstrating that our best model outperforms them (Fig. 1a).

Results

The Nucleotide Transformer model accurately predicts diverse genomics tasks

We developed a collection of transformer-based DNA language models, dubbed Nucleotide Transformer (NT), which have learned general nucleotide sequence representations from 6kb unannotated genomic data (Fig. 1a; Methods). Inspired by trends in NLP, where larger training datasets and model sizes have demonstrated improved performance [26], we constructed transformer models with varying parameter sizes and datasets: (i) a 500 million parameter model trained on sequences extracted from the human reference genome (Human ref 500M), (ii) a 500 million and (iii) a 2.5 billion parameter model both trained on 3,202 genetically diverse human genomes[27] (1000G 500M and 1000G 2.5B respectively), and (iv) a 2.5 billion parameter model, encompassing 850 species from diverse phyla (Multispecies 2.5B), including 11 model organisms (Fig. 1c; Supplementary Tables 1, 2, 3, 4).

To assess the performance of these models in diverse molecular phenotype predictions, we assembled 18 distinct genomic datasets, each of which had established baseline performance metrics [28, 29, 30, 31, 32]. These datasets were then processed into a standardized format to facilitate experimentation and ensure reproducibility (Methods). This collection of 18 genomic datasets represents a varied and robust panel for evaluating the models' performance (Fig. 1d; Supplementary Table 5). Based on these genomic tasks, we evaluated the transformer models after self-supervised training through two different techniques: probing and fine-tuning (Fig. 1b). Probing refers to the use of learned LM embeddings of DNA sequences as input features to simpler models for predicting genomic labels. Specifically, we probed ten arbitrarily chosen layers of the LMs using either a logistic regression or a small multi-layer perceptron (MLP) composed of up to two hidden layers. In the case of fine-tuning, the LM head is

substituted with either a classification or regression head, and a parameter-efficient technique is used for retraining (Methods). To ensure a fair and accurate comparison of the transformer models with the available baselines, we implemented a ten-fold cross-validation strategy. We considered the models to be equivalent to or better than the baseline if the resulting two standard deviations either overlapped or were superior to the reported baseline value, respectively.

Using this criterion, we observed performances that either matched or exceeded those of 12 out of the 18 baseline models through probing alone (Supplementary Fig.1 and Table 6), and significantly outperformed probing from raw tokens. In agreement with recent work [33], we observed that the best performance is both model- and layer-dependent (Supplementary Table 7). We also noted that the highest model performance is never achieved by using embeddings from the final layer, as shown in earlier work [5]. For instance, in the H3K4me1 histone mark occupancy classification task, we observed a relative difference as high as 38% between the highest and lowest performing layer, indicating significant variation in learned representations across the layers (Supplementary Fig. 3). In comparison to our probing strategy, our fine-tuned models matched or surpassed 15 of the 18 baselines models. Notably, the larger and more diverse models consistently outperformed their smaller counterparts (Fig. 2a and Supplementary Table 8 and 6). Our results also suggest that training on a diverse dataset, represented by the Multispecies 2.5B model, outperforms or matches the 1000G 2.5B model on several tasks derived from human-based assays (Fig. 2a,b). This implies that a strategy of increased diversity, rather than just increased model size, may lead to improved prediction performance, particularly when computational resources are limited.

Fine-tuning has not been extensively explored in previous work [5], possibly due to its demanding computational requirements. We overcame the limitation by adopting a recent parameter-efficient fine-tuning technique [34] which requires only 0.1% of the total model parameters (Fig. 1b; Methods). This approach allowed for faster fine-tuning on a single GPU, reduced storage needs by 1,000-fold over all fine-tuning parameters, while still delivering comparable performance. In practice, we observed that rigorous probing was slower and more computationally intensive than fine-tuning, despite the apparent simplicity of using straightforward downstream models on embeddings. This discrepancy arises from the significant impact of factors such as layer choice, downstream model selection, and hyperparameters on performance. Additionally, fine-tuning exhibited a smaller variance in performance, enhancing the robustness of the approach. Overall, this general approach is versatile and adaptable to various tasks without requiring adjustments to the model architecture or hyperparameters. This stands in contrast to supervised models, which typically feature distinct architectures and necessitate training from scratch for each task.

We applied the Multispecies 2.5B model to three additional genomic prediction tasks, which encompassed the classification of 919 chromatin profiles from a diverse set of human cells and tissues [10], predicting canonical splice acceptor and donor sites across the whole genome [35], and predicting developmental and housekeeping enhancer activities using *Drosophila melanogaster* S2 cells [12] (Methods). Remarkably, the Multispecies 2.5B model achieved performance levels closely aligned with those of specialized deep learning models. For instance, in the case of classifying chromatin feature profiles, we obtained area under the curve (AUC) values that were, on average, only approximately $\sim$1% lower than those achieved by DeepSEA (Fig. 2c). Regarding the prediction of whether each position in a pre-mRNA transcript is a splice donor, splice acceptor, or neither, we adapted Nucleotide Transformer model to deliver nucleotide-level splice site predictions and achieved a top-k accuracy of 95% and a precision-recall AUC of 0.98 (Fig. 2d). Notably, our 2.5B 6kb-context model matched the performance of the state-of-the-art SpliceAI-10k [35], which was trained on 15kb input sequences, in addition to other splicing baselines; and outperformed SpliceAI when tested on 6kb input sequences. Finally, in the case of housekeeping and developmental enhancer prediction, our model slightly surpassed (1%) and obtained lower (4%) correlation values respectively (Fig. 2e), when compared to those of DeepSTARR [12]. Across these three different tasks, we also conducted a comparison between our parameter-efficient fine-tuning and full model fine-tuning. Interestingly, we observed no significant improvement in chromatin and splicing predictions, and only a modest 3% enhancement in enhancer activity predictions (Supplementary Fig. 2), supporting the use of our efficient fine-tuning approach.

Benchmark of genomics foundational models

We compared the Nucleotide Transformer models to other genomics foundational models: DNABERT-1 [20], DNABERT-2 [23], HyenaDNA (1kb and 32kb context length; [25]) and the Enformer (used as a pre-trained model, [19]) (see Fig. 2a,b and Methods). To ensure fair comparisons, all models underwent fine-tuning and evaluation using the same protocol across the 18 downstream tasks (Methods). When compared with DNABERT, HyenaDNA-32kb and Enformer, our Multispecies 2.5B model achieved the highest performance across all tasks (Fig. 2a,b, Table 8). DNABERT-1 and Enformer achieved top performance only in predicting promoters with a TATA-box motif, while DNABERT-2 matched the top performance in 9 out of the 18 tasks. The results with HyenaDNA-1kb were more comparable: our 2.5B model exhibited lower performance in 5 tasks, matched performance in 5 tasks, and demonstrated improved performance in 8 out of the 18 tasks. Notably, despite HyenaDNA being pre-trained on the human reference genome, our Multispecies 2.5B model outperformed it in 6 out of the 8 tasks related to human datasets, highlighting the advantage of pre-training on a diverse set of genome sequences. Furthermore, the longer-input HyenaDNA-32kb model exhibited worse performance in all 18 tasks, indicating a trade-off between increasing the input context length and performance in high-resolution tasks. We have established an interactive leaderboard containing results for all models across each task to facilitate comparisons [1]. To the best of our knowledge, this represents the most extensive benchmark of foundational genomics models to date and should serve as a reference for the development of further language models in genomics (Fig. 1c).

[1] https://huggingface.co/spaces/InstaDeepAI/nucleotide_transformer_benchmark

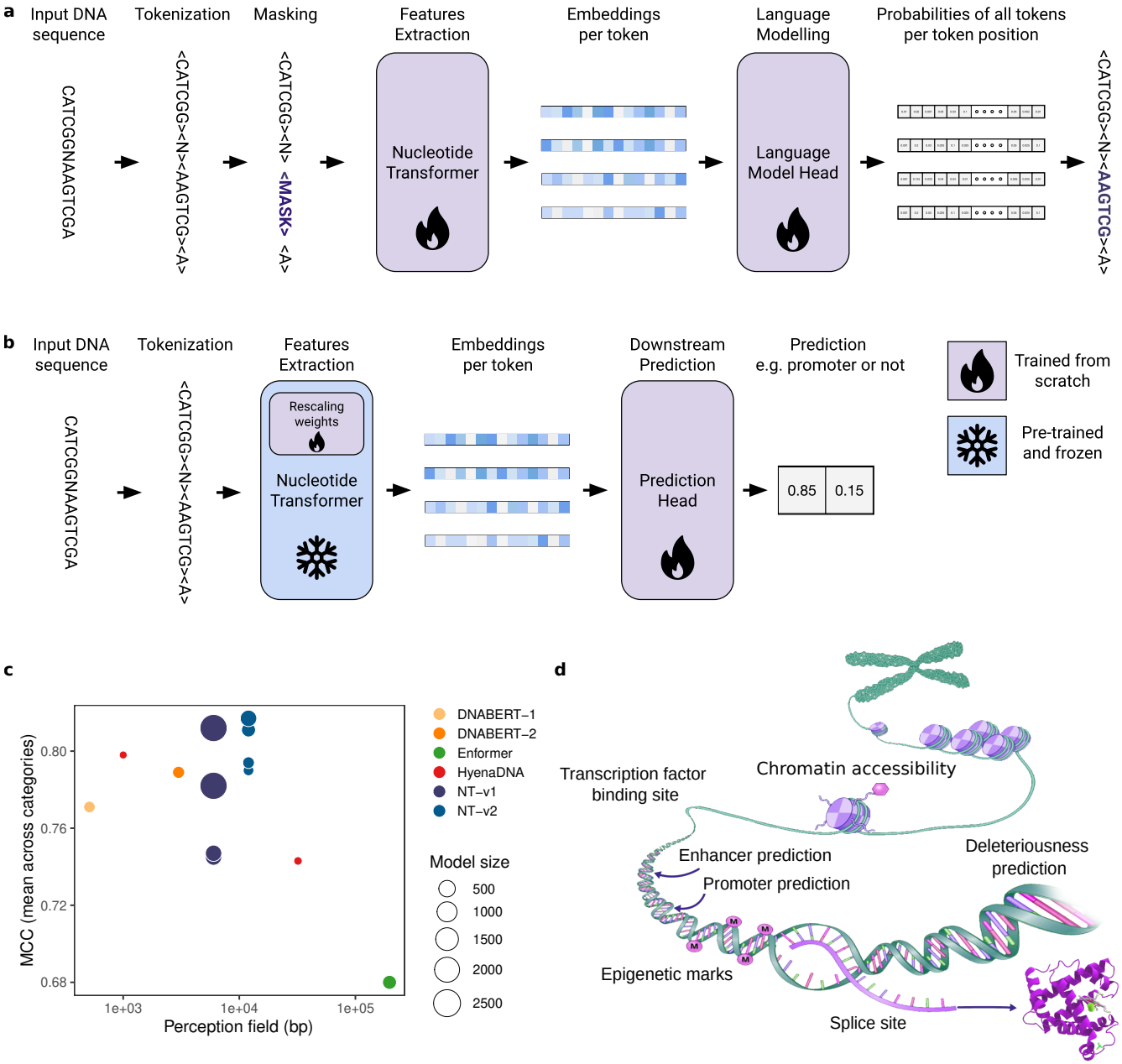


Figure 1: **The Nucleotide Transformer: effective methodology to pre-train, fine-tune, analyse and compare foundational models for genomics**. **a,b)** Overview of the Nucleotide Transformer training **(a)** and application for downstream genomic prediction tasks through fine-tuning **(b)**. Downstream task prediction through probing is similar but without the rescaling weights in the Nucleotide Transformer. **c)** Comparison of the Nucleotide Transformer models to other foundational genomics models in terms of perception field size, number of parameters and performance over our benchmark made of 18 curated downstream tasks. **d)** Graphical representation of genomic features considered for downstream tasks.

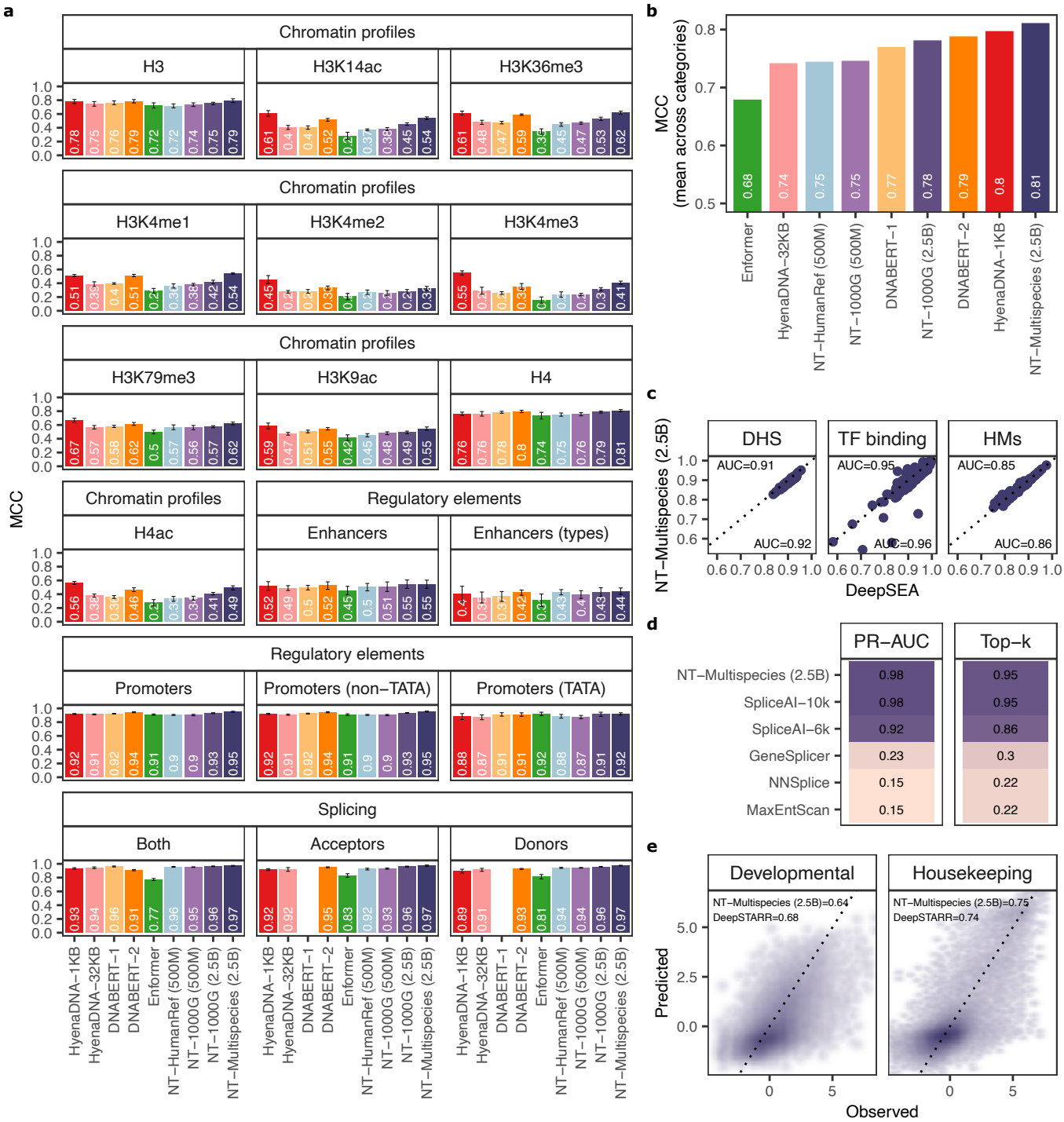


Figure 2: **The Nucleotide Transformer model accurately predicts diverse genomics tasks after fine-tuning. a)** Performance results across downstream tasks for fine-tuned NT models as well as HyenaDNA, DNABERT and Enformer pre-trained models (MCC: Matthew's correlation coefficient). DNABERT-1 has no values for splicing acceptor and donor tasks since their sequences are longer than what the model can handle. Error bars represent 2 SDs derived from 10-fold cross-validation. **b)** Normalized mean of MCC performance across downstream tasks (divided by category) for all language models after fine-tuning. **c)** The Multispecies 2.5B model performance on DNase I hypersensitive sites (DHS), histone marks (HMs), and transcription factor sites predictions from different human cells and tissues compared with the baseline DeepSEA model. Each dot represents the area under the ROC curve (AUC) for a different genomic profile. The average AUC per model is labeled. **d)** The Multispecies 2.5B model performance on predicting splice sites from the human genome, compared with the SpliceAI and other splicing models. **e)** The Multispecies 2.5B model performance on developmental and housekeeping enhancer activity predictions from *Drosophila melanogaster* S2 cells, compared with the baseline DeepSTARR model.

The Nucleotide Transformer model learned to reconstruct human genetic variants

To investigate the advantages of increasing the number of parameters in the models and enhancing the genomic diversity of the training datasets, we evaluated the models' ability to reconstruct masked nucleotides. Specifically, we divided the human reference genome into non-overlapping 6kb sequences, tokenized each sequence into 6-mers, randomly masked a certain number of tokens, and then calculated the proportion of tokens that were accurately reconstructed (Fig. 3a; Supplementary Fig. 4, Methods). We observed that the reconstruction accuracy of the Human reference 500M model exhibited higher median accuracy compared to the 1000G 500M model (median=0.202 versus 0.198, P<2.2e-16, two-sided Wilcoxon rank sum test). However, the accuracy of this model was lower than that achieved by the 1000G 2.5B model (median=0.216; P<2.2e-16, two-sided Wilcoxon rank sum test) and the 2.5B Multispecies model (median=0.219; P<2.2e-16, two-sided Wilcoxon rank sum test) (Supplementary Fig. 4), highlighting the impact of simultaneously increasing the model size and diversifying the dataset. Interestingly, while the Multispecies 2.5B model exhibited the highest overall reconstruction accuracy, specific sequences demonstrated significantly higher reconstruction accuracy for the Human 1000G 2.5B model (Fig. 3b). This likely indicates that certain characteristics of human sequences were better captured and learned by the latter model.

To gain a deeper understanding of how transformer language models represent variant sites in DNA and their capabilities in imputing such variants, we then focused on polymorphisms identified in human samples from diverse populations. In order to assess whether our models can generalize and reconstruct variants in unseen human genome sequences, we evaluated their sequence reconstruction effectiveness by recording model perplexity scores across single nucleotide polymorphisms (SNPs) occurring at frequencies ranging from 1% to 100% (Methods). To account for the influence of genomic background and genetic structure on the mutation reconstruction, we considered an independent dataset of genetically diverse human genomes, originating from 7 different meta-populations [36] (see Methods). Across varying SNP frequencies, we consistently observed that the 2.5B parameter models, with median perplexity across populations ranging from 95.9±17.9 (2SD) to 145±9.3, outperformed their 500M counterparts, for which the perplexity values fluctuate between 138.5±8.4 and 185.4±9.5 (Fig. 3c). Interestingly, the 1000G 2.5B model exhibited lower perplexity scores than the Multispecies 2.5B (median=121.5 versus 134.0, P<1.8e-12, two-sided Wilcoxon rank sum test) indicating that it effectively leveraged human genetic variability observed in the 1000G data to accurately reconstruct variants in unseen human genomes. These results are in line with the observed reconstruction accuracies over human genome sequences, and confirm the impact of model size on performance (Fig. 3b). Notably, and in contrast to typical genotype imputation methods where accuracy declines as variant frequency decreases [37], transformer models exhibited comparable performance across frequencies, suggesting their potential to enhance genotype imputation even for rare variants.

The attention layers detect known genomic elements in an unsupervised manner

To gain insights into the interpretability and understand the type of sequence elements that the nucleotide transformer utilizes when making predictions, we explored different aspects of the transformer's model architecture. First, we assessed the extent to which the embeddings can capture sequence information associated with five different categories of genomic elements (Supplementary Table. 9 6). We observed that the transformer models, without any supervision, learned to distinguish genomic sequences that were uniquely annotated as intergenic, intronic, coding, and UTR regions, albeit with varying degrees of proficiency across different layers (Fig. 3d; Supplementary Fig. 6; Methods). In particular, the 500 million-sized models and those trained on less diverse sequences, exhibited lower separation among genomic regions, reinforcing the enhanced ability of the largest models to capture relevant genomic patterns during self-supervised training. In the case of the Multispecies 2.5B model, the strongest separation at layer 1 was observed between intergenic and non-intergenic regions, followed by 5' UTR regions on layer 5, and a separation between most regions on layer 21 (Fig. 3d). The limited separation of 3' UTR regions from other elements suggests that the model has not fully learned to distinguish this type of element, or as previously suggested, that many of these regions

might be misannotated [38]. Consistent with these observations, our probing strategy demonstrated high classification performance for these elements, with accuracy values exceeding 0.78, especially for deeper layers (Fig. 3e). This demonstrated that the Nucleotide Transformer models have learned to detect known genomic elements within their embeddings in an unsupervised manner, which can be harnessed for efficient downstream genomics task predictions.

Next, we conducted an analysis of the transformer models through the lens of attention to comprehend which sequence regions are captured and utilized by the attention layers [39]. We computed the attention percentages across each model head and layer for sequences containing nine different types of genomic elements related to gene structure and regulatory features (Fig. 3f). In a formal sense, an attention head is considered to recognize specific elements when its attention percentage significantly exceeds the naturally occurring frequency of that element in the pre-training dataset (Methods). For example, a percentage of 50% implies that, on average over the human genome, 50% of the attention of that particular head is directed toward the type of element of interest. By applying this approach to each type of element across approximately 10,000 different 6kb windows, where the element can be situated at various positions and accounts for between 2-11% of the sequence (Table 9), we discovered that the model's attention is distinctly focused on various types of genomic elements across its diverse heads and layers (Fig. 3g, Supplementary Figs. 7 - 15). The number of significant attention heads across layers varied markedly across models, with the highest number of significant attention heads observed for the Multispecies 2.5B model for introns (117 out of 640 heads), exons (72) and TF binding sites (74) (Supplementary Figs. 7, 8 and Table 11), despite the relatively small proportion of sequences belonging to exons and TF motifs. Regarding enhancers, the maximum attention percentages were highest for the largest models, with the 1000G 2.5B model, for instance, achieving nearly 100% attention (Supplementary Fig. 14). Similar patterns were also observed for other genomic elements such as 3'UTR, promoter, and TF binding sites, where the 1000G 2.5B model showed highly specialized heads with high attention, particularly in the first layers (Supplementary Figs. 7 - 15).

To gain deeper insights into the pre-trained Nucleotide Transformer Multispecies 2.5B model at higher resolution (i.e. focusing on more local sequence features), we examined token probabilities across different types of genomics elements as a metric of sequence constraints and importance learned by the model. Specifically, we calculated the 6-mer token probabilities (based on masking each token at a time) for every 6kb window in chromosome 22. Our findings revealed that, in addition to repetitive elements, that are well reconstructed by the model as expected, the pre-trained model learned a variety of gene structure and regulatory elements. These included acceptor and donor splicing sites, polyA signals, CTCF binding sites and others (Supplementary Fig. 16a-d). Furthermore, we compared our token predictions with an experimental saturation mutagenesis splicing assay of exon 11 of the gene MST1R (data from Braun, Simon, et al. [40]). This analysis revealed a significant correlation between the experimental mutation effects and the token predictions made by our Multispecies 2.5B pre-trained model (Pearson Correlation Coefficient (PCC) = 0.44; Supplementary Fig. 16e). The model not only captured constraints at different splicing junctions but also identified a region in the middle of the second intron crucial for the splicing of this exon. This serves as robust validation of the biological knowledge acquired by the Nucleotide Transformer model during unsupervised pre-training.

Lastly, we took the Multispecies 2.5B model, which had been fully fine-tuned on the DeepSTARR enhancer activity data, and examined whether the model had learned about TF motifs and their relative importance specifically for enhancer activity. We used a dataset of experimental mutation of hundreds of individual instances of five different TF motif types across hundreds of enhancer sequences [12] and evaluated our model's accuracy in predicting these mutation effects. In comparison with the state-of-the-art enhancer activity DeepSTARR model, our model achieved similar performance for four TF motifs and demonstrated superior performance for the Dref motif (Supplementary Fig. 17). Collectively, these results illustrate how the transformer models have acquired the ability to recover gene structure and functional properties of genomic sequences and integrate them directly into its attention mechanism. This encoded information should be useful to assess the significance of genetic variants.

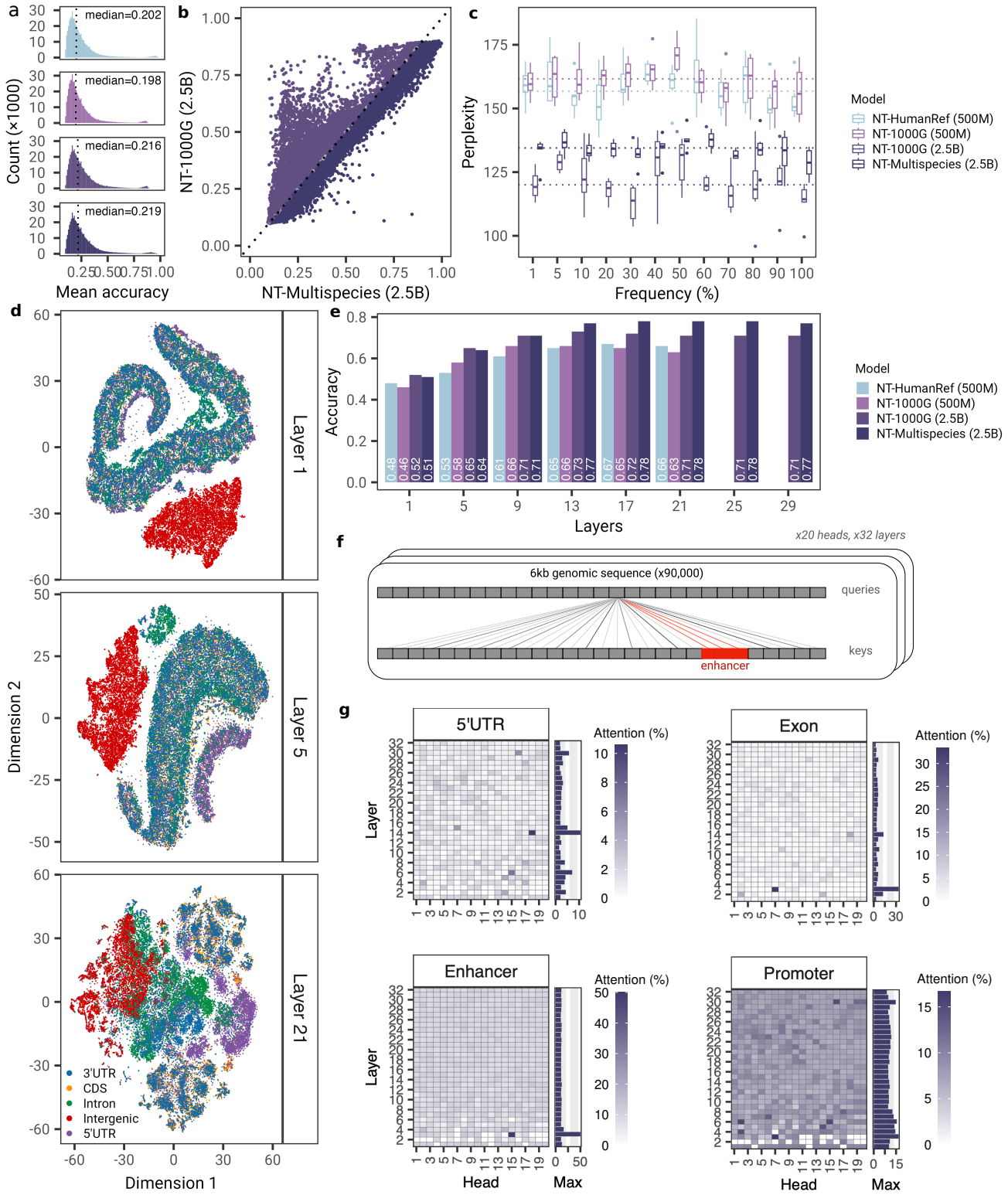


Figure 3: **The Nucleotide Transformer models acquired knowledge about genetic variations and genomic elements.** **a)** Mean reconstruction accuracy of 6kb sequences across transformer models. **b)** Comparison of reconstruction accuracies between the 1000G 2.5B and Multispecies 2.5B models. **c)** Reconstruction perplexity across allele frequencies. Perplexity values shown per frequency bin are based on 6 populations from the Human Genome Diversity Project. **d)** t-SNE projections of embeddings of 5 genomic elements from layer 1, 5, and 21 based on the Multispecies 2.5B model. **e)** Accuracy estimates based on probing to classify 5 genomic elements across layers. **f)** Cartoon describing the evaluation of attention levels at a given genomic element. **g)** Attention percentages per head and layer across transformer models computed on enhancers. Barplot on the right of each tile plot shows the maximum attention percentage across all heads for a given layer.

The Nucleotide Transformer embeddings predict the impact of mutations

Additionally, we evaluated the transformer models' capability to assess the severity of various genetic variants and prioritize those with functional significance. Initially, we investigated the potential of zero-shot scores, which are scores used to predict classes that were not seen by the model during training. We compared the distributions of these scores across 10 different types of genetic variants that varied in severity [41]. Precisely, we computed zero-shot scores using different aspects of vector distances in the embedding space, as well as those derived from the loss function (Fig. 4a and Methods). Encouragingly, several of these zero-shot scores exhibited a moderate correlation with severity across the models (Supplementary Fig. 18). This illustrates, how unsupervised training alone captured relevant information related to the potential severity of genetic mutations and highlighted the utility of testing different scoring methods. The high variability in correlations between scores also suggests that distinct aspects of the embedding space may more effectively capture information related to severity. Among these scores, cosine similarity exhibited the highest correlation with severity across models, with r^2 values ranging from -0.35 to -0.3 (P-value $< 6.55\text{e}^{-186}$) (Supplementary Fig. 18). Across transformer models, we observed that the lowest cosine similarity scores were assigned to genetic variants affecting protein function, such as stop-gained variants, as well as synonymous and missense variants (Fig. 4b). Conversely, we noted that higher scores were assigned to potentially less functionally important variants, such as intergenic variants.

To prioritize functional variants, especially those of high pathogenicity, we further explored the potential of zero-shot scores. Specifically, we assessed the ability of the models to classify genetic variants that influence gene expression regulation (i.e., expression quantitative trait loci [eQTLs]), genetic variants linked to DNA methylation variations (i.e., methylation quantitative trait loci [meQTLs]), genetic variants annotated as pathogenic in the ClinVar database, and genetic variants reported in the Human Gene Mutation Database (HGMD). Remarkably, the zero-shot scores demonstrated high classification performance, with the highest AUCs across the four tasks ranging from 0.7 to 0.8 (Fig. 4c). The highest performance obtained for the Clinvar variants (AUC=0.80 for the Multispecies 2.5B model), suggests that, at least for highly pathogenic variants, zero-shot scores might be readily applicable. To formally evaluate the effectiveness of transformer models, we also made predictions based on fine-tuned models, and compared their performance against several methods. These methods encompassed those measuring levels of genomic conservation, as well as scores obtained from models trained on functional features. Notably, the transformer models either slightly outperformed or closely matched the performance of the other models (Fig. 4d). The best-performing models for prioritizing molecular phenotypes (i.e. eQTLs and meQTLs) were those trained on human sequences, whereas the best-performing model for prioritizing pathogenic variants was based on multispecies sequences. Given that the most severely pathogenic variants tend to impact gene function due to amino acid changes, it is possible that the multispecies model leveraged sequence variation across species to learn about the degree of conservation across sites. Our results also suggest that higher predictive power for non-coding variants such as eQTLs and meQTLs, could be achieved by better-learned sequence variation derived from increased human genetic variability. Moreover, when compared to zero-shot scores, the dot product yielded AUC values of 0.73 and 0.71 for eQTLs and meQTLs, respectively, slightly surpassing or matching those obtained by the fine-tuned models. Given that most of these genetic variants tend to reside within regulatory regions [42, 43, 44], it is probable that the transformer models, without any supervision, have learned to distinguish relevant regulatory genomic features associated with gene expression and methylation variation. This is in accordance with the level of attention observed across layers and heads, especially for relevant regulatory sequences such as enhancers (Fig. 3d) and promoters (Supplementary Fig. 12), which have been shown to be enriched in meQTLs and eQTLs [42, 43, 44]. Overall, these results illustrate how DNA-based transformer models can help reveal and contribute to understanding the potential biological implications of variants linked to molecular phenotypes and diseases.

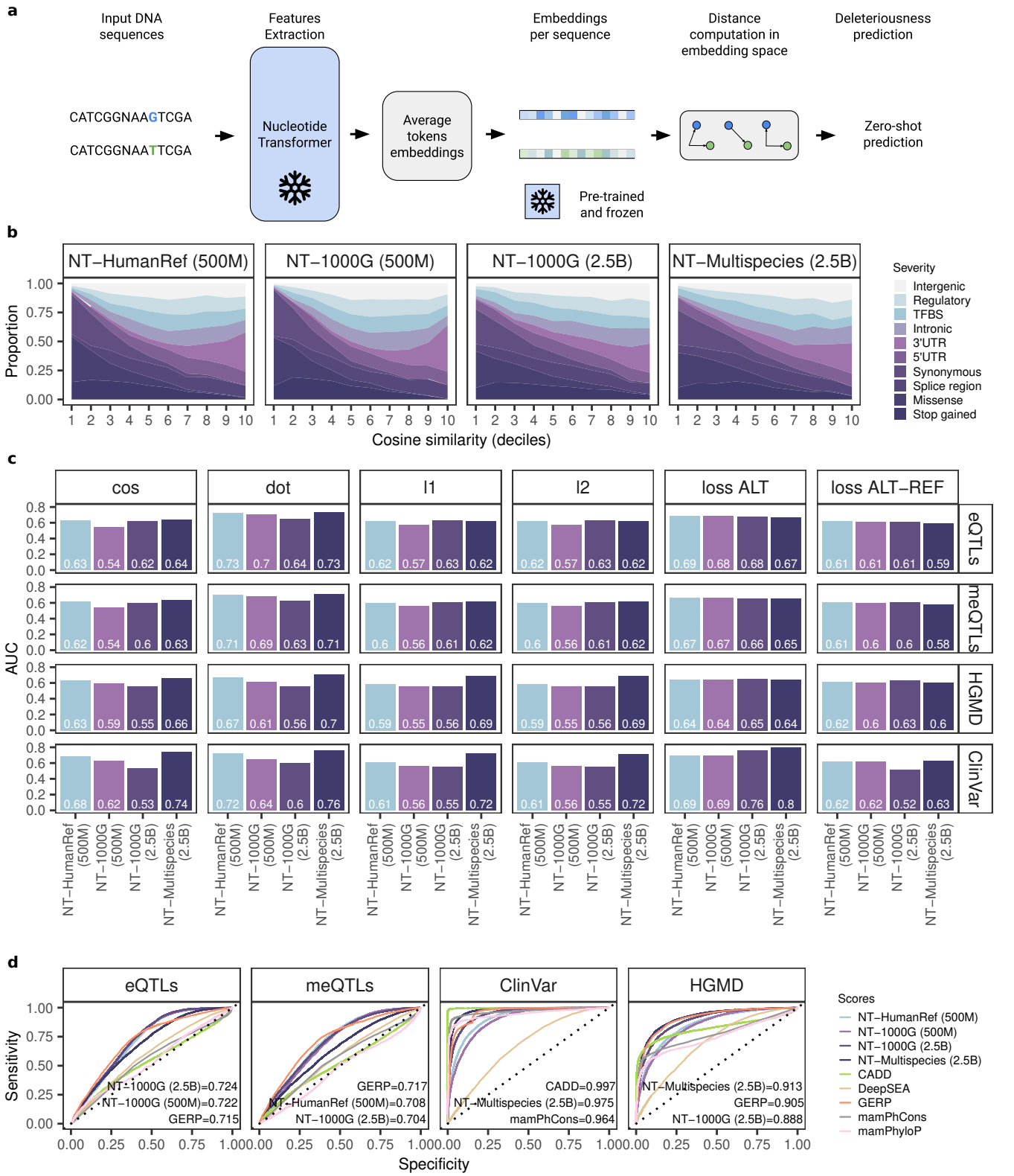


Figure 4: **Prioritizing functional genetic variants. a)** Overview of the Nucleotide Transformer application of zero-shot predictions. **b)** Proportion of variant consequence terms across deciles based on the cosine similarity metric across models. The consequence terms are shown in order of severity (less severe to more severe) as estimated by Ensembl. **c)** Comparison of zero-shot predictions for prioritizing functional variants based on different distance metrics. **d)** Comparison of fine-tuned models and available methods for prioritizing functional variants based on GRASP eQTLs and meQTLs, ClinVar, and HGMD annotated mutations. Model performance is measured with the area under the receiver operating characteristic curve (AUC). The AUC for the three best-performing models is shown.

Optimization of Nucleotide Transformer models towards cost-effective predictions in genomics

Finally, we explored the potential for optimizing our top-performing NT model by incorporating contemporary architectural advancements and extending the training duration. We developed four new Nucleotide Transformer models (NT-v2) with varying number of parameters ranging from 50 million to 500 million, and introduced a series of architectural enhancements (Supplementary Table 1; Methods). These include the incorporation of rotary embeddings, the implementation of swiGLU activations, and the elimination of MLP biases and dropout mechanisms, in line with the latest research [45]. Additionally, we expanded the context length to cover 12 kilobase pairs (kbp) to accommodate longer sequences and capture more distant genomic interactions. We extended the training duration of the 250 and 500 million parameter models to encompass 1 trillion tokens, aligning with recent recommendations in the literature [46] (Fig. 5a). After pre-training on the same multispecies dataset, all four NT-v2 models underwent fine-tuning and evaluation across the same set of 18 downstream tasks, with their results compared to those of the NT models (Fig. 5b, Supplementary Tables 6 and 8).

We observed that the 50 million-parameter NT-v2 model outperformed our two NT 500 million-parameter models as well as the 2.5 billion-parameter model trained on the 1000G dataset. This demonstrates that the synergy of a superior pre-training dataset, coupled with advancements in training techniques and architecture, can lead to a remarkable 50-fold reduction in model parameters while simultaneously enhancing performance (Fig. 1a). However, it is important to note that none of the NT-v2 models surpassed the 2.5 billion-parameter multi-species model in performance. The NT-v2 500 million-parameter model managed to achieve a similar performance level while maintaining a significantly leaner parameter count and doubling the perception field. Of particular interest is the NT-v2 250 million-parameter model, which attained an average MCC similar to that of the 10-times larger 2.5 billion-parameter model (Fig. 5b).

To gain further insights into the need for longer pre-training, we conducted a systematic evaluation of the NT-v2 models' performance in function of the number of tokens seen during pre-training (Fig. 5c). This showed that the 500 million-parameter model has a small improvement over the 250 million model only after training for longer than 500 billion tokens. In summary, the Nucleotide Transformer-v2 models, equipped with a context length of 12 kbp, are suitable for deployment on cost-effective accelerators due to their compact sizes. Consequently, they offer an economically viable and practical alternative for users seeking to leverage cutting-edge foundational models in their downstream applications.

We assessed the advantage of the longer context length of the NT-v2 models by evaluating the 500M model on the SpliceAI splicing task. We adapted our classification head to predict nucleotide-level probabilities of being a splicing donor, acceptor, or none (Fig. 5d; Methods). Compared to the NT 6kb models, our NT-v2 12kb models improved performance by 1% to a top-k accuracy of 96% and a precision-recall AUC of 0.98 (Fig. 5e). This performance surpasses that of the state-of-the-art SpliceAI-10k [35], which was trained on 15kb input sequences. It's worth noting that we did not attempt to optimize our model architectures specifically for the splicing prediction task; instead we applied a similar fine-tuning approach as used for other downstream tasks, with adjustments in the classification head to yield nucleotide-level predictions. Further architectural refinements tailored to specific tasks like splicing are likely to enhance performance. In summary, these results affirmed the utility and effectiveness of both Nucleotide Transformer v1 and v2 models for a wide range of genomics tasks, requiring minimal modifications while achieving high accuracy.

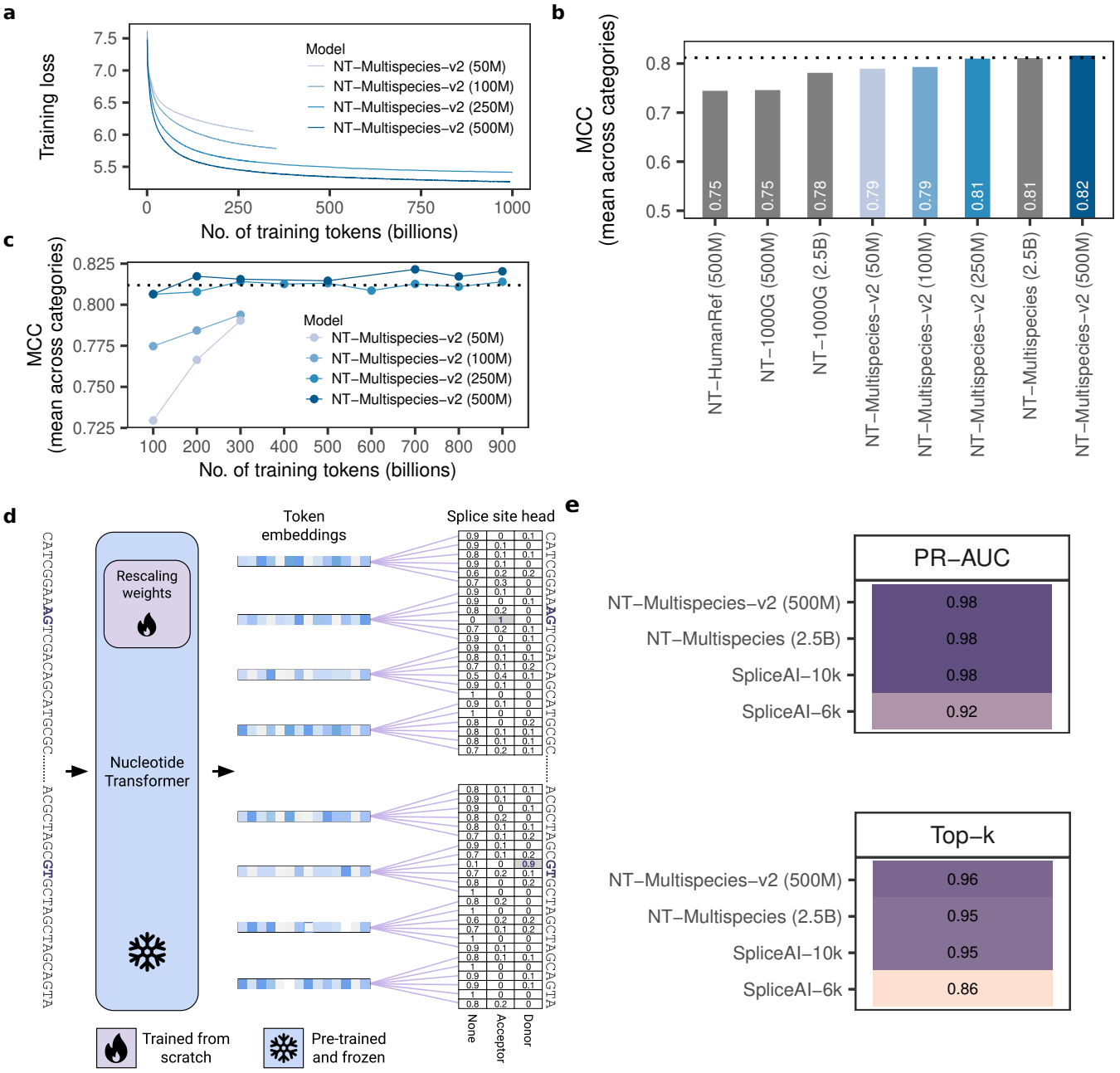


Figure 5: **Efficient model architecture allows to match performance while strongly reducing the number of model parameters.** **a)** Nucleotide Transformer-v2 models loss value evolution during training as a function of the number of tokens seen so far. **b)** Normalized mean of MCC performance across downstream tasks (divided by category) for all NT (black) and -v2 (blue) Nucleotide Transformer models after fine-tuning. Black dashed line represents the performance of the NT 2.5B multispecies model. **c)** Normalized mean of MCC performance for NT-v2 models as a function of the number of tokens seen during pre-training. **d)** Overview of the Nucleotide Transformer fine-tuning on nucleotide-level splice site prediction task. Pre-trained weights and weights trained from scratch are highlighted. **e)** The Multispecies v2 500M model performance on predicting splice sites from the human genome, compared with its 2.5B counterpart and SpliceAI.

Discussion

To our knowledge, this study represents the first attempt to investigate the impact of different datasets used to pre-train equally-sized transformer models on DNA sequences. Our results, based on distinct genomic prediction tasks, demonstrate that both intra-species (i.e., when training on multiple genomes of a single species) and inter-species (i.e., on genomes across different species) variability significantly influence accuracy across tasks (Fig. 1c, 2a). Models trained on genomes from different species outperform those trained exclusively on human sequences in many human prediction tasks. This suggests that transformer models trained on diverse species have learned to capture genomic features that likely have functional importance across species, thus enabling better generalization in various human-based prediction tasks. Based on this finding, we anticipate that future studies may benefit from leveraging genetic variability across species.

The transformer models trained in this study ranged from 50 million up to 2.5 billion parameters, which is five times larger than DNABERT-1 [20] and ten times larger than the Enformer [19] models. As previously demonstrated in NLP research [26], our results on genomic prediction tasks confirmed that increasing model size leads to improved performance. To train the models with the largest parameter sizes, we utilized a total of 128 GPUs across 16 compute nodes for 28 days. Significant investments were made in engineering efficient training routines that fully utilized the infrastructure, underscoring the importance of both specialized infrastructure and dedicated software solutions. Once trained, however, these models can be used for inference at a relatively low cost, and we provide notebooks to apply these models to any downstream task of interest, thereby facilitating further research.

Previous work, which was based on language models trained on biological data (primarily protein sequences), evaluated downstream performance exclusively by probing the last transformer layer [5]. This choice was likely driven by its perceived ease of use, relatively good performance, and low computational complexity. In this study, our aim was to evaluate downstream accuracy through computationally intensive and thorough probing of different transformer layers, downstream models, and hyperparameter sweeps. We observed that the best probing performance was achieved with intermediate transformer layers (Supplementary Fig. 1), which aligns with recent work in computational biology [33] and common practice in NLP [47]. Through probing alone, the Multispecies 2.5B model outperformed the baseline for 8 out of 18 tasks. Additionally, we explored a recent downstream fine-tuning technique that introduces a small number of trainable weights into the transformer. This approach provides a relatively fast and resource-efficient fine-tuning procedure with minor differences compared to full-model fine-tuning (IA^3 [34]). Notably, this fine-tuning approach only requires 0.1% of the total number of parameters, allowing even our largest models to be fine-tuned in under 15 minutes on a single GPU. In comparison to the extensive probing exercise, this technique yielded superior results while using fewer compute resources, confirming that downstream model engineering can lead to performance improvements [48]. The use of this technique makes fine-tuning competitive with probing from an operational perspective, both for training and inference.

The Nucleotide Transformer model represents a powerful and versatile approach that can be readily adapted to a wide array of genomics prediction tasks, encompassing areas such as histone modifications, splicing sites, and regulatory elements like enhancers and promoters. The value of our unsupervised pre-training approach becomes particularly evident when dealing with smaller datasets, where training supervised models from scratch typically yields limited results. Recognizing the pivotal role of foundational genomics models in the field, we have conducted an extensive comparison and benchmarking study, evaluating our model against four distinct pre-trained models: DNABERT-1 [20], DNABERT-2 [23], HyenaDNA [25] and Enformer [19]. These findings will serve as a reference point for the development of future language models in genomics.

Through various analyses related to the transformer architecture, we have demonstrated that the models have acquired the ability to recognize key regulatory genomic elements. This property is demonstrated throughout the analysis of attention maps, embedding spaces, token reconstruction, and probability distributions. Crucial regulatory elements governing gene expression, such as enhancers and promoters [42, 43, 44], were consistently detected by all models across multiple heads and layers. Additionally, we observed that each model contained at least one layer that produced embeddings

clearly distinguishing the five genomic elements analyzed. Given that self-supervised training facilitated the detection of these elements, we anticipate that this approach can be harnessed for the characterisation or discovery of novel genomic elements in future research.

We have demonstrated that transformer models can match and even outperform other methods for predicting variant effects and deleteriousness. In addition to developing supervised transformer models, we have also showcased the utility of zero-shot-based scores, especially for predicting non-coding variant effects. Given that these zero-shot-based scores can be derived solely from genomic sequences, we encourage their application in non-human organisms, especially those with limited functional annotation.

Lastly, we have demonstrated the potential of enhancing the model architecture to achieve a dual benefit of reducing model size and improving performance. Our subsequent series of NT-v2 models have a context length of 12 kbp. To put this in perspective, this length is 24x and 4x larger than the respective average context lengths of DNABERT-1 (512bp) and DNABERT-2 (3kb). These advanced models not only exhibit improved downstream performance but also offer the advantage of being suitable for execution and fine-tuning on economical hardware. This advantage arises from their compact size, complemented by the utilization of the efficient fine-tuning techniques that we have introduced.

While our models have an attention span that remains limited, the recently developed Enformer model [19] suggested that increasing the perception field up to 200kb is necessary to capture long-range dependencies in the human genome. The authors argued that these are needed to accurately predict gene expression, which is controlled by distal regulatory elements, usually located far greater than 10-20kb away from the transcription starting site. Processing such large inputs is intractable with the standard transformer architecture, due to the quadratic scaling of self-attention with respect to the sequence length. The Enformer model addresses this issue by passing sequences through convolution layers to reduce input dimensions before reaching the transformer layers. However, this choice hampers its effectiveness in language modeling. On the other hand, the recent HyenaDNA models were trained with perception fields up to 1M base pairs, but our benchmark analyses in downstream tasks show that the performance of these models quickly deteriorates when the perception field used during training increases. Based on our results, we suggest that developing transformer models with the ability to handle long inputs while maintaining high performance on shorter ones is a promising direction for the field. In an era where multi-omics data are rapidly expanding, we ultimately anticipate that the methodology presented here, along with the available benchmarks and code, will stimulate the adoption, development, and enhancement of large foundational language models in genomics.
.

Data availability

The Nucleotide Transformer pre-training sequences were obtained from publicly available resources. The 1000 Genomes Project sequences were obtained from http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000G_2504_high_coverage, and the human and multispecies reference genomes from https://ftp.ncbi.nlm.nih.gov/genomes/refseq/. Gene annotations were obtained from GENCODE (https://www.gencodegenes.org/) and Ensembl databases (https://www.ensembl.org). Variants effects predictions were obtained using the Variant Effect Prediction (VEP) API from Ensembl (https://www.ensembl.org/info/docs/tools/vep/index.html). Pathogenic and regulatory variants were extracted from ClinVar (https://ftp.ncbi.nlm.nih.gov/pub/clinvar/vcf_GRCh38/), the Genome-Wide Repository of Associations Between SNPs and Phenotypes (GRASP) (https://grasp.nhlbi.nih.gov), and The Human Gene Mutation Database (HGMD) (public version 2020.4) through Ensembl Biomart. The training and test datasets used in the downstream tasks were acquired from repositories that were referenced in the corresponding publications (see Supplementary Table 5). We have also created an interactive browser session with the pre-trained model token probabilities across the full chromosome 22 on the WashU Epigenome Browser at https://shorturl.at/jov28. HuggingFace versions of our pre-training and downstream tasks datasets can be found at https://huggingface.co/InstaDeepAI.

Code availability

Model code and weights of the pre-trained transformer models as well as inference code in Jax are available for research purposes at https://github.com/instadeepai/nucleotide-transformer. HuggingFace versions of the models, in PyTorch, can be found at https://huggingface.co/InstaDeepAI. Example notebooks are available on HuggingFace at https://huggingface.co/docs/transformers/notebooks#pytorch-bio.

Acknowledgements

We thank members of the Rostlab, particularly Tobias Olenyi, Ivan Koludarov, and Burkhard Rost for constructive discussions that helped identify interesting research directions. We also thank Maša Roller for helpful discussion and commenting on the manuscript. Furthermore, we would like to express our gratitude to all volunteers who generously provided their biological data, as well as to the researchers who deposited experimental data in public databases, the researchers who maintain these databases, and those who make analytical and predictive methods available to the scientific community.

References

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

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[4] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proceedings of the National Academy of Sciences*, vol. 118, no. 15, p. e2016239118, 2021.

[5] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, *et al.*, "Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing," *arXiv preprint arXiv:2007.06225*, 2020.

[6] M. Littmann, M. Heinzinger, C. Dallago, T. Olenyi, and B. Rost, "Embeddings from deep learning transfer go annotations beyond homology," *Scientific reports*, vol. 11, no. 1, pp. 1–14, 2021.

[7] C. Marquet, M. Heinzinger, T. Olenyi, C. Dallago, K. Erckert, M. Bernhofer, D. Nechaev, and B. Rost, "Embeddings from protein language models predict conservation and variant effects," *Human genetics*, vol. 141, no. 10, pp. 1629–1647, 2022.

[8] M. Littmann, M. Heinzinger, C. Dallago, K. Weissenow, and B. Rost, "Protein embeddings and deep learning predict binding residues for various ligand classes," *Scientific Reports*, vol. 11, Dec. 2021.

[9] Ž. Avsec, M. Weilert, A. Shrikumar, S. Krueger, A. Alexandari, K. Dalal, R. Fropf, C. McAnany, J. Gagneur, A. Kundaje, *et al.*, "Base-resolution models of transcription-factor binding reveal soft motif syntax," *Nature Genetics*, vol. 53, no. 3, pp. 354–366, 2021.

[10] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature methods*, vol. 12, no. 10, pp. 931–934, 2015.

[11] L. J. Mateo, N. Sinnott-Armstrong, and A. N. Boettiger, "Tracing dna paths and rna profiles in cultured cells and tissues with orca," *Nature protocols*, vol. 16, no. 3, pp. 1647–1713, 2021.

[12] B. P. de Almeida, F. Reiter, M. Pagani, and A. Stark, "Deepstarr predicts enhancer activity from dna sequence and enables the de novo design of synthetic enhancers," *Nature Genetics*, vol. 54, no. 5, pp. 613–624, 2022.

[13] G. Eraslan, Ž. Avsec, J. Gagneur, and F. J. Theis, "Deep learning: new computational modelling techniques for genomics," *Nature Reviews Genetics*, vol. 20, no. 7, pp. 389–403, 2019.

[14] J. Zhou, C. L. Theesfeld, K. Yao, K. M. Chen, A. K. Wong, and O. G. Troyanskaya, "Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk," *Nature Genetics*, vol. 50, pp. 1171–1179, July 2018.

[15] D. R. Kelley, "Cross-species regulatory sequence activity prediction," *PLOS Computational Biology*, vol. 16, p. e1008050, July 2020.

[16] D. R. Kelley, Y. A. Reshef, M. Bileschi, D. Belanger, C. Y. McLean, and J. Snoek, "Sequential regulatory activity prediction across chromosomes with convolutional neural networks," *Genome Research*, vol. 28, pp. 739–750, Mar. 2018.

[17] V. Agarwal and J. Shendure, "Predicting mRNA abundance directly from genomic sequence using deep convolutional neural networks," *Cell Reports*, vol. 31, p. 107663, May 2020.

[18] K. M. Chen, A. K. Wong, O. G. Troyanskaya, and J. Zhou, "A sequence-based global map of regulatory activity for deciphering human genetics," *Nature genetics*, vol. 54, no. 7, pp. 940–949, 2022.

[19] Ž. Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley, "Effective gene expression prediction from sequence by integrating long-range interactions," *Nature methods*, vol. 18, no. 10, pp. 1196–1203, 2021.

[20] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, "Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome," *Bioinformatics*, vol. 37, no. 15, pp. 2112–2120, 2021.

[21] M. T. Zvyagin, A. Brace, K. Hippe, Y. Deng, B. Zhang, C. O. Bohorquez, A. Clyde, B. Kale, D. Perez-Rivera, H. Ma, *et al.*, "Genslms: Genome-scale language models reveal sars-cov-2 evolutionary dynamics.," *bioRxiv*, 2022.

[22] C. Outeiral and C. M. Deane, "Codon language embeddings provide strong signals for protein engineering," *bioRxiv*, 2022.

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

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

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

[26] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[27] . G. P. Consortium *et al.*, "A global reference for human genetic variation," *Nature*, vol. 526, no. 7571, p. 68, 2015.

[28] T. H. Phaml, D. H. Tran, T. B. Ho, K. Satou, and G. Valiente, "Qualitatively predicting acetylation and methylation areas in dna sequences," *Genome Informatics*, vol. 16, no. 2, pp. 3–11, 2005.

[29] Q. Geng, R. Yang, and L. Zhang, "A deep learning framework for enhancer prediction using word embedding and sequence generation," *Biophysical Chemistry*, vol. 286, p. 106822, 2022.

[30] R. Wang, Z. Wang, J. Wang, and S. Li, "Splicefinder: ab initio prediction of splice sites using convolutional neural network," *BMC bioinformatics*, vol. 20, no. 23, pp. 1–13, 2019.

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

[32] N. Scalzitti, A. Kress, R. Orhand, T. Weber, L. Moulinier, A. Jeannin-Girardon, P. Collet, O. Poch, and J. D. Thompson, "Spliceator: Multi-species splice site prediction using convolutional neural networks," *BMC bioinformatics*, vol. 22, no. 1, pp. 1–26, 2021.

[33] F.-Z. Li, A. P. Amini, K. K. Yang, and A. X. Lu, "Pretrained protein language model transfer learning: is the final layer representation what we want?,"

[34] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *arXiv preprint arXiv:2205.05638*, 2022.

[35] K. Jaganathan, S. K. Panagiotopoulou, J. F. McRae, S. F. Darbandi, D. Knowles, Y. I. Li, J. A. Kosmicki, J. Arbelaez, W. Cui, G. B. Schwartz, *et al.*, "Predicting splicing from primary sequence with deep learning," *Cell*, vol. 176, no. 3, pp. 535–548, 2019.

[36] A. Bergström, S. A. McCarthy, R. Hui, M. A. Almarri, Q. Ayub, P. Danecek, Y. Chen, S. Felkel, P. Hallast, J. Kamm, H. Blanché, J.-F. Deleuze, H. Cann, S. Mallick, D. Reich, M. S. Sandhu, P. Skoglund, A. Scally, Y. Xue, R. Durbin, and C. Tyler-Smith, "Insights into human genetic variation and population history from 929 diverse genomes," *Science*, vol. 367, Mar. 2020.

[37] S. Rubinacci, O. Delaneau, and J. Marchini, "Genotype imputation using the positional burrows wheeler transform," *PLoS genetics*, vol. 16, no. 11, p. e1009049, 2020.

[38] G. Benegas, S. S. Batra, and Y. S. Song, "Dna language models are powerful zero-shot predictors of non-coding variant effects," *bioRxiv*, pp. 2022–08, 2022.

[39] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, and N. F. Rajani, "Bertology meets biology: interpreting attention in protein language models," *arXiv preprint arXiv:2006.15222*, 2020.

[40] S. Braun, M. Enculescu, S. T. Setty, M. Cortés-López, B. P. de Almeida, F. R. Sutandy, L. Schulz, A. Busch, M. Seiler, S. Ebersberger, *et al.*, "Decoding a cancer-relevant splicing decision in the ron proto-oncogene using high-throughput mutagenesis," *Nature communications*, vol. 9, no. 1, p. 3315, 2018.

[41] W. McLaren, L. Gil, S. E. Hunt, H. S. Riat, G. R. Ritchie, A. Thormann, P. Flicek, and F. Cunningham, "The ensembl variant effect predictor," *Genome biology*, vol. 17, no. 1, pp. 1–14, 2016.

[42] T. Lappalainen, M. Sammeth, M. R. Friedländer, P. A. 't Hoen, J. Monlong, M. A. Rivas, M. Gonzalez-Porta, N. Kurbatova, T. Griebel, P. G. Ferreira, *et al.*, "Transcriptome and genome sequencing uncovers functional variation in humans," *Nature*, vol. 501, no. 7468, pp. 506–511, 2013.

[43] G. Consortium, "The gtex consortium atlas of genetic regulatory effects across human tissues," *Science*, vol. 369, no. 6509, pp. 1318–1330, 2020.

[44] U. Võsa, A. Claringbould, H.-J. Westra, M. J. Bonder, P. Deelen, B. Zeng, H. Kirsten, A. Saha, R. Kreuzhuber, S. Yazar, *et al.*, "Large-scale cis-and trans-eqtl analyses identify thousands of genetic loci and polygenic scores that regulate blood gene expression," *Nature genetics*, vol. 53, no. 9, pp. 1300–1310, 2021.

[45] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[46] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[47] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020.

[48] H. Stärk, C. Dallago, M. Heinzinger, and B. Rost, "Light attention predicts protein location from the language of life," *Bioinformatics Advances*, vol. 1, no. 1, p. vbab035, 2021.

[49] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," 2019.

[50] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[51] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.

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

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[54] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[55] M. Byrska-Bishop, U. S. Evani, X. Zhao, A. O. Basile, H. J. Abel, A. A. Regier, A. Corvelo, W. E. Clarke, R. Musunuri, K. Nagulapalli, *et al.*, "High-coverage whole-genome sequencing of the expanded 1000 genomes project cohort including 602 trios," *Cell*, vol. 185, no. 18, pp. 3426–3440, 2022.

[56] P. Dmitry K, H. Christopher T, L. Stuart, C. Megan, M. H. Nancy, L. Tong Ihn, B. George W, W. Kimberly, R. P Alex, H. Elizabeth, Z. Julia, L. Fran, G. David K, and Y. Richard A, "Genome-wide map of nucleosome acetylation and methylation in yeast," *Cell*, vol. 122, pp. 517–27, 2005.

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

[58] R. Leslie, C. J. O'Donnell, and A. D. Johnson, "Grasp: analysis of genotype–phenotype results from 1390 genome-wide association studies and corresponding open access database," *Bioinformatics*, vol. 30, no. 12, pp. i185–i194, 2014.

[59] M. J. Landrum, J. M. Lee, M. Benson, G. R. Brown, C. Chao, S. Chitipiralla, B. Gu, J. Hart, D. Hoffman, W. Jang, *et al.*, "Clinvar: improving access to variant interpretations and supporting evidence," *Nucleic acids research*, vol. 46, no. D1, pp. D1062–D1067, 2018.

[60] P. D. Stenson, M. Mort, E. V. Ball, M. Chapman, K. Evans, L. Azevedo, M. Hayden, S. Heywood, D. S. Millar, A. D. Phillips, *et al.*, "The human gene mutation database (hgmd®): optimizing its use in a clinical diagnostic or research setting," *Human genetics*, vol. 139, pp. 1197–1207, 2020.

A Methods

A.1 Models

Language models (LMs) have been primarily developed within Natural Language Processing (NLP) to model spoken languages [1, 2]. A LM is a probability distribution over sequences of tokens (often words), i.e. given any sequence of words, an LM will return the probability for that sentence to exist. LMs gained in popularity thanks to their ability to leverage large unlabeled datasets to generate general-purpose representations that can solve downstream tasks even when little supervised data is available [49]. One technique to train LMs tasks models to predict the most likely tokens at masked positions in a sequence, often referred to as masked language modelling (MLM). Motivated by results obtained with MLM in the field of protein research [4, 5], where proteins are considered as sentences and amino-acids as words, we apply MLM to train language models transformers in genomics, considering sequences of nucleotides as sentences and k-mers (with k=6) as words. Transformers are a class of deep learning models that achieved breakthroughs in machine learning fields including NLP and computer vision. They consist of an initial embedding layer that transform positions in the input sequence into an embedding vector, followed by stack of self-attention layers that sequentially refine these embedding. The main technique to train language models transformers with MLM is called Bidirectional Encoder Representations from Transformers (BERT) [1]. In BERT, all positions in the sequence can attend to each other allowing the information to flow in both directions, which is essential in the context of DNA sequences. During training, the final embedding of the network is fed to a language model head that transforms it into a probability distribution over the input sequence.

A.1.1 Architecture

All our models follow an encoder-only transformer architecture. An embedding layer transforms sequences of tokens into sequences of embeddings. Positional encodings are then added to each embedding in the sequence to provide the model with positional information. We use a learnable positional encoding layer that accepts a maximum of 1000 tokens. We used 6-mer tokens as a trade-off between sequence length (up to 6kb) and embedding size, and because it achieved the highest performance when compared with other token lengths. The token embeddings are then processed by a transformer layer stack. Each transformer layer transforms its input through a layer normalisation layer followed by a multi-head self-attention layer. The output of the self-attention layer is summed with the transformer layer input through a skip connection. The result of this operation is then passed through a new layer normalisation layer and a two-layer perceptron with GELU activations [50]. The number of heads, the embedding dimension, the number of neurons within the perceptron hidden layer and the total number of layers for each model can be found in Table 1. During self-supervised training, the embeddings returned by the final layer of the stack are transformed by a language model head into a probability distribution over the existing tokens at each position in the sequence.

Our second version Nucleotide Transformer v2 models include a series of architectural changes that proved more efficient: instead of using learned positional embeddings, we use Rotary Embeddings [51] that are used at each attention layer; we use Gated Linear Units with swish activations without bias, making NLPs more efficient. These improved models also accept sequences up to $2,048$ tokens leading to a longer context window of 12kbp.

A.1.2 Training

The models are trained following the BERT methodology [1]. At each training step a batch of tokenized sequences is sampled. The batch size is adapted to available hardware and model size. We conducted all experiments on clusters of A100 GPUs, and took batches of sizes 14 and 2 sequences to train the 500M and 2.5B parameters models, respectively. Within a sequence, of a subset of 15% of tokens, 80% are replaced by a special mask [MASK] token. For training runs on the Human reference genome and multispecies datasets, an additional 10% of the 15% subset of tokens are replaced by randomly selected standard tokens (i.e. any token different from the class [CLS], pad [PAD] or mask [MASK] token), as was done in BERT. For training runs on the 1000G dataset, we skipped this additional data augmentation, as the added noise was greater than the natural mutation frequency present in

the human genome. 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. Gradients were accumulated to reach an effective batch size of 1M tokens per batch. We used the Adam optimizer [52] with a learning rate schedule, and standard values for exponential decay rates and epsilon constants, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and ϵ=1e-8. During a first warmup period, the learning rate was increased linearly between 5e-5 and 1e-4 over 16k steps before decreasing following a square root decay until the end of training.

We slightly modified the hyperparameters of our NT-v2 models: the optimizer and learning rate schedule are kept the same, however we increased batch size to 512 (1,000,000 tokens per batch).
Inspired by Chinchilla scaling laws [46], we also trained our NT-v2 models for longer duration compared to other deep learning models. Specifically, we pre-trained our NT-v2 50M and 250M parameters models for 300B tokens, while our 250M and 500M parameters models were trained for up to 1 trillion tokens to understand the scaling laws at play. In comparison, the NT-v1 2.5B parameters models were trained for 300B tokens, and their 500M counterparts were trained for 50B tokens. In the end we used the following model checkpoints for the NT-v2 models: checkpoint 300B tokens for 50M and 100M models, checkpoint 800B tokens for the 250M model, and checkpoint 900B tokens for the 500M model.

A.1.3 Probing

We refer to probing the assessment of the quality of the model embeddings to solve downstream tasks. After training, for each task, we probe each layer of the model and compare several downstream methods to evaluate in depth the representations capabilities of the model. In other words, given a dataset of nucleotide sequences for a downstream task, we compute and store the embeddings returned by ten layers of the model. Then, using the embeddings of each individual layer as inputs, we trained several downstream models to solve the downstream task. We tested logistic regression with the default hyperparameters from scikit-learn [53] and a multi-layer peceptron. As we observed that the choice of hyperparameters, such as the learning rate, the activation function and the number of layers per hidden layer impacted final performance, we also ran hyperparameters sweeps for each downstream model. We used a 10-fold validation scheme, where the training dataset was split ten times in a training and validation set, that contain different shuffles with 90% and 10% of the initial set. For a given set of hyperparameters, ten models were trained over the ten splits, and their validation performances were averaged. This procedure is run 100 times with a Tree-structured Parzen Estimator solver [54] guiding the search over the hyperparameters space, before evaluating the best performing set of models on the test set. Therefore, for each downstream task, for ten layers of each pre-trained model, the performance on the test set is recorded at the end of the hyperparameters search. The hyperparameters of the best performing probe across the pre-trained models and their layers are reported in Table 7. This probing strategy resulted in 760,000 downstream models trained, which provides detailed analysis into various aspects of training and using LMs, such as the role of different layers on downstream task performance.

As a baseline, we evaluated the performance of a logistic regression model that takes as input the tokenized sequence, i.e. before passing the tokens through the transformer layers. Using the raw tokenized sequences as input yielded much better performance than using a vector where the token ids were one-hot encoded and passed through a pooling layer (summing or averaging, over the sequence length axis).

A.1.4 Fine-tuning

In addition to probing our models through embedding extraction at various layers, we also performed parameter-efficient fine-tuning through the IA^3 technique [34]. Using this strategy, the language model head is replaced by either a classification or regression head depending on the task at hand. The weights of the transformer layers and embedding layers are frozen and new, learnable weights are introduced. For each transformer layer, we introduced three learned vectors $l_k \in \mathbb{R}^{d_k}$, $l_v \in \mathbb{R}^{d_v}$ and $l_{\text{ff}} \in \mathbb{R}^{d_{\text{ff}}}$, which were introduced in the self-attention mechanism as:

$$\text{softmax}\left(\frac{Q(l_k) \odot K^T}{\sqrt{d_k}}\right)(l_v \odot V)$$

and in the position-wise feed-forward networks as $(l_{\text{ff}}\gamma\,(W_1 x))\,W_2$, where γ is the feed-forward network nonlinearity, and $\odot$ represents the element-wise multiplication. This adds a total of $L\,(d_k + d_v + d_{\text{ff}})$ new parameters, where L is the number of transformer layers. We refer to these learnable weights as *rescaling* weights. The intuition is that during fine-tuning these weights will weigh the transformer layers to improve the final representation of the model on a downstream task, so that the classification/regression head can more accurately solve the problem. As we observed layer specialization during probing, we speculate that this fine-tuning technique will similarly select layers with greater predictive ability for particular tasks.

In practice, the number of additional parameters introduced by rescaling weights and the classification/regression head weights represented approximately 0.1% of the total number of weights of the model. This increased fine-tuning speed since just a fraction of parameters needed updating. Similarly, it alleviated storage requirements, needing to create space for just 0.1% new parameters over 500M and 2.5B for each downstream task using traditional fine-tuning. For instance, for the 2.5B parameters models, the weights represent 9.5GB. Considering 18 downstream tasks, classical fine-tuning would have required $9.5 \times 18 = 171$ GBs, whereas parameter-efficient fine tuning required only 171 MB.

Like in the probing scheme, the training dataset was split ten times in a training and validation set, that contain different shuffles with 90% and 10% of the initial set. For each, split, the model was fine-tuned for 10k steps and parameters yielding the highest validation score were then used to evaluate the model on the test set. We used a batch size of eight and the Adam optimizer with a learning rate of 3e-3. Other optimizer parameters were maintained from training regiments. Each model is fine-tuned for 10k steps for each task. These hyperparameters were selected as they led to promising results in the field of NLP [34]. Diverging hyperparameter choices did not yield significant gains in our experiments. We have also compared this approach with fine-tuning from a randomly initialized checkpoint.

A.1.5 Comparison with published pre-trained genomics models

We have compared the fine-tuned performance of Nucleotide Transformer models on the 18 downstream tasks with four different pre-trained models: DNABERT-1 [20], DNABERT-2 [23], HyenaDNA (1kb and 32kb context length; [25]) and Enformer [19]. We ported the architecture and trained weights of each model to our code framework and performed parameter-efficient fine-tuning on the transformer part of every model as described above, using the same cross-validation scheme for a fair comparison. All results can be visulaized in an interactive leader-board [2]. Only for HyenaDNA we performed full fine-tuning due to the incompatibility of our parameter-efficient fine-tuning approach with the model architecture.
Note that the Enformer has been originally trained in a supervised fashion to solve chromatin and gene expression tasks. For the sake of benchmarking, we re-used the provided model torso as a pre-trained model for our benchmark, which is not the intended and recommended use of the original paper. Though we think this comparison is interesting to highlight the differences between self-supervised and supervised learning for pre-training and observe that the Enformer is a very competitive baseline even for tasks that differ from gene expression.

A.2 Datasets

A.2.1 The Human reference genome dataset

The Human reference dataset was constructed by considering all autosomal and sex chromosomes sequences from reference assembly GRCh38/hg38 [3] and reached a total of 3.2 billion nucleotides.

[2] https://huggingface.co/spaces/InstaDeepAI/nucleotide_transformer_benchmark
[3] https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26

A.2.2 The 1000G dataset

To inform the model on naturally occurring genetic diversity in humans, we constructed a training dataset including genetic variants arising from different human populations. Specifically, we downloaded the variant calling format (VCF) files[4] from the 1000 Genomes project [55], which aims at recording genetic variants occurring at a frequency of at least 1% in the human population. The dataset contained 3202 high-coverage human genomes, originating from 27 geographically structured populations of African, American, East Asian, and European ancestry as detailed in Table 2, making up a total of 20.5 trillion nucleotides. Such diversity allowed the dataset to encode a better representation of human genetic variation. To allow haplotype reconstruction in the FASTA format from the VCF files, we considered the phased version of the data, which corresponded to a total of 125M mutations, 111M and 14M of which are single nucleotide polymorphisms (SNPs) and indels, respectively.

A.2.3 The Multispecies dataset

To build a dataset that encompassed a large and diverse set of genomes, we first parsed the genomes available on NCBI[5], before arbitrarily selecting only one species from each genus. Plant and virus genomes were not taken into account, as their regulatory elements differ from those of interest in this work. The resulting collection of genomes was downsampled to a total of 850 species, whose genomes add up to 174 billion nucleotides. The final contribution of each class, in terms of number of nucleotides, to the total number of nucleotides in the dataset, displayed in Table 3, is the same as in the original collection parsed from NCBI. Finally, we enriched this dataset by selecting several genomes that have been heavily studied in the literature (Table 4).

A.3 Data preparation

Once the FASTA files of each genome / individual were collected, they were assembled into one unique FASTA file per dataset that was then pre-processed before training. During this data processing phase, all nucleotides other than A,T,C,G were replaced by N. A tokenizer was employed to convert strings of letters to sequences of tokens. The tokenizer used as alphabet the $4^6 = 4096$ possible 6-mer combinations obtained by combining A,T,C,G, as well as five extra tokens to represent stand-alone A,T,C,G and N. It also included three special tokens, namely the padding [pad], masking [mask] and the beginning of sequence (also called class; [CLS]) token. This adds to a vocabulary of 4104 tokens. To tokenize an input sequence, the tokenizer will start with a class token and then convert the sequence starting from the left, matching 6-mer tokens when possible, or falling back on the stand-alone tokens when needed (for instance when the letter N is present or if the sequence length is not a multiple of 6).

For the multispecies and Human reference dataset, genomes are split into overlapping chunks of 6100 nucleotides, each sharing the first and last 50 nucleotides with the previous and last chunk, respectively. As a data augmentation exercise, for each epoch and chunk, a starting nucleotide index is randomly sampled between 0 and 100, and the sequence is then tokenized from this nucleotide until 1000 tokens is reached. The number of epochs was determined depending on the dataset so that the model processed a total of 300B tokens during training. At each step, a batch of sequences sampled randomly within the epoch set was fed to the model. For the 1000G dataset, batches of sequences from the Human reference genome, prepared as specified above, are sampled at each step. Then, for each sampled chunk, an individual from the 1000G dataset is randomly selected, and if that individual carries mutations at the positions and chromosome corresponding to that chunk, these mutations are introduced into the sequence, and the corresponding tokens replaced. This data processing technique ensured uniform sampling both over the genome and over the individuals during training, as well as enabled to efficiently store only mutations for each individual, instead of full genomes.

[4] http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000G_2504_high_coverage/working/20201028_3202_phased/

[5] https://www.ncbi.nlm.nih.gov/

A.3.1 Hardware

All models were trained on the Cambridge-1 Nvidia supercomputer system, using 16 nodes, each equipped with eight A100 GPUs, leading to a total of 128 A100 GPUs used. During training, model weights were replicated on each GPU, while batches were sharded across GPUs. Gradients were computed on each shard and accumulated before being averaged across devices and backpropagated. We relied on the jax library[6] that relied on the NCCL[7] protocol to handle communications between nodes and devices, and observed almost linear decrease of the training time with respect to the number of GPU available. The 500M parameters models were trained on a single node for a day, while the 2.5B models models required the whole cluster for 28 days to be trained. The Nucleotide Transformer version 2 (NT-v2) models with varying number of parameters ranging from 50M to 500M were similarly trained on a single node for a single day. All fine-tuning runs were performed on a single node with eight A100 GPUs. As for the training runs, the models weights were replicated and batches distributed across GPUs. As we used a batch size of eight for fine-tuning, each GPU processed a single sample before averaging the gradients and applying them. On average, a fine-tuning run lasted 20 minutes for the 500M parameter models, and 50 minutes for the 2.5B parameter models.

For the probing experiments, all embeddings (for all sequences in all downstream tasks, for selected layers of each model) were computed and stored on a single node with eight A100 GPUs, requiring two days to compute. Then, 760,000 downstream models were fit on a cluster of 3000 CPUs, requiring 2.5 days.

A.4 Downstream tasks

A.4.1 Epigenetic marks prediction

We downloaded the dataset[8] of epigenetic marks measured in the yeast genome, namely acetylation and metylation nucleosome occupancies [28]. Nucleosome occupancy values in these ten datasets were obtained with Chip-Chip experiments [56] and further processed into positive and negative observations to provide epigenetic training data for the following histone marks: H3, H4, H3K9ac, H3K14ac, H4ac, H3K4me1, H3K4me2, H3K4me3, H3K36me3 and H3K79me3.

A.4.2 Promoter sequence prediction

We built a dataset of promoter sequences to evaluate the capabilities of the model to identify promoter motifs. Following the DeePromoter method [31], we considered sequences of 300 base pairs (bp) genome-wide, selected to span 249bp upstream and 50bp downstream of transcription start sites. This resulted in 29,597 promoter regions, 3,065 of which were TATA-box promoters. For each promoter region sample, a negative sample (non-promoter sequence) with matching length was constructed by splitting the promoter sequence in 20 sub-sequences and randomly selecting and shuffling 12 of them, while keeping the other 8 intact. The final dataset is then composed of 59,194 sequences.

A.4.3 Enhancer sequence prediction

We used an enhancer dataset presented prior [29] to evaluate the capacity of the transformer models to provide an accurate representation of enhancer sequences. The original dataset included 742 strong enhancers, 742 weak enhancers and 1484 non-enhancers. As suggested by previous work[29] to increase training performance, we augmented this dataset with 6000 synthetic enhancers and 6000 synthetic non-enhancers produced through a generative model.

A.4.4 Splice site prediction

We used two datasets to evaluate splice site prediction. First, we downloaded the dataset used by SpliceFinder [30], which was composed of donor, acceptor, and non-splice sites, containing sequences detected in human genes. Each sequence was 400 nucleotides long and contained either a splicing site

[6] https://jax.readthedocs.io/en/latest/_autosummary/jax.pmap.html
[7] https://developer.nvidia.com/nccl
[8] http://www.jaist.ac.jp/~tran/nucleosome/members.htm

in the center (i.e. an acceptor or donor site), or a non-splicing site. This is a multi-label prediction task with labels being acceptor, donor or none (Splice site all). Second, to leverage a more diverse set of splicing sites, we also downloaded the training dataset[9] of the Spliceator model [32]. Contrary to the SpliceFinder dataset, this set was based primarily on the G3PO database, which included sequences from 147 phylogenetically diverse organisms (ranging from protists to primates, including humans). All sequences were 600bp and included either a splicing site at the center (i.e. an acceptor or donor site), or a non-splicing site. Here we created two binary classification tasks: one for splice acceptor prediction (Splice acceptor) and other for donor prediction (Splice donor). Following the Spliceator study, we only included sequences that were part of the balanced 'Gold Standard' dataset (referred to as 'GS_1' in the study).

A.4.5 Chromatin Profiles Prediction

We used the dataset[10] compiled in Zhou et al. 2015 [10] for chromatin profiles prediction. The dataset is composed of 2.4 million sequences, each of size 1000 nucleotides, and associated with 919 chromatin features. These include 690 transcription factor (TF), 125 DNAse, and 104 histone features. As in the original publication, our model is trained simultaneously on the 919 classification tasks, with 919 independent classification heads, and a loss taken as the average of the cross entropy losses. Since each label is highly unbalanced and is composed mostly of negative samples, the losses associated with positive samples are upscaled by a factor of 8. Contrarily to the DeepSEA method [10], which trained two models independently, one on the forward sequences and one on the corresponding reverse-complementary, and evaluated the average of their predictions, the model presented here was trained only on the forward sequences.

A.4.6 SpliceAI benchmark

We used the scripts available at the Illumina Basespace platform [11] to reproduce the training dataset presented in SpliceAI [35]. Briefly, this training dataset is constructed using GENCODE v24lift37 annotations and RNA-seq data from the GTEx cohort, focusing solely on splice site annotations from the principal transcript. The training dataset comprises annotations from genes located on chromosomes 2, 4, 6, 8, and 10-22, as well as chromosomes X and Y, while annotations from genes on the remaining chromosomes, which were not paralogs, constitute the test dataset. Each sequence produced by this pipeline has a length of 15,000 bp, with the central 5,000 bp containing the sites to be predicted. Additional details about the construction of the training dataset can be found in the original publication. We adapted this original SpliceAI dataset to be able to run our models by reducing the sequence length to 6,000 bp (for NT-v1 model) and 12,000 bp (for NT-v2 model), reducing the flanking contexts but keeping the central 5,000 bp. We also removed sequences that contained Ns. When comparing against SpliceAI on the first dataset, which we refer to as SpliceAI-6k, we appended 9,000 "N" nucleotides as flanking sequence since SpliceAI is based on a model with a 15,000 bp input. When comparing against SpliceAI on the second dataset, we report the performance presented in the original publication, which, compared to this dataset, includes a sequence length of 15,000 bp instead of 12,000 bp.
This task is a multi-label classification for each of the input sequence's nucleotides similar to Splice AI [35] (Fig. 5d). From each embedding outputted by the transformer model, a head predicts, for each of the 6 nucleotides represented by the token embedding, three label probabilities: splice acceptor, splice donor or none. The head is a simple classification layer that predicts 18 classes, i.e 3 labels for each of the 6 nucleotides. To ensure that each embedding is associated with a 6-mer, the sequences are cut so that their length is divisible by 6. Furthermore, all sequences with Ns are removed from both training and test set, which represents a neglectable portion of the data. Note that if we were to use a Byte Pair Encoding tokenizer like DNABERT-2 [23], the number of nucleotides represented by each embedding would vary and make nucleotide-level prediction tasks substantially trickier to implement.

[9] https://git.unistra.fr/nscalzitti/spliceator/-/tree/master/Data/Datasets
[10] http://deepsea.princeton.edu/media/code/deepsea_train_bundle.v0.9.tar.gz
[11] https://basespace.illumina.com/projects/66029966/

A.4.7 Enhancer Activity Prediction

We used the enhancer activity dataset [12] released in de Almeida et al. 2022 [12]. The dataset is composed of 484,052 DNA sequences of size 249 nucleotides, each measured for their quantitative enhancer activity towards a developmental or a housekeeping promoter. We added two independent regression heads to our models to predict both enhancer activities in simultaneous. Following the methodology used in [3], we chose to treat this regression task as a multi-label classification problem. Specifically, each label y was discretized over a set of 50 values $(b_i)_{i\in[1,50]}$, evenly spaced between the minimum and maximum value. For each label, the model predicts the normalized weights $(w_i)_{i\in[1,50]}$ such that

$$y = \sum_{i=1}^{50} w_i b_i.$$

A.4.8 Performance Metrics

The baselines considered here used different performance metrics, as each study used datasets that varied in size and number of positive/negative observations (Table 5). In order to benchmark the transformer models presented here, we decided to use the following ranking across performance metrics: The Matthews Correlation Coefficient (MCC) was preferred over the F1-scores, which was preferred over accuracy. The term *performance*, which is used throughout the text, should therefore be considered as an umbrella term to refer to the performance used by each baseline. For a fair comparison to all the baselines, we used the same train/test folds as respective models. For a final comparison across models, we calculated for each model the mean MCC across the 3 different categories of task, where for each category we use the median MCC across tasks.

A.5 Additional Performance Analysis

A.5.1 t-SNE projections of embeddings

T-distributed Stochastic Neighbor Embedding (t-SNE) was used to reduce Nucleotide Transformer inner embeddings to 2-D vectors to visualize the separation of different genomic elements. Nucleotide Transformer embeddings for each genomic element were computed at several transformer layer outputs and the mean embeddings computed across the sequence locations corresponding to the element were calculated. These mean embeddings were then passed as input into a TSNE reducer object with default parameters from the sklearn python package [57].

A.5.2 Reconstruction accuracy and perplexity

We studied how pre-trained models could reconstruct masked tokens. We considered a trained language model with parameters θ. Within a nucleotide sequence $\mathbf{s}$ of interest, we masked tokens using one of two strategies (i.e. we replaced the tokens at these positions with the mask token [MASK]). We either masked the central token of the sequence only, or we masked randomly 15% of the tokens within the sequence. The masked sequence is then fed to the model and the probabilities over tokens at each masked position are retrieved. The loss function $l(\theta, \mathbf{s})$ and the accuracy $acc(\theta, \mathbf{s})$ are defined as follows:

$$\begin{cases} l(\theta, \mathbf{s}) = \sum\limits_{i\in\mathcal{P}_{\text{masked}}} \sum\limits_{\text{tok}\in\mathcal{V}} \log p(\theta, i, \text{tok}) \cdot \mathbf{1}\left(\text{tok} = \mathbf{s}(i)\right) \\ acc(\theta, \mathbf{s}) = \dfrac{1}{|\mathcal{P}_{\text{masked}}|} \sum\limits_{i\in\mathcal{P}_{\text{masked}}} \mathbf{1}\left(\underset{\text{tok}\in\mathcal{V}}{\text{argmax}}\left(\log p(\theta, i, \text{tok})\right) = \mathbf{s}(i)\right) \end{cases}$$

where $\mathcal{P}_{\text{masked}}$ is the set of masked positions and $\mathcal{V}$ is the vocabulary, i.e. the set of all existing tokens. The perplexity is usually defined in the context of autoregressive generative models. Here, we rely on an alternative definition used in Rives [4], and define it as the exponential of the loss function computed over the masked positions:

$$\text{perplexity}(\theta, \mathbf{s}) = 2^{l(\theta, \mathbf{s})}. \tag{1}$$

[12] https://zenodo.org/record/5502060#.Y9qO7hzMLUc

Perplexity measures how well a model can reconstruct masked positions, and is a more refined measure than accuracy, as it does also account for magnitude. In contrast to accuracy, lower perplexity suggests better reconstruction ability, thus better performance.

A.5.3 Reconstruction of tokens in different genomics elements

We have also performed this token reconstruction approach across the full chromosome 22 in 6kb windows. We only kept windows without Ns. For each window masked each token at a time, recovering the predicted probability for the original token in the sequence. We display these scores as a WashU Epigenome Browser session [13]. To obtain average token probabilities across different types of genomic elements, we retrieved gene annotation regions from Ensembl (exons, introns, splice acceptors and donors, 5'UTR, 3'UTR, TSS, TES [14]), polyA signal sites from GENCODE [15] and regulatory elements from ENCODE (enhancers, promoters and CTCF-bound sites from SCREEN database [16]).

A.5.4 Functional variant prioritization

To obtain genetic variants with varying levels of associated severity we used the Variant Effect Prediction (VEP) software [41] and annotated sequences across the human genome. Specifically, we randomly sampled sequences throughout the human genome, and kept genetic variants within those sequences annotated to any of the following classes: "intron variant", "intergenic variant", "regulatory region variant", "missense variant", "3 prime UTR variant", "synonymous variant", "TF binding site variant", "5 prime UTR variant", "splice region variant", and "stop gained variant". After keeping only single nucleotide polymorphisms and filtering out variants annotated to more than one consequence (e.g. those annotated as stop gained and splice variants), we obtained a final dataset composed of 920 genetic variants per class.

As a positive set of functional genetic variants we compiled SNPs from four different resources. We used SNPs associated to gene expression (i.e. expression quantitative trait loci [eQTLs]) and to methylation variation (i.e., meQTLs) from the Genome-Wide Repository of Associations Between SNPs and Phenotypes (GRASP) database [58] with a P-value $<10^{-12}$, SNPs with "likely pathogenic "annotations from ClinVar [59] and SNPs reported in The Human Gene Mutation Database (HGMD) (public version 2020.4) [60]. After these filters, we retained a total of 80,590, 11,734, 70,224, and 14,626 genetic variants for the eQTLs, meQTLs, ClinVar, and HGMD SNP datasets, respectively. For each of these four datasets, we then constructed a set of negative variants based on SNPs from the 1000 Genomes Project with a minor allele frequency (MAF) >5%, that did not overlap with any variant reported in the dataset tested, and that were within 100kb of the associated variants, resulting in four balanced datasets.

To compute zero-shot based scores for a given site of interest we did the following: For each SNP, we obtained a 6,000 bp sequence centered on the SNP of interest based on the human reference genome. We then, created two sequences, one carrying the reference allele and a second carrying the alternative allele at the SNP position. We then computed several zero-shot scores that capture different aspects of the vector distances in the embedding space between those two sequences, namely: the L1 distance (Manhattan), (ii) the L2 distance (Euclidean), (iii) the cosine similarity, and (iv) the dot-product (not normalized cosine similarity). We also computed the loss of the alternative allele and the difference in the loss between the sequence carrying the alternative and reference alleles, as two additional zero-shot scores. In the case of functional variants, in addition to zero-shot scores, we also fine-tuned the transformer models to classify positive and negative variants. We employed a similar strategy as the one previously described, with the primary difference being that the training and test sets were divided by chromosomes and strictly kept non-overlapping. Specifically, we divided the 22 chromosomes into 5 sets and sequentially used each of them as a test set and the 4 others as a training set. By fine-tuning on the training set, we could derive probabilities of being a positive variant for each sequence in the

[13] https://shorturl.at/jov28
[14] https://www.ensembl.org/info/data/ftp/index.html
[15] https://www.gencodegenes.org/human/
[16] https://api.wenglab.org/screen_v13/fdownloads/GRCh38-ccREs.bed

test set. We use those probabilities as a score for each SNP.

To compare these predictions against other methods, we randomly sampled 10,000 positive and negative SNPs from each of the four datasets. We then used the Combined Annotation Dependent Depletion tool (version GRCh38-v1.6) to compute CADD, GERP, phastCons, and phyloP scores. The DeepSEA scores were computed using the Beluga model available at: https://hb.flatironinstitute.org/sei/. The score considered was the "disease impact score"reported for each SNP.

A.5.5 Attention Maps Analysis

We analysed how attention maps gathered from the pre-trained models capture key genomic elements. We followed a methodology proposed in previous work [39]. For a genomic element, we define the indicator function over tokens $f(i)$ that equals 1 if one or several nucleotides within token i belong to the element, and 0 otherwise. We computed the average proportion of attention focused on that genomic element, in one attention head, aggregated over a dataset of nucleotide sequences $\mathbf{X}$ as:

$$p_\alpha(f) = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} \frac{\sum_i \sum_j f(i) \mathbf{1}\left(\alpha(i,j) > \mu\right)}{\sum_i \sum_j \mathbf{1}\left(\alpha(i,j) > \mu\right)}$$

where $\alpha(i,j)$ is the attention coefficient between tokens i and tokens j defined such that $\sum_i \alpha_{i,j} = 1$ and μ is a confidence threshold.

We computed the values of $p_\alpha(f)$ for all the heads and all the layers of all models, and considered nine elements ("5' UTR", "3' UTR", "exon", "intron", "enhancer", "promoter", "CTCF binding site", "open chromatin", and "transcription factor binding sites"). We perform these analyses over a dataset made of 90,000 sequences, 10,000 per feature, of length 6k bp extracted from the Human reference genome. The average proportion of tokens belonging to each element can be found in Table 9. For each sequence, the position of the feature within the sequence was sampled uniformly during the dataset creation. As suggested in previous work [39], we selected a confidence threshold $\mu = 0.3$ for all experiments.

We considered that a feature is captured by an attention head if the quantity $p_\alpha(f)$ is significantly greater than the natural occurring frequency of the feature within the dataset (Table 9). To validate this, we conducted a two proportion z-test with the null hypothesis as the natural frequency of the feature, and the alternate hypothesis as $p_\alpha(f)$. The total number of heads of each model is used as a Bonferroni correction to the significance level, α, of 0.05. We computed z-scores and associated p-value for each head in every model for every genomic element as follows:

$$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}\left(1 - \hat{p}\right)\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

where $\hat{p}_1$ represents the proportion of attention above μ associated with each genomic element, $\hat{p}_2$ represents the proportion of the sequence occupied by the genomic element, n_1 is the total number of sequence positions with attention above μ, n_2 is the total number of sequence positions. Attention heads with p-values below the Bonferonni corrected significance level are considered to be significant.

A.5.6 Prediction of important TF motif instances from DeepSTARR data

We retrieved experimental mutagenesis data from the DeepSTARR dataset [12] where individual TF motif instances are mutated and their impact is measured in the activity of developmental and housekeeping enhancers. We assessed the performance of the fully-finetuned NT2.5B multispecies model (since it was the model with the highest test set performance) to predict the contribution of each TF motif instance by predicting the activity of the wildtype and respective motif-mutant sequence and calculating their log2 fold-change. We compared our predicted mutation effects with the ones predicted by the original method DeepSTARR and the experimentally derived log2 fold-changes.

B Supplementary Tables

	NT-1000G (2.5B)	NT-Multispecies (2.5B)	NT-1000G (500M)	NT-HumanRef (500M)	NT-Multispecies-v2 (500M)	NT-Multispecies-v2 (250M)	NT-Multispecies-v2 (100M)	NT-Multispecies-v2 (50M)
alphabet	k-mers	k-mers	k-mers	k-mers	k-mers	k-mers	k-mers	k-mers
k_for_kmers	6	6	6	6	6	6	6	6
num_warmup_updates	16000	16000	16000	16000	16000	16000	16000	16000
warmup_init_lr	5E-05	5E-05	5E-05	5E-05	5E-05	5E-05	5E-05	5E-05
warmup_end_lr	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
training_set_proportion	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
masking_ratio	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
masking_prob	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
random_token_prob	0	0.1	0	0.1	0.1	0.1	0.1	0.1
alphabet_size	4105	4105	4105	4105	4105	4105	4105	4105
prepend_bos	True	True	True	True	True	True	True	True
append_eos	False	False	False	False	False	False	False	False
attention_heads	20	20	20	20	16	16	16	16
embed_dim	2560	2560	1280	1280	1024	768	512	512
ffn_embed_dim	10240	10240	5120	5120	4096	3072	2048	2048
num_layers	32	32	24	24	29	24	22	12
token_dropout	True	True	True	True	False	False	False	False
num_parameters	2547800585	2547800585	485729545	485729545	496245771	233545995	96839179	54855179
dataset	1000G	Multispecies	1000G	Human Reference	Multispecies	Multispecies	Multispecies	Multispecies

Supplementary Table 1: Models hyperparameters.

Population name	Population code	Number of individuals
African Ancestry SW	ASW	74
African Caribbean	ACB	116
Bengali	BEB	131
British	GBR	91
CEPH	CEU	179
Colombian	CLM	132
Dai Chinese	CDX	93
Esan	ESN	149
Finnish	FIN	99
Gambian Mandinka	GWD	178
Gujarati	GIH	103
Han Chinese	CHB	103
Iberian	IBS	157
Japanese	JPT	104
Kinh	KHV	2
Kinh Vietnamese	KHV	120
Luhya	LWK	99
Mende	MSL	99
Mexican Ancestry	MXL	97
Peruvian	PEL	122
Puerto Rican	PUR	139
Punjabi	PJL	146
Southern Han Chinese	CHS	163
Tamil	STU	114
Telugu	ITU	107
Toscani	TSI	107
Yoruba	YRI	178

Supplementary Table 2: Number of individuals per population in the 1000G dataset.

Class	Number of species	Number of nucleotides (B)
Bacteria	667	17.1
Fungi	46	2.3
Invertebrate	39	20.8
Protozoa	10	0.5
Mammalian Vertebrate	31	69.8
Other Vertebrate	57	63.4

Supplementary Table 3: Genomes in the multispecies dataset.

Class	Selected Species
Bacteria	Escherichia coli
Fungi	Saccharomyces cerevisiae
Invertebrate	Caenorhabditis elegans, Drosophila melanogaster
Protozoa	Plasmodium vivax, Plasmodium falciparum
Mammalian Vertebrate	Homo sapiens, Mus musculus, Rattus norvegicus
Other Vertebrate	Danio rerio, Xenopus tropicalis

Supplementary Table 4: Model organisms genomes in the multispecies dataset.

	Num train sequences	Num test sequences	Max sequence length in bp	Reported Metric	Baseline Name
H3K4me3	25953	2884	500	MCC	SVM [28]
H3K4me2	27614	3069	500	MCC	SVM [28]
H3K36me3	31392	3488	500	MCC	SVM [28]
H3K9ac	25003	2779	500	MCC	SVM [28]
Splice donor	19775	2198	600	F1	Spliceator [32]
Splice site all	27000	3000	400	Acc	SpliceFinder [30]
H4ac	30685	3410	500	MCC	SVM [28]
H3K4me1	28509	3168	500	MCC	SVM [28]
Enhancer	14968	400	200	MCC	LSTM-CNN [29]
Enhancer types	14968	400	200	MCC	LSTM-CNN [29]
H4	13140	1461	500	MCC	SVM [28]
Splice acceptor	19961	2218	600	F1	Spliceator [32]
H3K79me3	25953	2884	500	MCC	SVM [28]
Promoter non-TATA	47767	5299	300	F1	DeePromoter [31]
Promoter all	53276	5920	300	F1	DeePromoter [31]
H3K14ac	29743	3305	500	MCC	SVM [28]
H3	13468	1497	500	MCC	SVM [28]
Promoter TATA	5509	621	300	F1	DeePromoter [31]

Supplementary Table 5: Downstream tasks metadata.

Dataset	H3	H3K14ac	H3K36me3	H3K4me1	H3K4me2	H3K4me3
DNABERT-1	0.763 (± 0.013)	0.403 (± 0.012)	0.474 (± 0.009)	0.396 (± 0.005)	0.282 (± 0.013)	0.258 (± 0.01)
DNABERT-2	**0.785 (± 0.012)**	0.515 (± 0.009)	0.591 (± 0.005)	0.512 (± 0.008)	0.333 (± 0.013)	0.353 (± 0.021)
Enformer	0.724 (± 0.018)	0.284 (± 0.024)	0.345 (± 0.019)	0.291 (± 0.016)	0.207 (± 0.021)	0.156 (± 0.022)
HyenaDNA-1KB	**0.781 (± 0.015)**	**0.608 (± 0.02)**	**0.614 (± 0.014)**	0.512 (± 0.008)	**0.455 (± 0.028)**	**0.55 (± 0.015)**
HyenaDNA-32KB	0.747 (± 0.017)	0.405 (± 0.015)	0.479 (± 0.015)	0.387 (± 0.017)	0.276 (± 0.01)	0.291 (± 0.025)
NT 2.5B Randomly Initialized	0.731 (± 0.105)	0.419 (± 0.103)	0.188 (± 0.01)	0.367 (± 0.077)	0.257 (± 0.039)	0.289 (± 0.074)
NT 500M Randomly Initialized	0.687 (± 0.116)	0.347 (± 0.083)	0.408 (± 0.086)	0.318 (± 0.057)	0.219 (± 0.055)	0.216 (± 0.06)
NT-1000G (2.5B)	0.754 (± 0.008)	0.453 (± 0.008)	0.53 (± 0.012)	0.418 (± 0.012)	0.278 (± 0.015)	0.311 (± 0.012)
NT-1000G (500M)	0.736 (± 0.012)	0.381 (± 0.011)	0.468 (± 0.008)	0.38 (± 0.009)	0.26 (± 0.016)	0.235 (± 0.008)
NT-HumanRef (500M)	0.718 (± 0.014)	0.372 (± 0.006)	0.448 (± 0.012)	0.361 (± 0.015)	0.27 (± 0.015)	0.236 (± 0.018)
NT-Multispecies (2.5B)	**0.793 (± 0.013)**	0.538 (± 0.009)	**0.618 (± 0.011)**	**0.541 (± 0.005)**	0.324 (± 0.014)	0.408 (± 0.011)
NT-Multispecies-v2 (100M)	**0.787 (± 0.012)**	0.521 (± 0.007)	0.594 (± 0.005)	0.523 (± 0.008)	0.303 (± 0.013)	0.376 (± 0.009)
NT-Multispecies-v2 (250M)	**0.792 (± 0.009)**	0.54 (± 0.008)	**0.621 (± 0.007)**	**0.541 (± 0.005)**	0.322 (± 0.009)	0.403 (± 0.01)
NT-Multispecies-v2 (500M)	**0.786 (± 0.012)**	0.549 (± 0.007)	**0.624 (± 0.004)**	**0.55 (± 0.007)**	0.32 (± 0.012)	0.406 (± 0.01)
NT-Multispecies-v2 (50M)	**0.792 (± 0.01)**	0.511 (± 0.006)	0.582 (± 0.007)	0.514 (± 0.007)	0.299 (± 0.016)	0.333 (± 0.016)
Raw Probe	0.295 (± 0.005)	0.013 (± 0.006)	0.052 (± 0.006)	0.053 (± 0.01)	0.086 (± 0.009)	0.062 (± 0.005)

Dataset	H3K79me3	H3K9ac	H4	H4ac	enhancers	enhancers types
DNABERT-1	0.578 (± 0.007)	0.505 (± 0.009)	0.784 (± 0.007)	0.359 (± 0.01)	0.495 (± 0.016)	0.367 (± 0.034)
DNABERT-2	0.615 (± 0.01)	0.545 (± 0.009)	**0.797 (± 0.008)**	0.465 (± 0.013)	**0.525 (± 0.026)**	**0.423 (± 0.018)**
Enformer	0.498 (± 0.013)	0.415 (± 0.02)	0.735 (± 0.023)	0.275 (± 0.022)	0.454 (± 0.029)	0.312 (± 0.043)
HyenaDNA-1KB	**0.669 (± 0.014)**	**0.586 (± 0.021)**	0.763 (± 0.012)	**0.564 (± 0.011)**	**0.52 (± 0.031)**	**0.403 (± 0.056)**
HyenaDNA-32KB	0.567 (± 0.013)	0.472 (± 0.01)	0.761 (± 0.017)	0.379 (± 0.012)	0.489 (± 0.018)	0.352 (± 0.04)
NT 2.5B Randomly Initialized	0.533 (± 0.086)	0.181 (± 0.015)	0.755 (± 0.094)	0.117 (± 0.017)	0.379 (± 0.035)	0.425 (± 0.05)
NT 500M Randomly Initialized	0.505 (± 0.082)	0.408 (± 0.104)	0.711 (± 0.103)	0.303 (± 0.092)	0.507 (± 0.053)	0.26 (± 0.041)
NT-1000G (2.5B)	0.574 (± 0.007)	0.491 (± 0.009)	0.787 (± 0.006)	0.408 (± 0.009)	**0.546 (± 0.029)**	**0.432 (± 0.03)**
NT-1000G (500M)	0.562 (± 0.015)	0.479 (± 0.01)	0.755 (± 0.011)	0.342 (± 0.014)	**0.509 (± 0.033)**	0.395 (± 0.027)
NT-HumanRef (500M)	0.566 (± 0.016)	0.451 (± 0.011)	0.75 (± 0.011)	0.332 (± 0.017)	**0.502 (± 0.026)**	**0.429 (± 0.018)**
NT-Multispecies (2.5B)	0.623 (± 0.01)	0.547 (± 0.011)	**0.808 (± 0.007)**	0.492 (± 0.014)	**0.545 (± 0.028)**	**0.444 (± 0.022)**
NT-Multispecies-v2 (100M)	0.605 (± 0.008)	0.541 (± 0.008)	**0.792 (± 0.009)**	0.48 (± 0.004)	0.491 (± 0.023)	**0.413 (± 0.02)**
NT-Multispecies-v2 (250M)	0.616 (± 0.014)	0.55 (± 0.007)	**0.797 (± 0.006)**	0.501 (± 0.008)	**0.534 (± 0.033)**	**0.451 (± 0.024)**
NT-Multispecies-v2 (500M)	0.63 (± 0.009)	**0.567 (± 0.014)**	**0.799 (± 0.007)**	0.496 (± 0.009)	**0.559 (± 0.04)**	**0.438 (± 0.043)**
NT-Multispecies-v2 (50M)	0.594 (± 0.008)	0.526 (± 0.011)	**0.805 (± 0.008)**	0.46 (± 0.011)	**0.516 (± 0.033)**	**0.411 (± 0.025)**
Raw Probe	0.081 (± 0.007)	0.134 (± 0.007)	0.15 (± 0.007)	0.113 (± 0.007)	0.391 (± 0.007)	0.263 (± 0.003)

Dataset	H3K79me3	H3K9ac	H4	H4ac	enhancers	enhancers types
DNABERT-1	0.961 (± 0.001)	0.962 (± 0.002)	0.956 (± 0.006)	NaN	0.975 (± 0.002)	NaN
DNABERT-2	0.972 (± 0.002)	0.972 (± 0.002)	**0.955 (± 0.006)**	0.975 (± 0.002)	0.939 (± 0.003)	0.963 (± 0.002)
Enformer	0.955 (± 0.002)	0.955 (± 0.003)	**0.959 (± 0.006)**	0.915 (± 0.007)	0.847 (± 0.005)	0.906 (± 0.008)
HyenaDNA-1KB	0.959 (± 0.002)	0.959 (± 0.002)	0.944 (± 0.011)	0.959 (± 0.003)	0.956 (± 0.004)	0.947 (± 0.007)
HyenaDNA-32KB	0.956 (± 0.002)	0.954 (± 0.003)	0.939 (± 0.008)	0.96 (± 0.007)	0.962 (± 0.004)	0.957 (± 0.006)
NT 2.5B Randomly Initialized	0.956 (± 0.163)	0.963 (± 0.088)	0.951 (± 0.092)	0.97 (± 0.109)	0.676 (± 0.03)	0.971 (± 0.073)
NT 500M Randomly Initialized	0.936 (± 0.123)	0.945 (± 0.084)	0.945 (± 0.096)	0.948 (± 0.119)	0.956 (± 0.115)	0.95 (± 0.09)
NT-1000G (2.5B)	0.965 (± 0.002)	0.967 (± 0.001)	**0.957 (± 0.007)**	0.98 (± 0.002)	0.976 (± 0.002)	0.979 (± 0.001)
NT-1000G (500M)	0.951 (± 0.003)	0.951 (± 0.002)	0.936 (± 0.006)	0.965 (± 0.002)	0.968 (± 0.001)	0.971 (± 0.001)
NT-HumanRef (500M)	0.952 (± 0.002)	0.952 (± 0.002)	0.942 (± 0.006)	0.962 (± 0.003)	0.97 (± 0.002)	0.971 (± 0.002)
NT-Multispecies (2.5B)	**0.975 (± 0.002)**	**0.977 (± 0.002)**	**0.959 (± 0.004)**	**0.986 (± 0.002)**	**0.982 (± 0.002)**	**0.987 (± 0.001)**
NT-Multispecies-v2 (100M)	0.969 (± 0.001)	0.97 (± 0.001)	0.954 (± 0.003)	0.979 (± 0.002)	0.979 (± 0.002)	**0.981 (± 0.001)**
NT-Multispecies-v2 (250M)	0.972 (± 0.001)	**0.974 (± 0.003)**	**0.962 (± 0.003)**	0.982 (± 0.001)	**0.98 (± 0.001)**	**0.982 (± 0.003)**
NT-Multispecies-v2 (500M)	**0.976 (± 0.002)**	**0.976 (± 0.002)**	**0.965 (± 0.004)**	**0.981 (± 0.003)**	**0.984 (± 0.002)**	**0.987 (± 0.006)**
NT-Multispecies-v2 (50M)	0.96 (± 0.001)	0.962 (± 0.002)	0.947 (± 0.004)	0.977 (± 0.003)	0.975 (± 0.002)	0.973 (± 0.002)
Raw Probe	0.658 (± 0.001)	0.67 (± 0.037)	0.649 (± 0.005)	0.679 (± 0.003)	0.635 (± 0.002)	0.708 (± 0.023)

Supplementary Table 6: Downstream performance per task for all models. Every model was fine-tuned with PEFT, except for "Raw Probe" that used probing strategy.

	Model	Layer probed	Number of hidden layers	Hidden layer dimension	Batch size	Learning rate
H3K4me3	2.5B MS	17	1	256	500	1e-4
H3K4me2	2.5B MS	10	1	256	500	1e-4
H3K36me3	2.5B MS	17	1	256	500	1e-4
H3K9ac	2.5B MS	10	1	256	500	1e-4
Splice site all	2.5B MS	14	1	512	650	6.1e-5
H4ac	2.5B MS	17	1	256	500	1e-4
H3K4me1	2.5B MS	14	2	512	500	1e-4
Enhancer	500M HR	18	2	512	500	1e-4
H4	2.5B MS	28	1	256	500	1e-4
Splice acceptor	2.5B MS	10	1	512	500	1e-4
H3K79me3	2.5B MS	14	2	512	650	1.8e-4
Promoter non-TATA	2.5B MS	10	1	512	500	1e-4
Promoter all	2.5B MS	10	1	512	500	1e-4
H3K14ac	2.5B MS	14	1	512	500	1e-4
H3	2.5B MS	14	1	256	500	1e-4
Promoter TATA	2.5B MS	10	1	512	500	1e-4

Supplementary Table 7: Hyperparameters used to train the probing model with the best cross-validation performance for each of the downstream task, along with the model and the layer used to compute the corresponding embeddings. The best probe method for the tasks **Enhancer types** and **Splice donor** was the logistic regression with default hyperparameters.

Model name	Model type	Approach	Average performance
Raw Probe	Control	Probing	0.283
NT-HumanRef (500M)	NT-v1	Probing	0.582
NT-1000G (500M)	NT-v1	Probing	0.602
NT-1000G (2.5B)	NT-v1	Probing	0.619
NT-Multispecies (2.5B)	NT-v1	Probing	0.661
Peer-Reviewed Baselines	Baseline	Fine-tuned	0.674
Enformer	Enformer	Fine-tuned	0.680
NT 500M Randomly Initialized	Control	Fine-tuned	0.718
NT 2.5B Initialized	Control	Fine-tuned	0.724
HyenaDNA-32KB	HyenaDNA	Fine-tuned	0.743
NT-HumanRef (500M)	NT-v1	Fine-tuned	0.745
NT-1000G (500M)	NT-v1	Fine-tuned	0.747
DNABERT-1	DNABERT	Fine-tuned	0.771
NT-1000G (2.5B)	NT-v1	Fine-tuned	0.782
DNABERT-2	DNABERT	Fine-tuned	0.789
NT-Multispecies-v2 (50M)	NT-v2	Fine-tuned	0.790
HyenaDNA-1KB	HyenaDNA	Fine-tuned	0.798
NT-Multispecies-v2 (100M)	NT-v2	Fine-tuned	0.801
NT-Multispecies-v2 (250M)	NT-v2	Fine-tuned	0.811
NT-Multispecies (2.5B)	NT-v1	Fine-tuned	0.812
NT-Multispecies-v2 (500M)	NT-v2	Fine-tuned	**0.821**

Supplementary Table 8: Normalized summed downstream performance of all models, sorted by performance. These values are obtained by averaging the median MCC score obtained by each model on the three categories of downstream tasks: Chromatin Profiles, Regulatory elements and Splicing. Information about model type and probing/fine-tuning approach is also provided.

	Percentage of sequences containing this element	Mean Percentage of Sequence Length belonging to this element
Enhancer	10.18%	11.09%
Open chromatin	11.32%	6.16%
3' UTR	11.37%	4.30%
TF binding	11.28%	6.38%
Intron	10.45%	11.08%
Exon	11.87%	3.18%
5'UTR	11.84%	1.99%
Promoter	10.95%	9.78%
CTCF Binding Site	10.69%	7.36%

Supplementary Table 9: Proportions of the genomic elements in the dataset used for attention maps and embedding space visualization experiments.

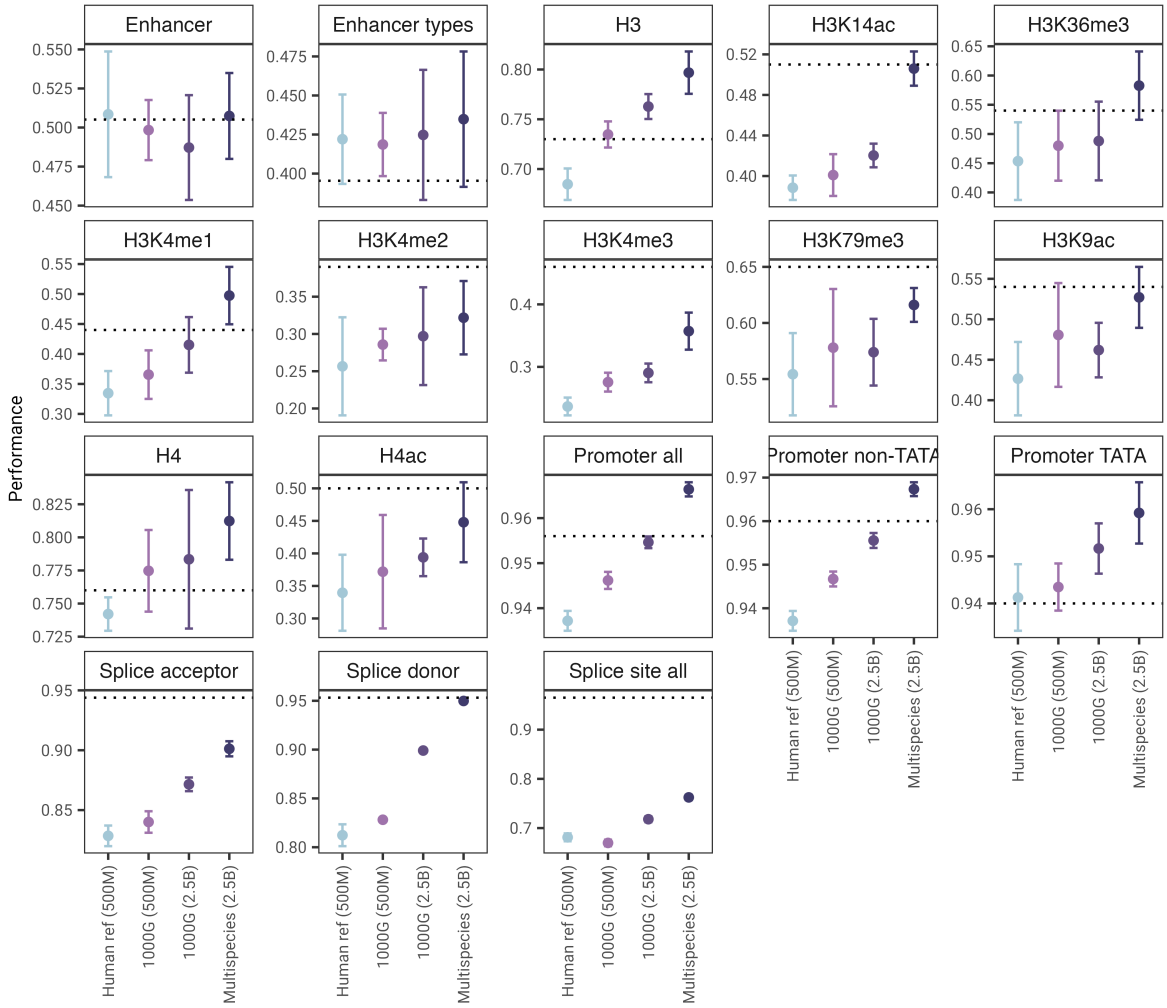
	CDS	CTCF	Enhancer	Exon	5'UTR	Intron	Open chromatin	Promoter	TF	3'UTR	No. heads
Human ref (500M)	40	26	27	33	37	37	27	33	26	37	480
1000G (500M)	52	27	24	48	41	40	26	55	27	49	480
Multispecies (2.5B)	89	35	36	72	42	117	44	50	74	51	640
1000G (2.5B)	38	32	28	40	33	61	40	55	45	54	640

Supplementary Table 10: Number of significant attention heads capturing gene features and regulatory elements. Note that the number of total heads varies between the 500M and 2.5B models.

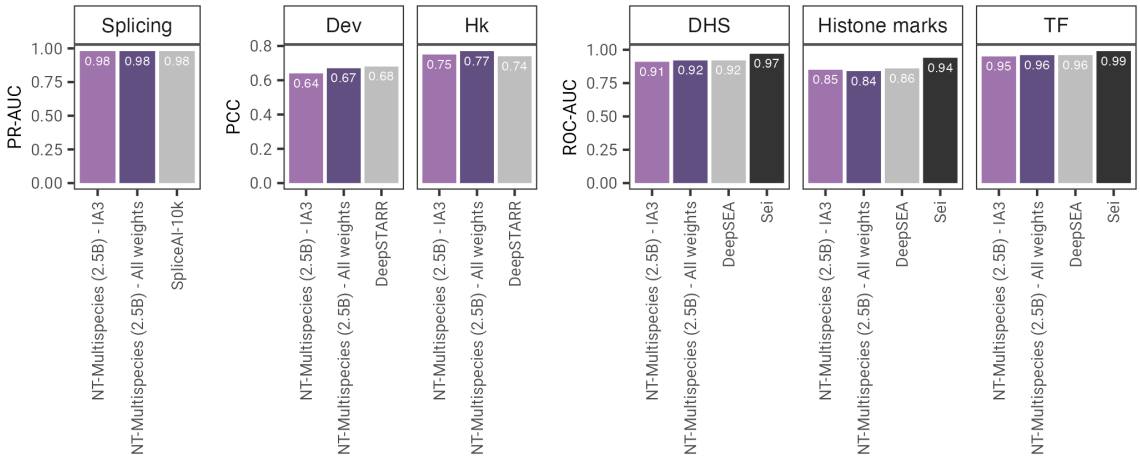
	CDS	CTCF	Enhancer	Exon	5'UTR	Intron	Open chromatin	Promoter	TF	3'UTR	No. heads
Human ref (500M)	40	26	27	33	37	37	27	33	26	37	480
1000G (500M)	52	27	24	48	41	40	26	55	27	49	480
Multispecies (2.5B)	89	35	36	72	42	117	44	50	74	51	640
1000G (2.5B)	38	32	28	40	33	61	40	55	45	54	640

Supplementary Table 11: Number of significant attention heads capturing gene features and regulatory elements. Note that the number of total heads varies between the 500M and 2.5B models.

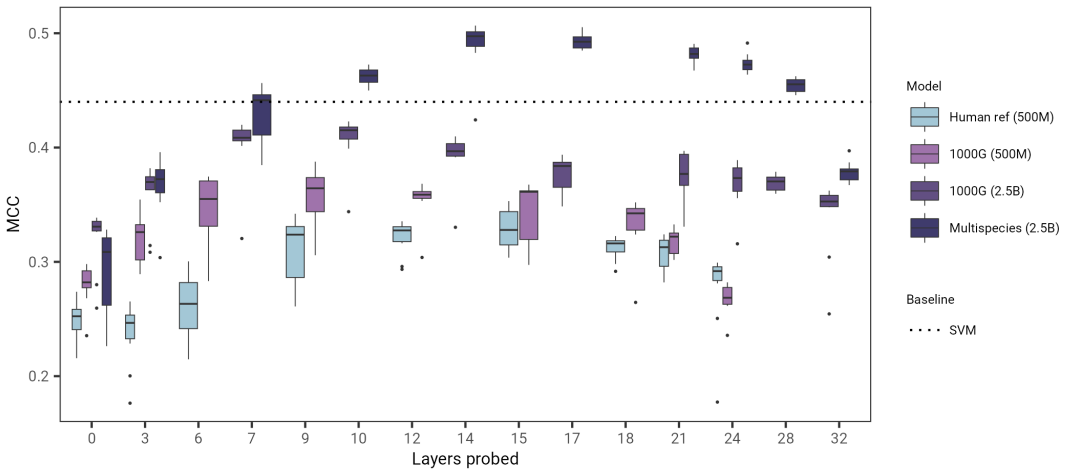
C Supplementary Figures



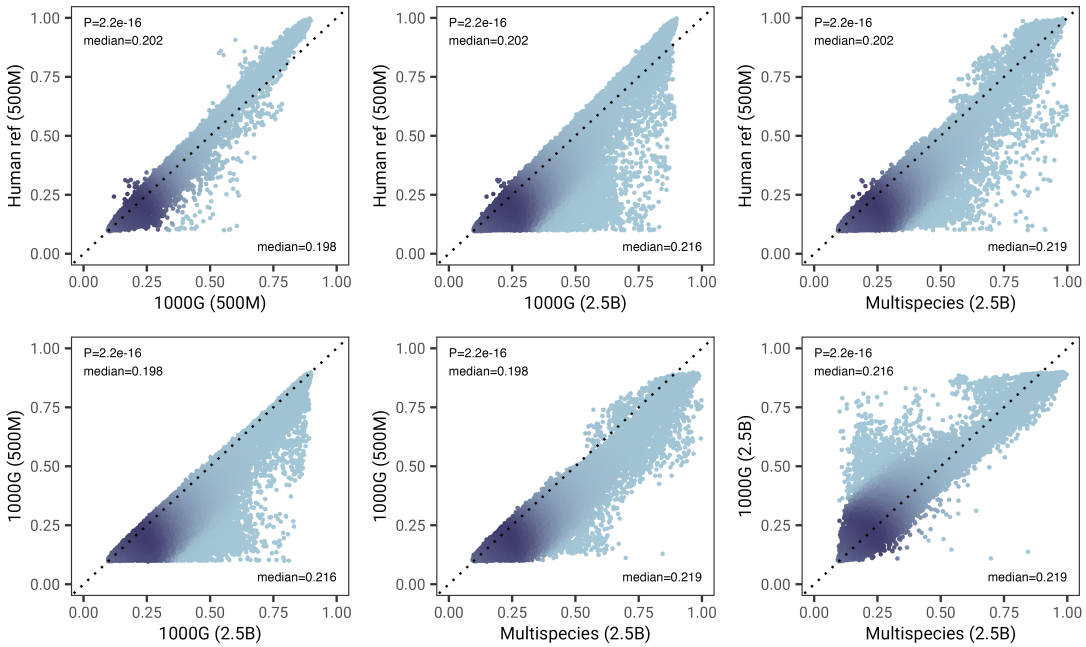
Supplementary Figure 1: Probing performance results on test sets across downstream tasks for each Nucleotide Transformer model. The performances of the best layers and best downstream models are shown. The error bars represent 2 std from the 10-fold cross-validation procedure.



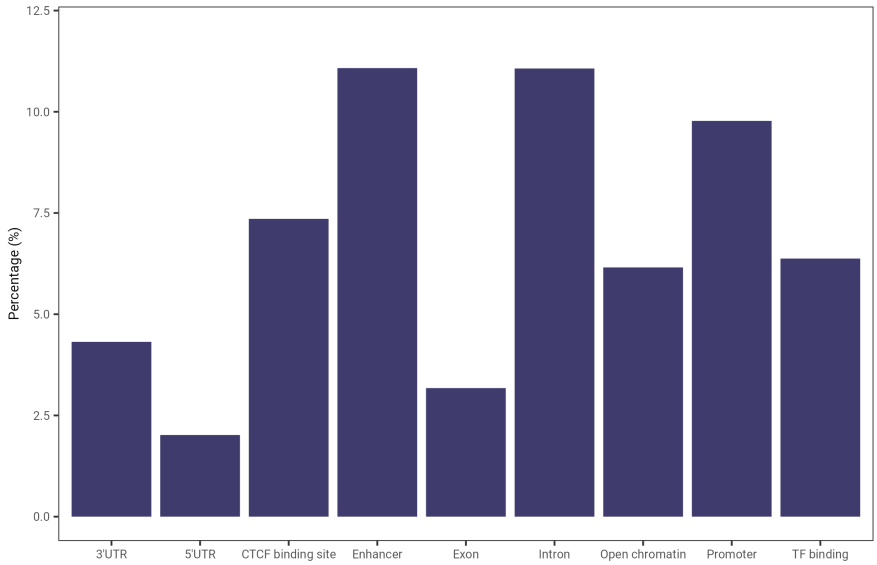
Supplementary Figure 2: Parameter efficient fine-tuning compared with full-model fine-tuning. The performances across different tasks are shown and compared with respective baselines.



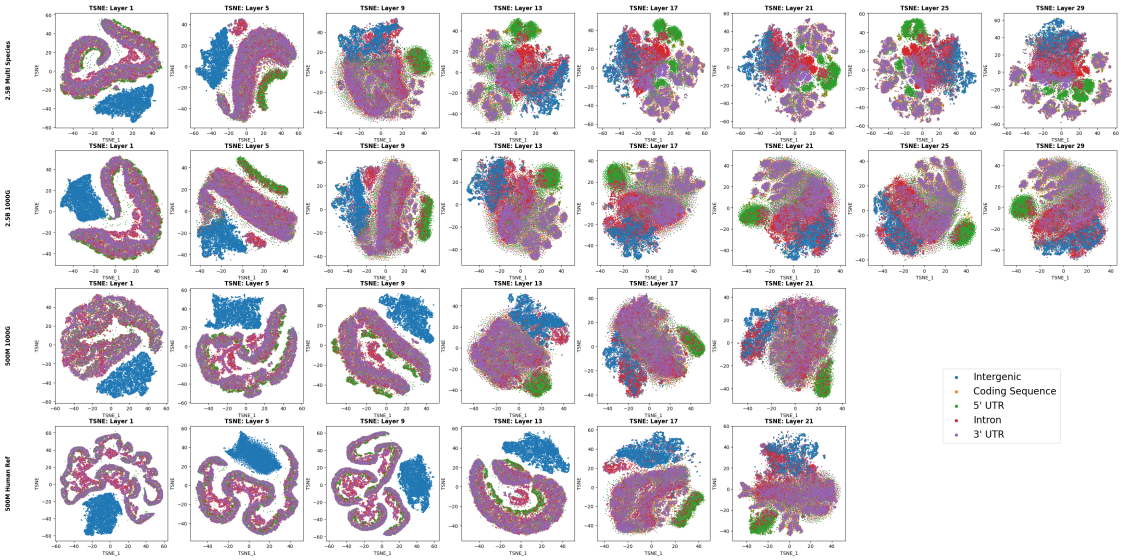
Supplementary Figure 3: Probing performance across layers for the H3K4me1 prediction task. Boxplots show the MCC values of 10 layers based on 10 fold cross-validation experiments.



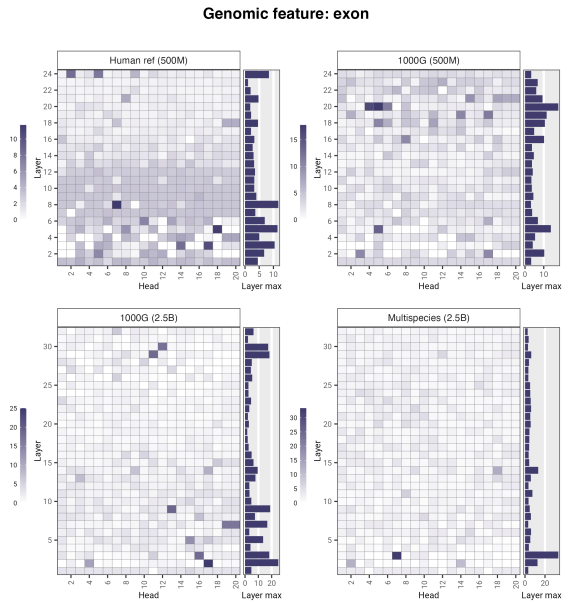
Supplementary Figure 4: Pairwise comparison of token reconstruction accuracy across transformer models. P-values refer to a two-sided Wilcoxon signed rank test between models. Median values for the two models compared are shown.



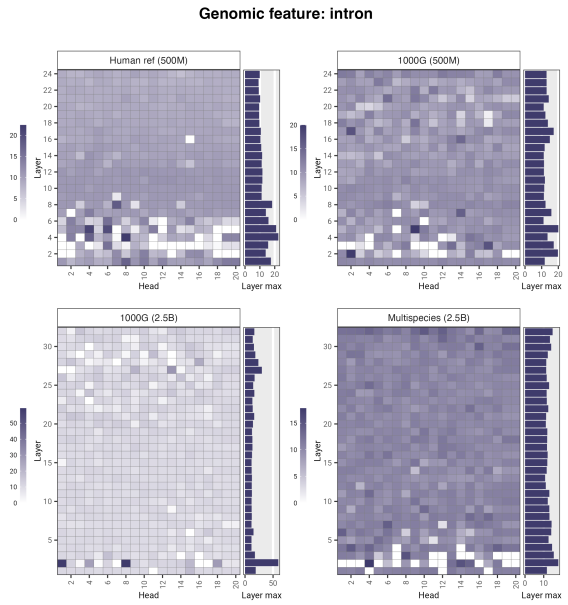
Supplementary Figure 5: Genomic element length's percentage of input sequence (6kbp). This corresponds to the dataset used for attention maps and embedding spaces visualization experiments. Details in Table 9.



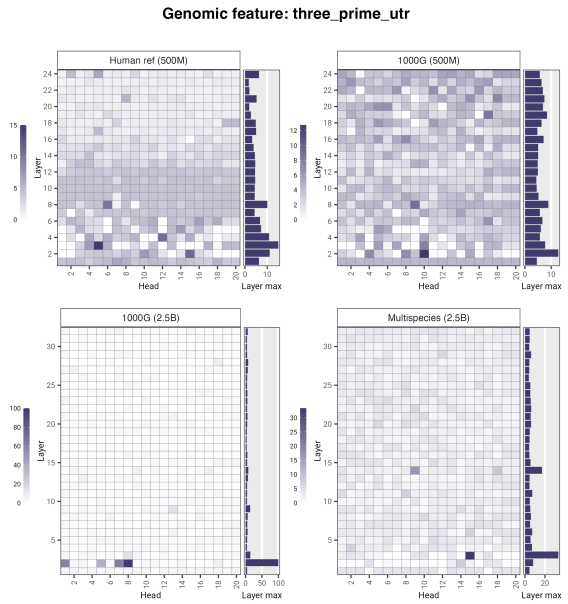
Supplementary Figure 6: t-SNE projections of embeddings of 5 genomic elements across transformer models and layers.



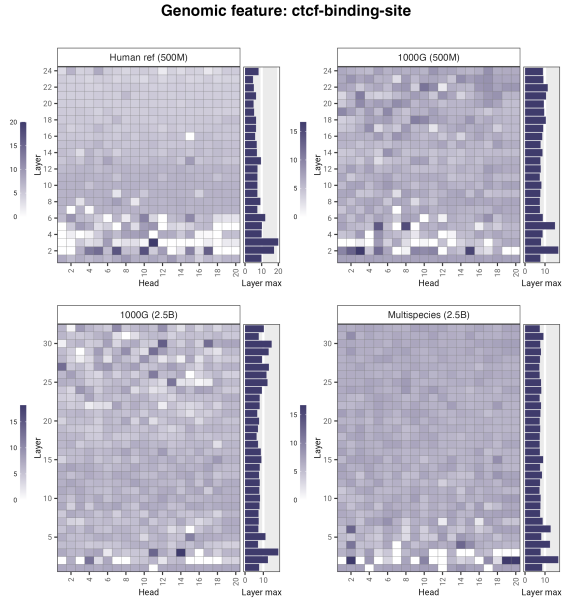
Supplementary Figure 7: Nucleotide Transformer models' attention percentages per head computed for exons.



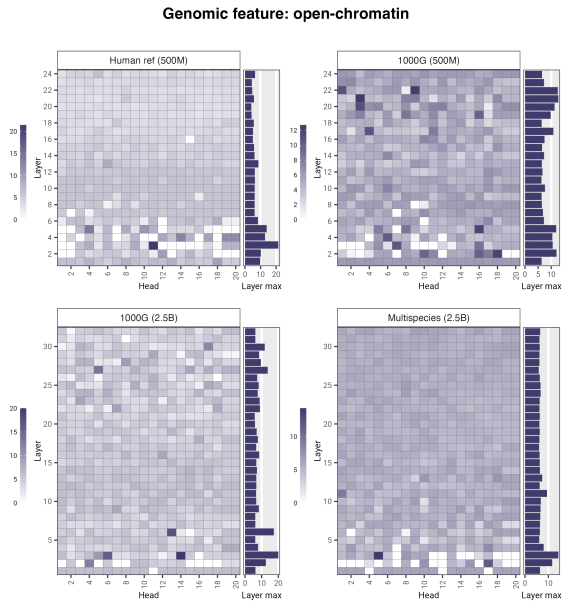
Supplementary Figure 8: Nucleotide Transformer models' attention percentages per head computed for introns.



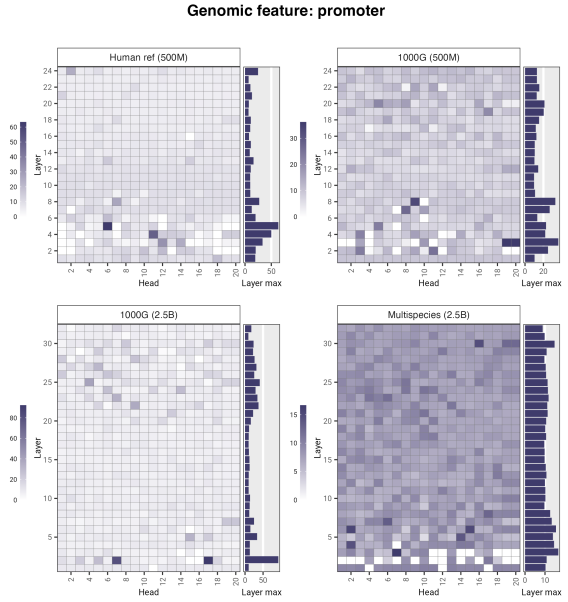
Supplementary Figure 9: Nucleotide Transformer models' attention percentages per head computed for 3' UTR regions.



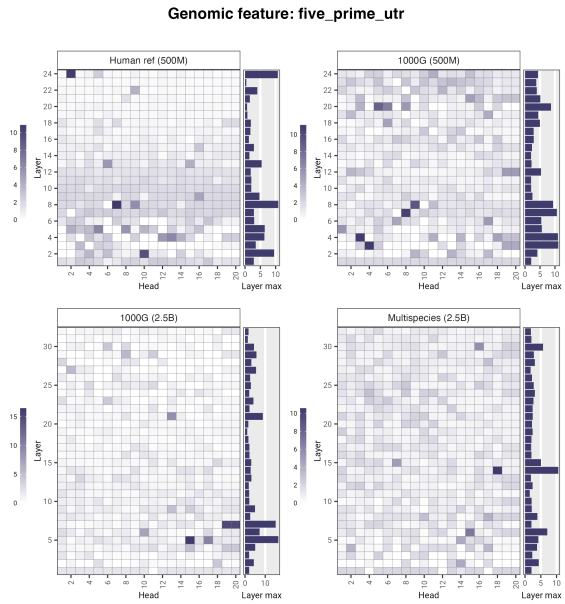
Supplementary Figure 10: Nucleotide Transformer models' attention percentages per head computed for CTCF binding sites.



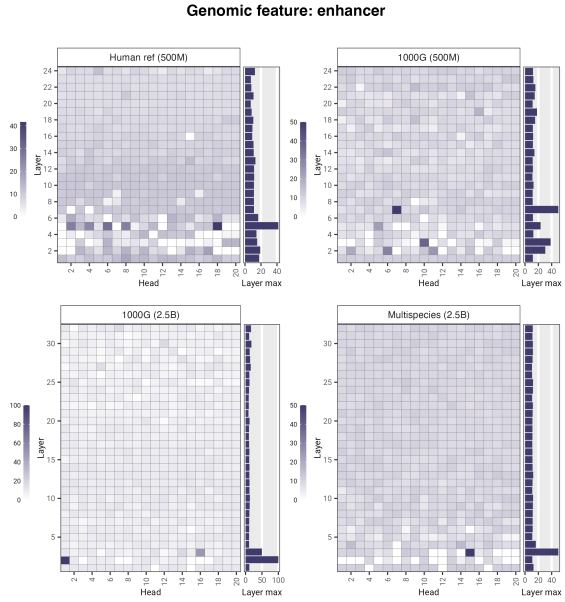
Supplementary Figure 11: Nucleotide Transformer models' attention percentages per head computed for open chromatin.



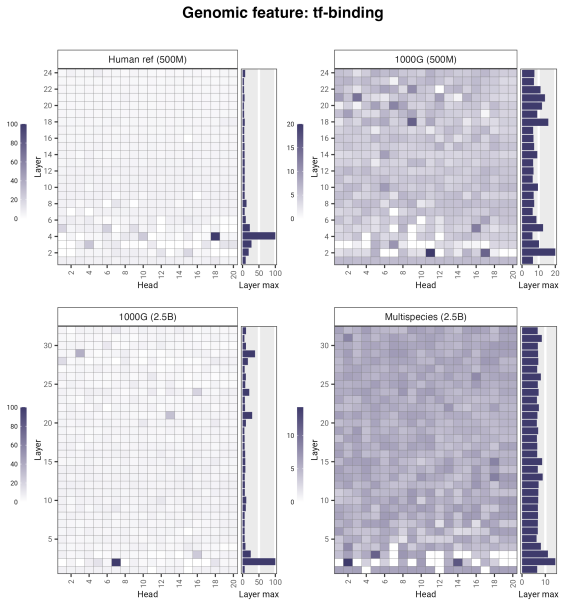
Supplementary Figure 12: Nucleotide Transformer models' attention percentages per head computed for promoters.



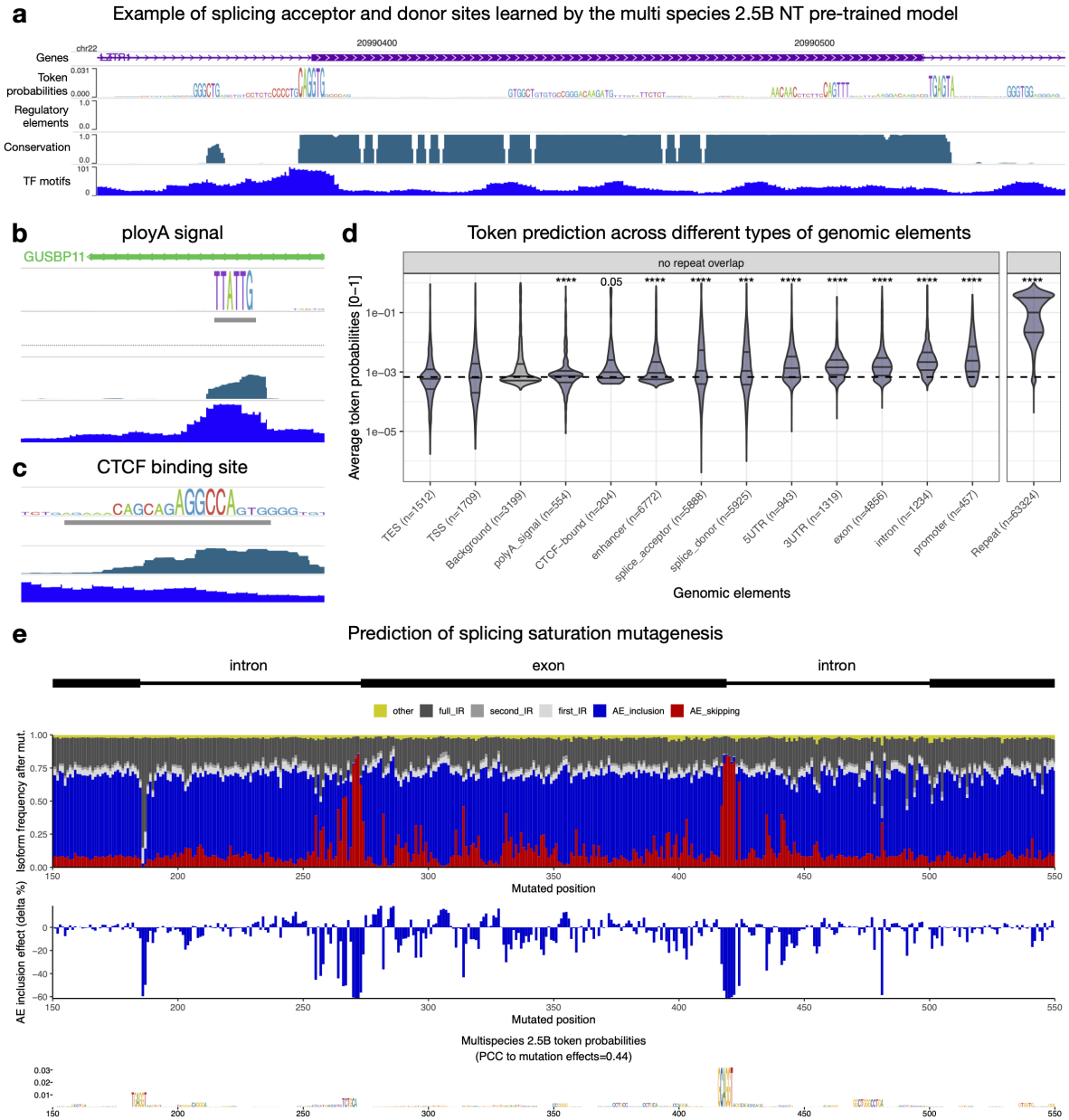
Supplementary Figure 13: Nucleotide Transformer models' attention percentages per head computed for 5' UTR regions.



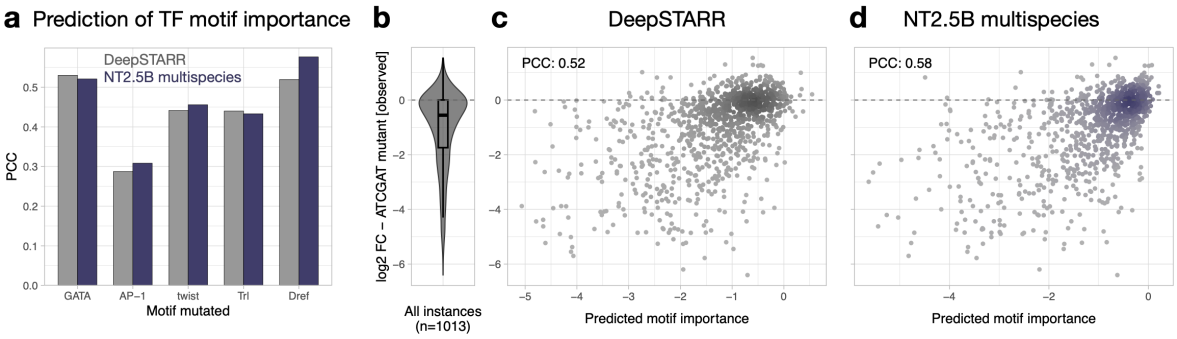
Supplementary Figure 14: Nucleotide Transformer models' attention percentages per head computed for enhancers.



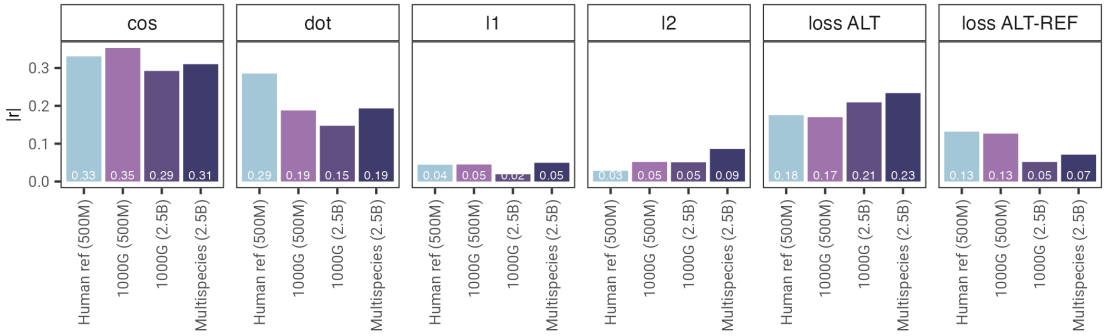
Supplementary Figure 15: Nucleotide Transformer models' attention percentages per head computed for transcription factor binding sites.



Supplementary Figure 16: Interpretation of Nucleotide Transformer multispecies 2.5B pre-trained model at higher resolution. **a-c)** Genome browser screenshot depicting token probabilities for **(a)** an exon region of gene LZTR1 (chr22:20990296-20990630), **(b)** a polyA signal from GUSBP11 3'UTR (chr22:23638476-23638540), and **(c)** a CTCF binding site (chr22:22254952-22254981). ENCODE regulatory elements, conservation, and Jaspar TF motifs are also shown. **d)** Distribution of average token probability scores across different types of genomic elements that do not overlap repeat elements (left) or across repeat elements (right). Violins are shown with horizontal lines marking the median and lower and upper quartile. ****P ¡ 0.0001, ***P ¡ 0.001, **P ¡ 0.01, *P ¡ 0.05, (two-sided Wilcoxon rank-sum test). **e)** Top: experimentally measured impact of mutating every nucleotide position in the different splicing isoforms of exon 11 of gene MST1R. Middle: effect in Alternative Exon (AE) 11 inclusion as delta percentage. Bottom: token probability predictions by the multispecies 2.5B pre-trained model. Pearson Correlation Coefficient (PCC) between token predictions and AE11 inclusion mutation effects is shown.



Supplementary Figure 17: Prediction of transcription factor motif importance. **a)** Bar plots showing the Pearson Correlation Coefficient (PCC) between predicted (by DeepSTARR or the multispecies 2.5B full-finetuned model) and observed log2FC for mutating individual instances of each motif type. **b-d)** Distribution of experimentally measured enhancer activity log2FC after mutating 1,013 different Dref motif instances across enhancers (violin plot), compared with the log2FC predicted by **(c)** DeepSTARR or **(d)** the multispecies 2.5B model.



Supplementary Figure 18: Performance of zero-shot based scores for SNPs annotations based on Ensembl Variant Effect Prediction (VEP). Performance is based on the Pearson correlation between the scores and severity as estimated by Ensembl.